International Journal of Smart Computing and Artificial Intelligence
International Institute of Applied Informatics
2018, Vol.2, No.1, P.59 - 78

A Machine-Learning Approach to Select Important Variables for Recombination on Many-objective Evolutionary Optimization

Miyako Sagawa [*] , Hernán Aguirre [*] , Fabio Daolio [*] ,
Arnaud Liefooghe [†] , Bilel Derbel [†] ,
Sébastien Verel [‡] , Kiyoshi Tanaka [*]

Abstract

There exist numerous many-objective real-world optimization problems in various application domains for which it is difficult or time-consuming to derive Pareto optimal solutions. In an evolutionary algorithm, variation operators such as recombination and mutation are extremely important to obtain an effective solution search. In this paper, we study a machine learning-enhanced recombination that incorporates an intelligent variable selection method. The method is based on the importance of variables with respect to the ranking of solutions in objective space that express convergence to the Pareto front. We verify the performance of the enhanced recombination on benchmark test problems with three or more objectives using the many-objective evolutionary algorithm AεSεH as a baseline algorithm. Our experimental analysis reveals that variable importance can effectively enhance the performance of many-objective evolutionary algorithms.

Keywords: Evolutionary computation, Machine learning, Multi-objective optimization, Many-objective optimization, Variable selection, Random forest.

1 Introduction

Multi-objective evolutionary algorithms (MOEAs), such as NSGA-II [1], SPEA2 [2] or NCGA [3], are able to derive a good approximation of the set of Pareto optimal

[*] Shinshu University, Nagano, Japan
[†] Univ. Lille, CNRS, UMR 9189 – CRIStAL / Inria Lille-Nord Europe, France
[‡] Univ. Littoral Côte d'Opale, LISIC, France

solutions (POS) for two- or three-objective optimization problems. However, it is difficult for MOEAs to derive an approximation of the set of POS on many-objective optimization problems (MaOPs). Recently, several many-objective evolutionary algorithms (MaOEAs) have been developed, such as MOEA/D [4], NSGA-III [5] or AεSεH [6]. Such MaOEAs have shown better convergence of the POS approximation set on many-objective test problems. Thus, it is expected that MaOEAs will contribute to solve real-world problems with four and more objectives. In industrial applications, there are cases where not all objectives have the same priority. Sometimes it is critical to find good compromises (in terms of Pareto efficiency) between a specific subset of all possible objectives. In those situation, the decision maker will be interested in speeding up the convergence towards those objectives.

Several methods have been proposed to help optimization algorithms solve difficult problems by learning the structure of the problem, namely the specific patterns of interactions among variables at the time of determining the objective function [7]. For example, estimation of distribution algorithms (EDAs) learn the problem structure during the solution search [8, 9, 10]. A new modeling approach that learns a joint model of objectives and variables, differentiating their role in the network, was proposed in [11] for MO-EDAs to generate new solutions. The method can capture not only the relationships between variables but also the relationships between objectives and variables, and the relationships between objectives. In addition, MOEA based on decision variable analysis was proposed in [12] to decompose variables into low dimensional subcomponents. The method estimates the role of variables and correlation between variables.

Our aim is to further improve the solution search ability of MaOEAs to find solutions faster, with good properties in terms of convergence and diversity. In an evolutionary algorithm, variation operators such as recombination and mutation are extremely important for effective solution search in order to solve complex optimization problems. In ordinary MOEAs, variables that undergo recombination or mutation are usually selected uniformly at random with respect to some user-defined probability. Thus, variables are not explicitly selected based on their contribution to improve the rank of solutions.

In this work, we propose a machine learning-enhanced method to select variables for recombination. Our method models the relationships between Pareto rank in objective space and variables in decision space to estimate important variables. This method uses random forest to derive variable importance (VI) according to the Pareto rank of solutions. During recombination, the values of VI are used to select the variables that should be recombined, aiming to find better solutions in the direction that improves their ranking towards the Pareto optimal front. We also investigate two sources of information for random forest to determine VI. One is the instantaneous population and the other one is the archive of solutions visited so far. The idea of learning about the problem while optimizing it, in order to improve the solution search, is not specific to multi-objective optimization. Instead, it belongs to the more general conceptual framework of "intelligent optimization". Potentially, the proposed method could be fruitful in any evolutionary scenario where solutions can be ranked according to a score, from which important search directions can

be derived. Thus, we expect that MaOEAs that adopt such intelligent optimization methods could be applied to MaOPs with an increased effectiveness to find solutions with good convergence.

In this work, we apply the proposed machine learning-enhanced method to optimization problems with up to six objectives. This method uses Pareto ranking to score solutions. In many-objective optimization problems, it is well known that the number of non-dominated solutions increases exponentially with the number of objectives [6]. Thus, a low variety of rankings in the population is expected, which could affect the estimation of variable importance. We are interested in assessing the scalability of Pareto ranking as a score to derive variable importance on many-objective problems, and verify whether we can improve the performance of MaOEAs. We analyze the impact of using the instantaneous population or the archive of solutions as a source of information for random forest, correlating with some features of multi- and many-objective optimization problems and the variety of Pareto rankings used as scores. An initial version of this paper, reporting experimental results applying this method to two-objectives problems, was presented in [13]. We use the multi- and many-objective optimizer AεSεH as a baseline algorithm. We include the proposed method into AεSεH and compare its performance with the baseline algorithm and an ideal algorithm, which knows in advance the variables related to convergence and selects from them for recombination.

In the remainder, we describe variable importance in Section 2, we introduce the considered algorithms in Section 3, we give our experimental setup in Section 4, we report our experimental analysis in Section 5, and we conclude in Section 6.

2 Method

We identify variables that affect Pareto improvement based on the value of variable importance. Random forest, a machine learning method, can calculate the relative importance of variables in predicting the score of solutions. In this work, we use the Pareto ranking induced by non-dominated sorting [14] as the score. Then, we apply recombination to variables which are identified to affect Pareto improvements. In order to calculate the ranking, non-dominated sorting can be applied to the combined current population of parents and offspring or to a sample of the archive of solutions visited so far. Figure 1 presents a schematic view of the overall algorithm including the proposed methods. The procedures within the shaded area express the conventional operations of an Evolutionary Multi-objective Optimization (EMO) algorithm and the procedures in the unshaded area the machine-learning related operations for variable selection. In the Construct Data Set procedure, we prepare the data to be submitted to random forest, which could be either the current population and its offspring or a sample from the archive. Then, the importance of each variable is derived by random forest, and it is used to select variables for recombination. This method can be regarded as a variable selection method for recombination, and can be easily added to any MOEA.

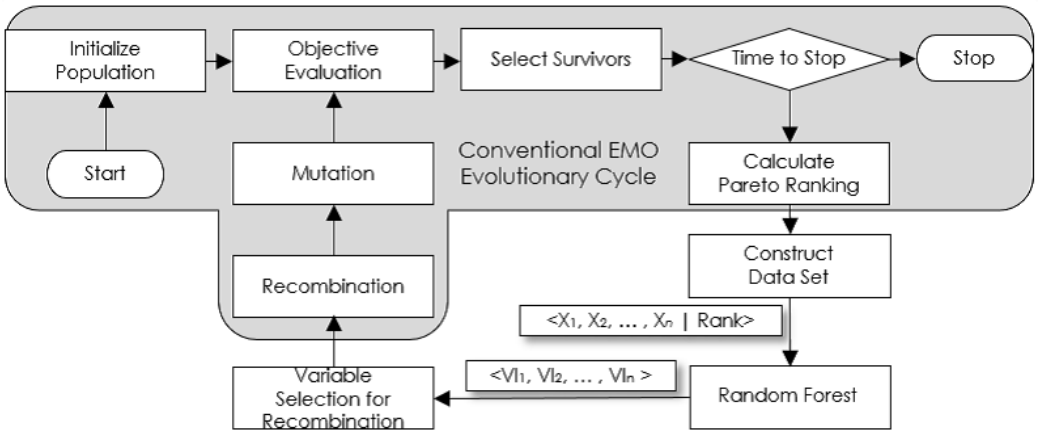

Figure 1: Overall concept.

2.1 Extracting Variable Importance

In this work, we use the random forest [15, 16] implementation in R [17] to extract variable importance. Random forest can use two variable permutation measures to determine variable importance. One is *raw importance* and the other one is *scaled importance* (z-score). We use the raw importance, as suggested in [18]. Let us define the number of trees as nt and an index of a tree as $t, (t = 1, 2, \cdots, nt)$. We provide random forest with a sample of solutions and their ranking. We denote this as original data P. In this work, P could be either the combined population of parents and offspring or a sample taken from the archive of solutions. In Section 3, we will describe in detail how the archive is actually sampled. To grow each tree of the forest, and to compute the variable importance in the tree, we apply the following procedure.

1. Split randomly the original data P into a learning set (two thirds of the original data) and a testing set (one third of the original data). Determine the training data set for growing a tree by randomly sampling from the learning data, allowing duplication, until picking a set with the same size as original data P.

2. Grow the tree by splitting the nodes. The most discriminative variable among m randomly-selected candidate variables is used to split a node.

3. Calculate the accuracy of the estimation using the testing data. We submit the variable values (input value) of the solutions in the testing data set to the tree and obtain the predicted class (Pareto rank). The learning error of the tree, denoted as LE^t, is calculated by finding the mean squared error (MSE) between the predicted class and original class in the testing data set.

4. Calculate the prediction error after permutation. For each variable $x_i, (i = 1, 2, \cdots, n)$, we permute the value of x_i among solutions in the testing data

set, while keeping the other variables $x_j, (i \not\equiv j)$ fixed. The solutions with the permuted values of x_i are submitted to the tree to get their predicted class. Then, we measure the MSE between the predicted class and the original class in the testing data set. Let us define the MSE as $EE^t_{x_i}$.

5. Compute variable importance in the tree. Variable importance for each variable $VI^t_{x_i}$ of a tree is calculated by the difference between learning error and prediction error of each variable.

$$VI^t_{x_i} = (EE^t_{x_i} - LE^t). \tag{1}$$

In order to get the overall variable importance on the forest for each variable VI_{x_i}, we calculate the average of the variable importance for each variable on all trees [16, 19].

$$VI_{x_i} = \Big(\frac{1}{nt}\sum_{t=1}^{nt} VI^t_{x_i}\Big). \tag{2}$$

This is called mean decrease in accuracy (MDA) or permutation importance.

2.2 Guiding Recombination

In general, a recombination operator selects some variables at random to recombine two parent individuals based on a probability set in advance. In this work, we select variables for recombination based on importance towards Pareto improvement. Using the result of deriving variable importance from random forest, we bias the probability to apply recombination towards the variables that have larger variable importance. We consider two ways of variable selection for recombination: probabilistic and deterministic. The deterministic approach sorts the variables in the order of importance and selects the $P_{cv} \times n$ most important ones. On the other hand, the probabilistic approach selects variables based on a probability that depends on the value of variable importance given by

$$P^{(i)}_{cv} = P_{cv}\frac{VI_i}{\sum_{j=1}^{i} VI_j}, \tag{3}$$

where $P^{(i)}_{cv}$ is the crossover probability of the i-th variable, P_{cv} is the overall crossover probability per variable, VI_i is the estimated importance of the i-th variable, and n is the total number of variables. In this work, we adopt the deterministic approach.

3 Algorithms

The concept described in the previous section could be potentially adopted by any MOEA. In this work, we compare the performance of three algorithms as follows.

3.1 Baseline Algorithm (orig)

We use the Adaptive $\varepsilon-$Sampling and $\varepsilon-$Hood (AεSεH) [6, 20] as a baseline algorithm. AεSεH is a population-based multi- and many-objective elitist evolutionary algorithm. It has two important features: the $\varepsilon-$Hood method used to select parents for recombination, and the $\varepsilon-$ Sampling method used for survival selection. We use the SBX crossover [21] as a recombination operator applied with a rate p_c per individual and p_{cv} per variable. The performance of AεSεH is similar or better than NSGA-II on different MOPs [20], and it shows good performance for many-objective optimization [6].

3.2 Recombination applied to Convergence-Related Variables (ideal)

Let us assume that the algorithm is aware of which variables are related to convergence. We modify the baseline algorithm in order to take advantage of this information, and apply recombination to the variables that determine the distance to the Pareto front. This corresponds to a *cheating* algorithm having a perfect knowledge of the variables that are important to get closer to the Pareto front. This algorithm is expected to show the best search ability in terms of convergence. This approach, denoted ideal, will allow us to appreciate the convergence that can be achieved with a given recombination operator that perfectly learns variable importance.

Let n be the number of variables for the problem under consideration, n_d the number of distance-related variables, n_p the number of position-related variables, and p_{cv} the probability of crossover per variable. If $p_{cv} \times n \leq n_d$, $p_{cv} \times n$ variables are selected randomly from the subset of distance-related variables. On the other hand, if $p_{cv} \times n > n_d$, all distance-related variables are selected for crossover and the remaining $p_{cv} \times n - n_d$ are selected at random from the subset of n_p position-related variables.

3.3 Recombination based on Variable Importance (VI) using Instantaneous Population

We include the method that guides recombination into the baseline algorithm Aε Sε H, as illustrated in Figure 1, using at each generation the combined population of par-ents and offspring to construct the data set to be submitted to random forest. As described in Section 2.2, we obtain the estimated variable importance for Pareto improvement from the random forest machine learning model, and we select for recombination the variables that have higher importance. Constructing the data set from the current population allows us to select variables that are important at a given particular state of the search process. This could be relevant in problems where variables do not have the same importance throughout the generations.

3.4 Recombination based on Variable Importance using Archive (VI_Arc)

The flow of algorithm VI_Arc is mostly similar to VI, shown in Figure 1. The difference is that in VI_Arc we construct the data set to be submitted to random forest from a sample of solutions taken from the archived populations. In this algorithm, at each generation, we add the current population to an archive. In order to form the sample, we select an equal number of solutions from each one of the populations in the archive. Thus, the history of evolution from the initial to the current generation is used to train the random forest model. In the following, we describe precisely how the sample is taken.

Let us define the size of the sample set for random forest as N_{RF}. We select probabilistically around $K = N_{RF}/j$ individuals from each population in the archive, where j is the index of the current generation. The probability of selection from an archived population is $P_{select} = N_{RF}/|A_j|$, where $|A_j|$ denotes the archive size at generation j.

When we pick a solution from an archived population, we select it based on the median rank $\hat{R}$ of solutions in order to prevent selecting individuals around the axis (extreme points). The median individual is calculated for each objective function. Ranking of individuals is based on the order of the distance from the median individual. Thus, an individual closer to the median individual is given a better rank (smaller value). Calculation of median rank for each individual is as follows.

1. Sort in descending order of objective function f_i using index $k \geq 0$.
2. Calculate median rank $\hat{R}_i$ for the i-th objective f_i using the archived population size $|P_t|$ at the t-th generation as follows

$$\hat{R}_i = \left| \frac{|P_t|}{2} - (k+1) \right|. \tag{4}$$

3. Repeat the above operations for all M objective functions.
4. Aggregate the median rank $\hat{R}$ over all objective functions.

$$\hat{R} = \sum_{i=1}^{M} \hat{R}_i. \tag{5}$$

We select individuals which have a better median rank in order to form the sample set. Once the sample is collected, we apply a non-dominated sorting to the selected sample set and give each individual a Pareto rank. In this paper, the size of the sample set N_{RF} is set to 500. For the first t generations, where the archive size is smaller than the sample size, $|A_t| < N_{RF}$, we use all solutions in the archive as the sample set.

4 Experimental Setting

4.1 Benchmark Problems

We use the well-known DTLZ1, DTLZ2 and DTLZ3 continuous test functions [22] to study the performance of the algorithms. These functions are scalable in the number of objectives M and in the total number of variables n. DTLZ1 has a linear Pareto-optimal surface, whereas DTLZ2 and DTLZ3 have a non-convex Pareto-optimal surface that lies inside the first quadrant of the unit hyper-sphere. The objective functions in DTLZ2 are unimodal, whereas in DTLZ1 and DTLZ3 they are multimodal. Thus, DTLZ1 and DTLZ3 introduce a large number of local Pareto-optimal fronts in order to test the convergence ability of the algorithm. A summary of the main features of each problem is shown in Table 1.

Table 1: Features of DTLZ problems.

	Separability	Modality	Geometry
DTLZ1	S	M	Linear
DTLZ2	S	U	Concave
DTLZ3	S	M	Concave

DTLZ problems have $M-1$ position-related variables that determine the spread along the front and $n-(M-1)$ distance-related variables that affect convergence towards the Pareto front. We set the number of objectives to $M = 3,4,5,6$ and the total number of variables to $n = (M-1)+9$. The combination of the number of position-related variables n_p and distance-related variables n_d are set to $(n_p, n_d) = (M-1, 9)$. In the following we describe in detail first the multimodal problems DTLZ1 and DTLZ3 and then the unimodal DTLZ2 and a variation of it that we denote as Modified-DTLZ2.

The DTLZ1 problem can be described as follows.

$$\begin{aligned}
\text{minimize } f_1(x) &= \frac{1}{2}x_1x_2\cdots x_{M-1}(1+g(x_M)) \\
\text{minimize } f_2(x) &= \frac{1}{2}x_1x_2\cdots(1-x_{M-1})(1+g(x_M)) \\
&\vdots \qquad\qquad \vdots \\
\text{minimize } f_M(x) &= \frac{1}{2}(1-x_1)(1+g(x_M)) \\
&\quad 0 \le x_i \le 1, \text{ for } i = 1,2,\cdots,n \\
\text{where } g(x_M) &= 100[|x_M| + \sum_{x_i \in x_M}(x_i-0.5)^2 - \cos(20\pi(x_i-0.5))]
\end{aligned} \tag{6}$$

The DTLZ3 problem can be described as follows. This formulation has (3^k-1) local Pareto-optimal fronts, and one concave global Pareto-optimal front when $x_M = (0.5, 0.5, \cdots, 0.5)^T$.

$$\begin{aligned}
\text{minimize } f_1(x) &= (1+g(x_M))\cos(x_1\pi/2)\cdots\cos(x_{M-1}\pi/2) \\
\text{minimize } f_2(x) &= (1+g(x_M))\cos(x_1\pi/2)\cdots\sin(x_{M-1}\pi/2) \\
&\vdots \quad\quad \vdots \\
\text{minimize } f_M(x) &= (1+g(x_M))\sin(x_1\pi/2) \\
&\quad 0 \le x_i \le 1, \text{ for } i = 1,2,\cdots,n \\
\text{where } g(x_M) &= 100[|x_M| + \sum_{x_i \in x_M} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))]
\end{aligned} \tag{7}$$

In DTLZ3 and DTLZ1, it is difficult to find global optimal solutions because of its g function.

The DTLZ2 problem can be described as follows.

$$\begin{aligned}
\text{minimize } f_1(x) &= (1+g(x_M))\cos(x_1\pi/2)\cdots\cos(x_{M-1}\pi/2) \\
\text{minimize } f_2(x) &= (1+g(x_M))\cos(x_1\pi/2)\cdots\sin(x_{M-1}\pi/2) \\
&\vdots \quad\quad \vdots \\
\text{minimize } f_M(x) &= (1+g(x_M))\sin(x_1\pi/2) \\
&\quad 0 \le x_i \le 1, \text{ for } i = 1,2,\cdots,n \\
\text{where } g(x_M) &= \sum_{x_i \in x_M} (x_i - 0.5)^2
\end{aligned} \tag{8}$$

In DTLZ2 it is not difficult to find solutions with good convergence because of its g function. In this problem, even random solutions are very close to the true POS. In this work, in addition to the conventional DTLZ2, we use a modified version of it by replacing function g as follows

$$g'(x_M) = 1000 \times \sum_{x_i \in x_M} (x_i - 0.5)^2 . \tag{9}$$

We denote this problem as Modified-DTLZ2. In this problem non-optimal solutions are further apart in objective space than in the original DTLZ2. The problem, however, remains separable and uni-modal.

4.2 Parameter Setting for MOEA

The number of generations in the algorithms is set to 2000, and the population size is set to 100 individuals. The distribution exponents are set to $\eta_c = 15$ for SBX and $\eta_m = 20$ for polynomial mutation operator. The crossover probability per individual is set to $P_c = 1.0$ and the crossover probability for each variable is $P_{cv} = 0.5$. The mutation probability is set to $P_m = 1/n$. We report results collected from 30 independent runs.

4.3 Parameter Setting for Random Forest

In the random forest procedure, we set some parameters based on the values recommended in [16]. The number of trees to grow is set to 500 and the number of variables randomly sampled as candidates at each split is set to $n/3$. We calculate the raw value of mean decrease in accuracy for variable importance, as mentioned in Section 2.1.

4.4 Metrics

In order to evaluate the search ability of the algorithms we use an archive that keeps all non-dominated solutions found through the generations. We calculate Generational Distance (GD) [23] to evaluate convergence of the population, and Inverted Generational Distance (IGD) [24] to evaluate diversity. For IGD, we use a reference set of 100,000 solutions for each problem.

5 Simulation Results

5.1 Recombination based on VI using the Instantaneous Population

We applied the three algorithms `ideal`, `orig` and `VI` described in Section 3 to DTLZ1, DTLZ3, DTLZ2 and Modified-DTLZ2 using 3, 4, 5 and 6 objective functions.

In the following we first discuss results on the multimodal problems DTLZ3 and DTLZ1. Figure 2 shows GD (on top) and IGD (at the bottom) values obtained by the three algorithms on DTLZ3. In the top of the graphs we indicate the number of objectives, number of variables, and the combination of position- and distance-related variables described above. The algorithms are shown in red, green and blue lines and are labeled `ideal`, `orig` and `VI`, respectively. Looking at Figure 2, it can be seen that `ideal` achieves the best GD-values through generations, whatever the number of objectives. This is expected because `ideal` represents an algorithm with a perfect model for variables related to convergence. We can see that the method `VI`, which learns online which variables are important to improve convergence and emphasizes their recombination, has better GD-values than the baseline algorithm `orig` after 500 generations. At the 2000 generation, `VI` obtains significantly better GD than the `orig` algorithm. It can also be seen that `orig` has slightly better or nearly equal IGD-values than `VI` in all objectives.

Figure 3 shows the VI-values of each variables obtained by `VI` at different generations. Results are shown for the 5-objectives DTLZ3 problem. We pick snapshots after 10, 100, 500, 1000, 15000, 2000 generations. The number of generations is shown on the top of each graph. From generations 10 and 100, note that no clear difference can be seen on the values of VI between distance-related variables $x_4 \sim x_{13}$ and position-related variables $x_1 \sim x_4$ at the beginning of the search. However af-ter 500 generations, distance-related variables have larger variable importance than position-related variables. This shows that the regression model in random forest

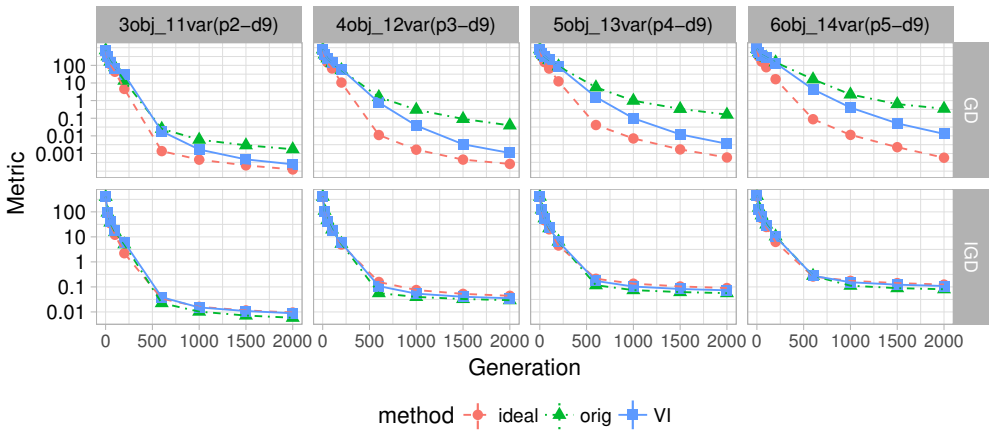

Figure 2: GD (top) and IGD (bottom) on DTLZ3.

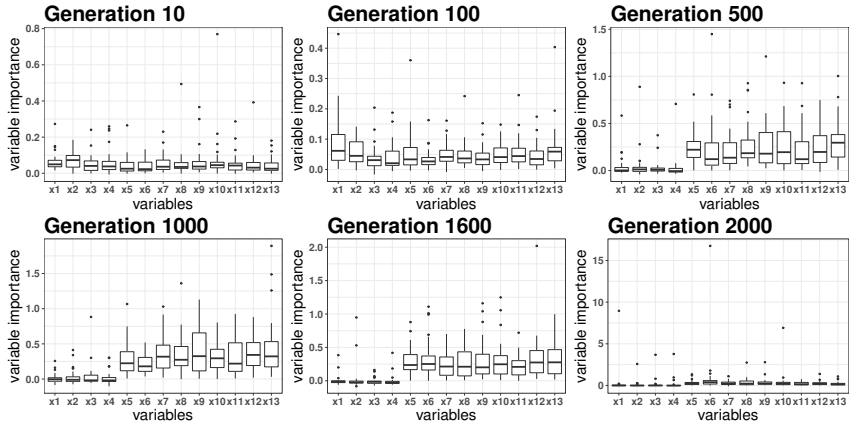

Figure 3: VI on DTLZ3, 5 objectives.

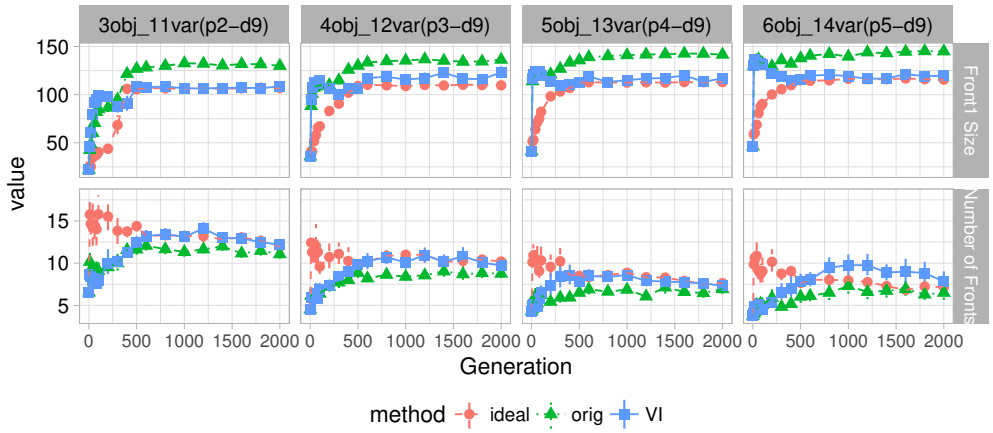

Figure 4: Size of Front 1 (top) and number of fronts (bottom) on DTLZ3.

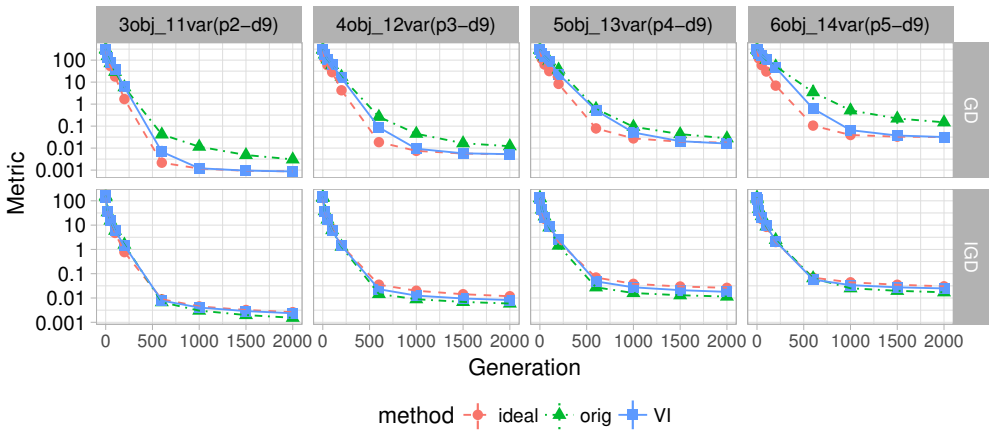

Figure 5: GD(top) and IGD(bottom) on DTLZ1.

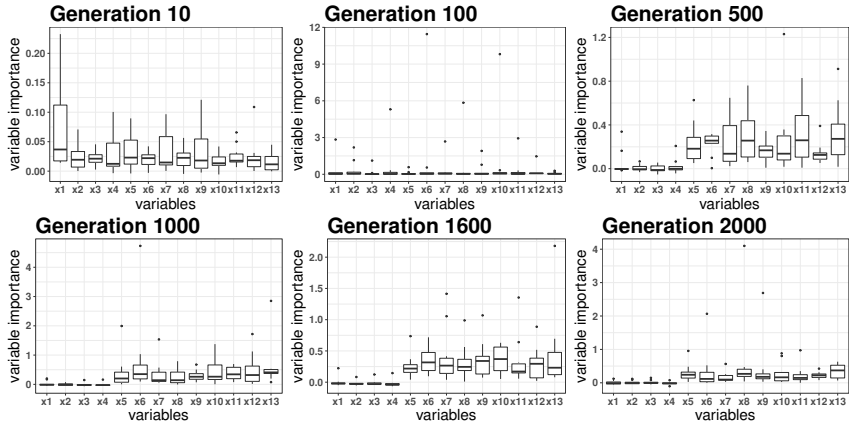

Figure 6: VI on DTLZ1, 5 objectives.

Figure 7: Size of Front 1 (top) and number of fronts (bottom) on DTLZ1.

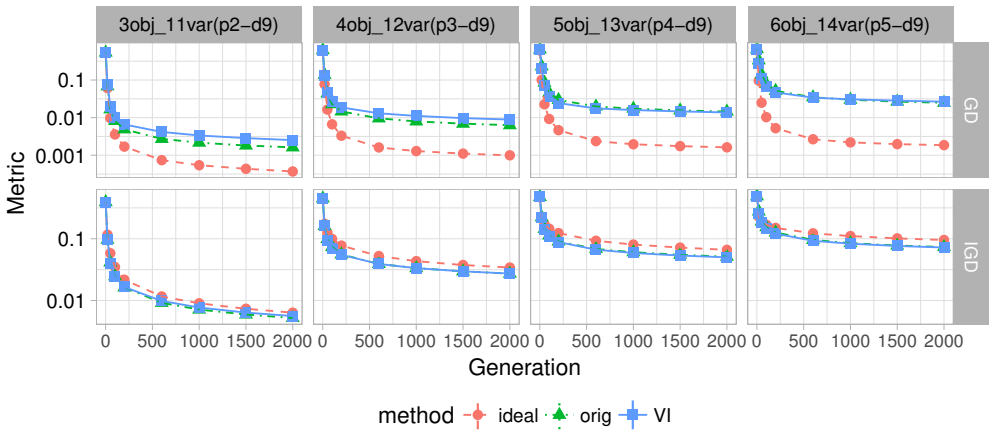

Figure 8: GD(top) and IGD(bottom) on DTLZ2.

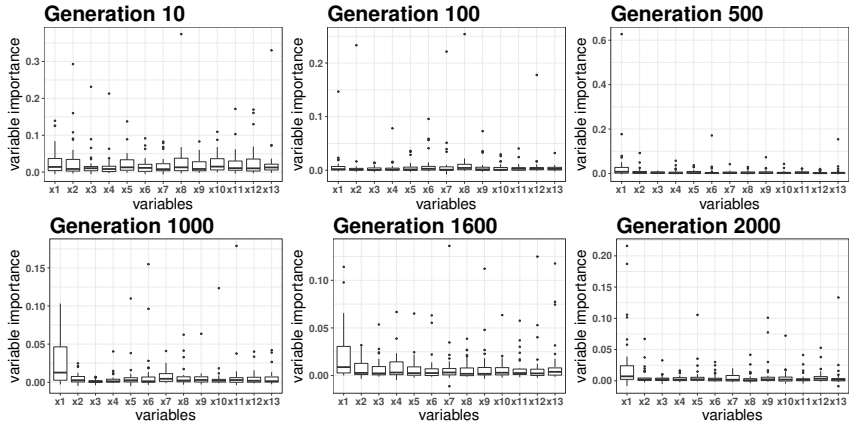

Figure 9: VI on DTLZ2, 5 objectives.

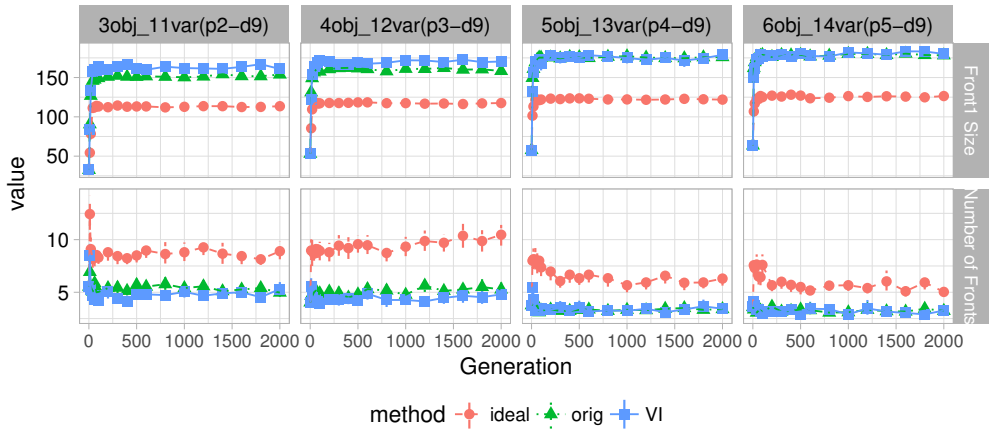

Figure 10: Size of Front 1 (top) and number of fronts (bottom) on DTLZ2.

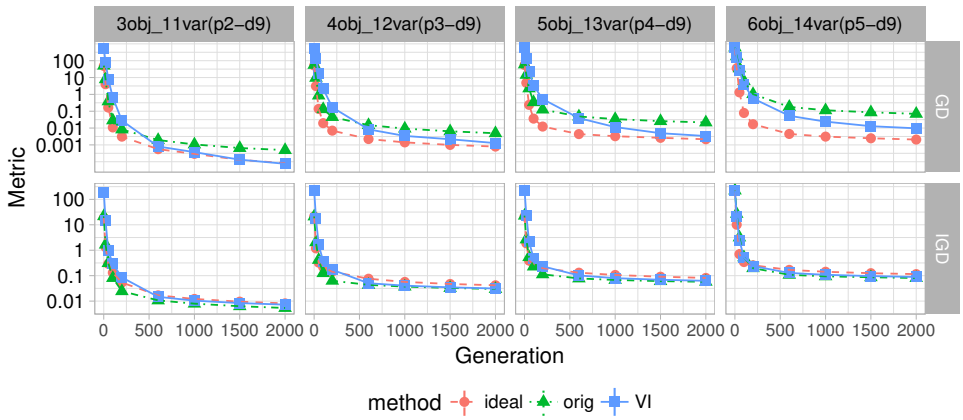

Figure 11: GD(top) and IGD(bottom) on modified DTLZ2.

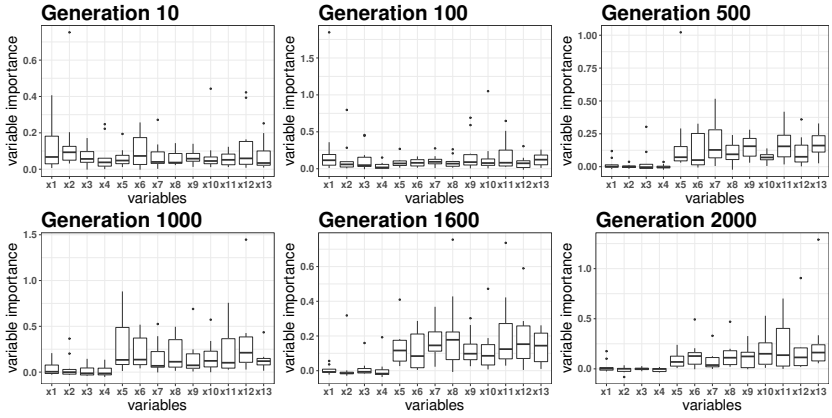

Figure 12: VI on modified DTLZ2, 5 objectives.

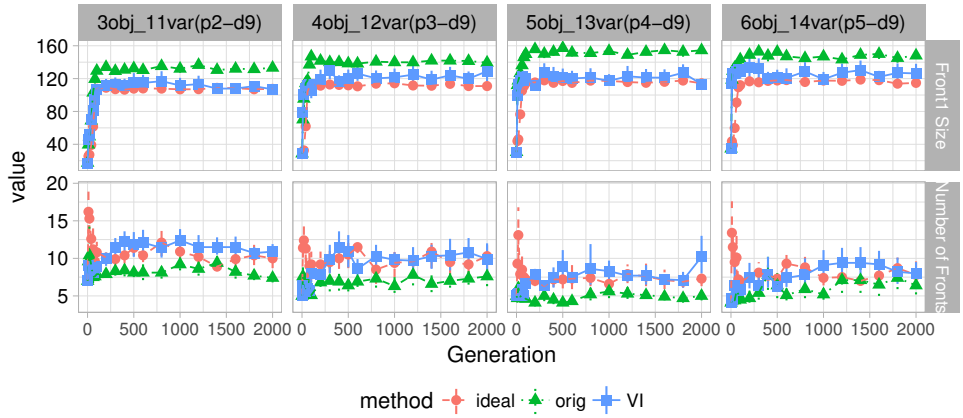

Figure 13: Size of Front 1 (top) and number of fronts (bottom) on modified DTLZ2.

is able to correctly distinguish between distance- and position-related variables. In the evolution process of VI algorithm, the variables with a larger VI-values get a higher chance to be recombined. Therefore, we can find solutions which have a better convergence in the objective space by giving high recombination opportunity to distance-related variables. Figure 4 shows the size of the first front and the number of fronts in the combined population of parents and offspring. This combined population is the one submitted to random forest and used for estimation of variable importance. From these figures, we note that the size of the first front in VI and ideal are similar and smaller than orig after 500 generations. Likewise, the number of fronts in VI and ideal are similar and larger than orig. So VI has at least 7 fronts after 500 generation in all cases.

Figure 5 shows GD- and IGD-values obtained by the three algorithms on DTLZ1. It can be seen that VI has a better GD-value than orig and almost the same as ideal after 1000 generations on all objectives. Figure 6 shows the VI-values of each variable obtained by VI. Similar to DTLZ3, distance-related variables have larger variable importance than position-related variables after 500 generations, although there are no clear differences on VI-values at the beginning of the search process. Looking at Figure 7, we can see that the size of the first front in VI and ideal are smaller than orig after 500 generations and the number of fronts in VI and ideal are larger than orig.

Let us now focus our discussion on the unimodal problems DTLZ2 and Modified-DTLZ2. Figure 8 shows the GD- and IGD-values obtained by the three algorithms on DTLZ2. We can see that VI has worse or similar GD- and IGD-values than orig, but ideal obtained significantly better GD-values. Figure 9 shows the VI-values of each variable obtained by VI. Note that there is no difference in VI-values between position- and distance-variables through all generations. This means that VI could not be learned in this problem, and therefore we could not increase the chance to apply recombination to distance-variables. Looking at Figure 10, we can see that the size of the first front in VI is above 150, and greater than orig since early generations. Likewise, the number of fronts in the combined population is around 5 in the three-objective problem and around 2 in the six-objective problem, similar or smaller than orig. On the other hand, the size of the first front in ideal is just above 100, and the number of fronts is significantly larger. We can see that the population submitted to random forest has few fronts, and therefore the different ranks were too few to accurately estimate variable importance towards the Pareto optimal front.

We applied the same algorithms to Modified-DTLZ2, as described in Section 4. Figure 11 shows the GD- and IGD-values obtained by the three algorithms. We can see that VI has better GD-values than orig after 500 generations, whatever the number of objectives, and that it approaches ideal as evolution progresses on 3-, 4- and 5-objectives problems. On the 6-objective problem, GD-values by VI are better than orig but do not approach ideal. Figure 12 shows the VI-value of each variable obtained by VI on the 5-objective Modified-DTLZ2 problem. Note that, after 500 generations, the VI-value of distance-related variables $x_5 \sim x_{13}$ is higher than position-related variables $x_1 \sim x_4$. Therefore, similar to DTLZ3, distance-related

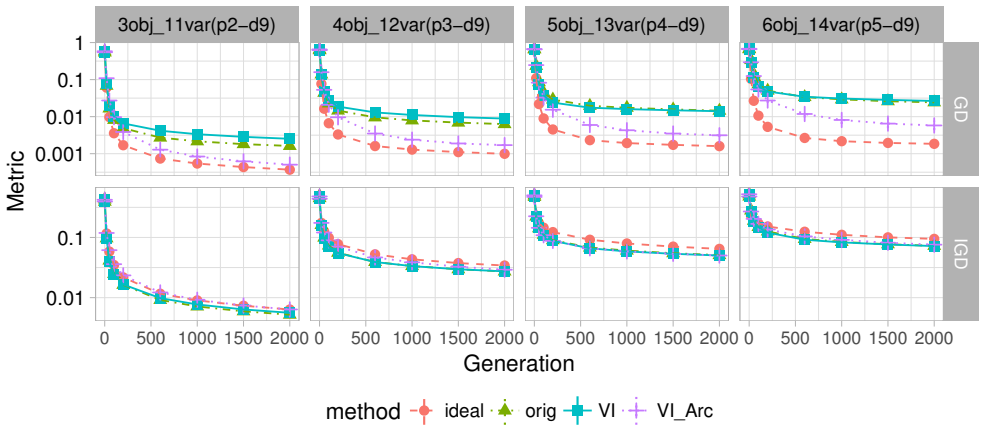

Figure 14: GD(top) and IGD(bottom) on modified DTLZ2 comparing with `VI_Arc`.

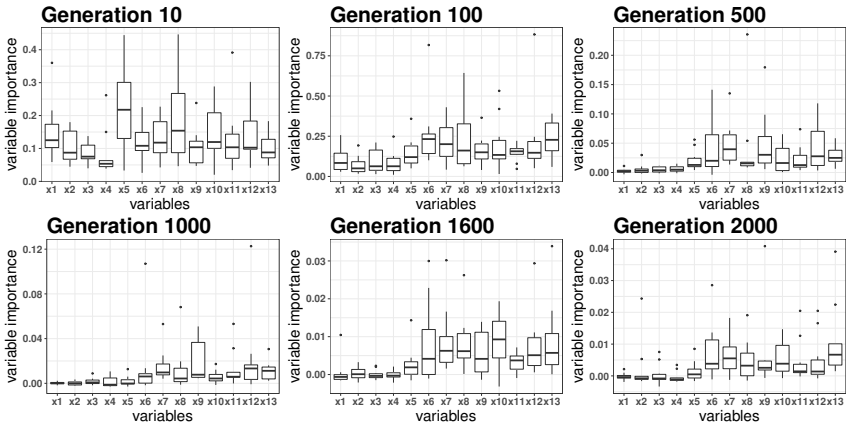

Figure 15: VI on modified DTLZ2, 5 objectives computed by `VI_Arc`.

variables get the chance to be recombined more often, which improves convergence. Figure 13 shows the size of the first front and the number of fronts in the combined population. Looking at Figure 13, we can see that the size of the first front in `VI` is around 120, and similar to `ideal`. Note also that the number of fronts by `VI` on Modified-DTLZ2 is larger than in DTLZ2. In Modified-DTLZ2 there is a more clear separation between local fronts than in DTLZ2 and the population contains solutions with a larger variety of rankings. In this case, random forest could estimate VI properly and could accurately identify distance-related variables for recombination.

Table 2 reports the average GD-values obtained by `orig`, `VI` and `ideal` algorithms on DTLZ1, DTLZ2, modified-DTLZ2 and DTLZ3, computed at the last generation of the 30 runs. Standard deviations are shown in parenthesis. We show results in bold if there is any significant difference in result between `orig` and `VI`, based on a Mann-Whitney non-parametric statistical test with a p-value of 0.05.

Table 2: Comparison of algorithms (org, VI, and ideal) on DTLZ1, DTLZ2, modified-DTLZ2 and DTLZ3 with respect to generational distance (GD). The first value is the average indicator-value, the second value in parentheses is the standard deviation. Any statistical difference between org and VI is shown in **bold**. ($\times 10^{-2}$)

		3obj	4obj	5obj	6obj
DTLZ1	orig	0.40 (0.32)	1.5 (1.3)	2.8 (0.49)	20 (19)
	VI	**0.087** (3.8e-3)	**0.53** (0.035)	**1.6** (0.060)	**3.2** (0.35)
	ideal	0.088 (0.011)	0.53 (0.027)	1.7 (0.23)	3.2 (0.13)
DTLZ2	orig	**0.16** (5.6e-3)	**0.63** (0.020)	1.4 (0.045)	**2.5** (0.11)
	VI	0.25 (0.018)	0.89 (0.044)	**1.4** (0.068)	2.7 (0.12)
	ideal	0.037 (1.2e-3)	0.10 (3.4e-3)	0.16 (4.9e-3)	0.19 (6.0e-3)
Modified -DTLZ2	orig	0.050 (6.5e-3)	0.50 (0.091)	2.2 (0.25)	7.2 (1.9)
	VI	**9.5e-3** (8.9e-3)	**0.16** (0.16)	**0.35** (0.10)	**0.99** (0.21)
	ideal	8.2e-3 (8.4e-4)	0.080 (5.7e-3)	0.22 (0.012)	0.21 (0.030)
DTLZ3	orig	0.26 (0.26)	6.2 (6.0)	21 (17)	42 (26)
	VI	**0.041** (0.072)	**0.38** (0.90)	**1.1** (2.2)	**2.0** (1.9)
	ideal	0.018 (0.015)	0.056 (0.10)	0.15 (0.24)	0.10 (0.12)

5.2 Recombination based on VI using the Archive

With the VI algorithm, the ranking of data passed to random forest was not diversified on DTLZ2. We were able to improve the method so that various ranks are included in the data for random forest. To do so, we applied the VI_Arc algorithm proposed in Section 3.4. It uses the archived populations to construct the data provided to random forest. GD and IGD results on DTLZ2 are reported in Figure 14. We can see that VI_Arc obtains better GD values than orig and VI, and gets close to ideal, whatever the number of objectives. Figure 15 shows the VI-values of each variable for the 5-objective DTLZ2 problem obtained by VI_Arc at different generations. Note that the VI-value of distance-related variables is higher than position-related variables after 500 generations. Since this algorithm was able to discriminate between variables, we were able to effectively search the variable space related to convergence and could then find solutions with good convergence properties.

We also applied VI_Arc to multimodal problems DTLZ1 and DTLZ3. In these problems, however, GD-values are similar or worse than the orig algorithm. This is because the archive allows to increase the rank diversify by keeping information from early generations. In multimodal problems, however, this becomes detrimental since the latest information is more relevant to escape from local optima.

VI and VI_Arc are effective for problems with different characteristics. This suggests that approaches based on explorative landscape analysis, capturing the features of the problem to be solved, could be useful to determine the problem type the optimizer has to face. In the future, we plan to investigate such approaches in order to adaptively select between these two strategies.

6 Conclusions

In this work, we investigated the ability of a machine learning-enhanced method to learn variables that favor Pareto improvements on many-objective optimization problems. This method uses random forest to perform a regression of the Pareto ranking over decision variables in order to estimate the importance of variables at each iteration. We compared the convergence ability of a baseline algorithm AεSεH, a version enhanced with the method that estimates variable importance, as well as an ideal version with a perfect knowledge of the variables that are important for convergence on 3, 4, 5 and 6 objective DTLZ1, DTLZ2 and DTLZ3 test problems. In addition to DTLZ2, we also use Modified-DTLZ2 to better illustrate the ability of the proposed method. We were able to show that the machine learning-enhanced algorithm that uses the instantaneous population to determine variable importance achieves a significantly better convergence on DTLZ1, DTLZ3 and Modified-DTLZ2. We revealed that the regression model was able to accurately distinguish between distance- and position-related variables throughout the generations, based on the estimated variable importance on many-objective problems, except in the problem where it is easy to converge towards the Pareto front. We also showed that a variation of the machine learning-enhanced algorithm that uses a sample of the archived populations to determine variable importance can achieve better convergence on DTLZ2. Ours results revealed that, in order to correctly distinguish between distance- and position-related variables, we should pay attention to the data set that is submitted to random forest for training. If the number of ranks in the population is too small, the importance of the variable for the rank cannot be predicted accurately. A possible approach to increase the rank diversity is to use archived populations from different generations, even if this may have a side effect for multi-modal problems.

In the future, we would like to explore adaptive algorithms based on landscape analysis to determine whether the population or the archive should be submitted to random forest. Also, we plan to apply the machine learning-enhanced algorithm to problems with many variables, and extend the guiding method for convergence in order to guide mutation in addition to recombination. The proposed method is based on Pareto ranking. We would also like to look into alternative ways to rank the population. At last, we would like to validate the ability of the proposed approach when applied to other evolutionary algorithms. The methodology discussed hereby can be easily incorporated into various EAs to assist solution search without changing the flow of the host algorithm.

References

[1] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *Trans. Evol. Comp*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[2] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou, and T. Fogarty, Eds. International Center for Numerical Methods in Engineering, 2001, pp. 95–100.

[3] S. Watanabe, T. Hiroyasu, and M. Miki, "NCGA: Neighborhood cultivation genetic algorithm for multi-objective optimization problems." in *GECCO Late Breaking Papers*. AAAI, 2002, pp. 458–465.

[4] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition." *IEEE Trans. Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[5] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints." *IEEE Trans. Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[6] H. Aguirre, A. Oyama, and K. Tanaka, "Adaptive ε-sampling and ε-hood for evolutionary many-objective optimization," in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science, vol. 7811, 2013, pp. 322–336.

[7] R. Santana, "Gray-box optimization and factorized distribution algorithms: where two worlds collide," *CoRR*, vol. abs/1707.03093, 2017.

[8] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions i. binary parameters," in *Parallel Problem Solving from Nature — PPSN IV*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 178–187.

[9] J. A. Lozano, P. Larrañaga, I. n. Inza, and E. Bengoetxea, *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms (Studies in Fuzziness and Soft Computing)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[10] P. Larrañaga, H. Karshenas, C. Bielza, and R. Santana, "A review on probabilistic graphical models in evolutionary computation," *Journal of Heuristics*, vol. 18, no. 5, pp. 795–819, Oct. 2012.

[11] H. Karshenas, R. Santana, C. Bielza, and P. Larrañaga, "Multiobjective estimation of distribution algorithm based on joint modeling of objectives and variables," *IEEE Trans. Evolutionary Computation*, vol. 18, no. 4, pp. 519–542, 2014.

[12] X. Ma, F. Liu, Y. Qi, L. Li, L. Jiao, X. Deng, X. Wang, B. Dong, Z. Hou, Y. Zhang, and J. Wu, "MOEA/D with biased weight adjustment inspired by user preference and its application on multi-objective reservoir flood control problem," *Soft Comput.*, vol. 20, no. 12, pp. 4999–5023, 2016.

[13] M. Sagawa, H. Aguirre, F. Daolio, A. Liefooghe, B. Derbel, S. Verel, and K. Tanaka, "Learning variable importance to guide recombination," in *IEEE SSCI*, 2016.

[14] S. Huband, P. Hingston, L. Barone, and R. While, "A review of multi-objective test problems and a scalable test problem toolkit," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, 2007.

[15] L. Breiman, J. Friedman, C.Stone, and R.Olshen, *Classification and regression trees*. CRC press, 1984.

[16] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[17] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[18] C. Strobl and A. Zeilei, "Why and how to use random forest variable importance measures," *useR! 2008*, 2008.

[19] L. Breiman, "Manual on setting up, using, and understanding random forests v3. 1," *Statistics Department University of California Berkeley, CA, USA*, 2002.

[20] H. Aguirre, Y. Yazawa, A. Oyama, and K. Tanaka, "Extending AεEεH from many-objective to multi-objective optimization," in *Conference on Simulated Evolution and Learning*, ser. Lecture Notes in Computer Science, vol. 8886, 2014, pp. 239–250.

[21] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, pp. 115–148, 1995.

[22] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multi-objective optimization," *Evolutionary Multiobjective Optimization*, pp. 105–145, 2005.

[23] D. A. V. Veldhuizen, "Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations." *Proceedings of the 1999 ACM symposium on Applied computing*, pp. 351–357, 1999.

[24] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, pp. 117–132, 2003.