Learning variable importance to guide recombination on many-objective optimization

Miyako Sagawa, Hernán Aguirre, Fabio Daolio, Arnaud Liefooghe, Bilel Derbel, Sébastien Verel, Kiyoshi Tanaka

Learning Variable Importance to Guide Recombination on Many-objective Optimization

Miyako Sagawa*, Hernán Aguirre*, Fabio Daolio†, Arnaud Liefooghe‡
Bilel Derbel‡, Sébastien Verel§ and Kiyoshi Tanaka*
*Shinshu University, Nagano, Japan
Email: 15st204e@shinshu-u.ac.jp
†University of Stirling, Scotland UK
‡Univ. Lille, CNRS, UMR 9189 – CRIStAL / Inria Lille-Nord Europe, France
§Univ. Littoral Côte d'Opale, LISIC, France

***Abstract*—There are numerous many-objective real-world 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 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. Results show that variable importance can enhance the performance of many-objective evolutionary algorithms.**

I. Introduction

Multi-objective optimization evolutionary algorithms (MOEAs), such as NSGA-II [1], SPEA2 [2] and NCGA [3], can derive a good approximation of the set of Pareto optimal solutions (POS) on 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 optimization evolutionary algorithms (MaOEAs) have been developed, such as MOEA/D [4], NSGA-III [5] and AϵSϵH [6]. MaOEAs have shown better convergence of the approximation of the set of POS on many-objective test problems. Thus, it is expected that MaOEAs will contribute to solve real world problems with four and more objectives. It can be made the case that in industrial applications 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.

In this work, our aim is to improve further the solution search ability of MaOEAs. 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 randomly with respect to some user-defined probability. Thus, variables are not explicitly selected based on their contribution to improve the rank of solutions.

In [7], a machine learning-enhanced method to select variables for recombination and applied it to two-objectives problems is proposed. 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. The idea of learning about the problem while optimising it, in order to improve the solution search, is not specific to multi-objective optimisation. Instead, it belongs to the more general conceptual framework of "intelligent optimisation". 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 increased effectiveness to find solutions with good convergence.

In this work, we apply the machine learning-enhanced method proposed in [7] to optimization problems with up to six objectives optimization problems. This method uses Pareto ranking to score solutions. In many-objective problems it is well known that the number of non-dominated solutions increase exponentially with the number of objectives. Thus, in these problems it is expected a low variety of rankings in the population, which could affect the estimation of variable importance. We are interested on 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 use the multi- and many-objective optimizer AϵSϵH as 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 select from them for recombination.

In the reminder, we describe variable importance in Section II, we introduce the considered algorithms in Section III, we give our experimental setup in Section IV, we report our experimental analysis in Section V, and we conclude in Section VI.

II. 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 [8] as the score. Then, we apply crossover to variables which are identified to affect Pareto improvements.

A. Extraction of Variable Importance

In this work, to extract variable importance, we use the random forest [9] [10] implementation in R [11]. Let us define the number of trees as nt and an index of a tree as $t, (t = 1, 2, \cdots, nt)$. Also, let us define the combined population of parents and offspring at a given generation as the original data P. To grow each tree of the forest and compute the variable importance in the tree we apply the following procedure.

1) Split randomly the original data P into learning (70%) and testing (30%) data. Determine the training data set for growing a tree 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. 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.

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

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 [10] [12].

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

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

B. How to Guide Recombination

In general, a recombination operator selects at random some variables 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 of 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.

III. Algorithms

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

A. Baseline Algorithm (`orig`)

We use the Adaptive $\epsilon-$Sampling and $\epsilon-$Hood (AϵSϵH) [6] [13] as a baseline EMO algorithm. AϵSϵH is a population-based multi- and many-objective elitist evolutionary algorithm. It has two important features, $\epsilon-$Hood method used to select parents for recombination and $\epsilon-$ Sampling method used for survival selection. We use SBX [14] crossover as 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 [13], and it shows good performance for many-objective optimization [6].

B. Recombination applied to Convergence-Related Variables (`ideal`)

Let us assume that the algorithm knows 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 of 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 hands, 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.

C. Recombination based on Variable Importance (VI)

We include the method that guides recombination into the baseline algorithm AϵSϵH. As described in section II-B, we obtain the estimated variable importance for Pareto improvement from the random forest statistical model and we select the variables which have high importance for recombination.

IV. Experimental Setting

We use the DTLZ2 and DTLZ3 test problems [15]. Objective functions in DTLZ3 are separable and multi-modal, whereas those of DTLZ2 are separable and uni-modal. Both have a concave geometry on the optimal Pareto front. DTLZ problems have $M-1$ position-related variables and $n-(M-1)$ distance-related variables. 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 $(n_p, n_d) = (M-1, 9)$.

The DTLZ2 problem can be described as follows.

$$\begin{aligned}
\text{minimize } f_1(x) &= (1+g(x_M))\cos(x_1\pi/2)\cdots\cos(x_{M-1}\pi/2)\\
\text{minimize } f_2(x) &= (1+g(x_M))\cos(x_1\pi/2)\cdots\sin(x_{M-1}\pi/2)\\
&\vdots \qquad\qquad \vdots\\
\text{minimize } f_M(x) &= (1+g(x_M))\sin(x_1\pi/2) \qquad (4)\\
&0 \le x_i \le 1, \text{for} i = 1, 2, \cdots, n\\
\text{where } g(x_M) &= \sum_{x_i \in x_M} (x_i - 0.5)^2
\end{aligned}$$

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 by modifying function g as follows

$$g'(x_M) = 1000 \times \sum_{x_i \in x_M} (x_i - 0.5)^2 . \qquad (5)$$

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.

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.

In the random forest procedure, we set some parameters based on the recommended value [10]. 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.

To evaluate the search ability of the algorithms we use an an archive that keeps all non-dominated solutions found through the generations. We calculate Generational Distance (GD) [16] to evaluate convergence of the population and Inverted Generational Distance (IGD) [17] to evaluate diversity of the population. For IGD, we use a reference set of 100,000 solutions in each problem.

V. Simulation Results

We applied the three algorithms described in Section III to DTLZ3, DTLZ2 and Modified-DTLZ2 using 3, 4, 5 and 6 objective functions. Figure 1 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 1, it can be seen that `ideal` achieves the best GD-values through generations in all 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 what 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 also can be seen that `orig` has slightly better or nearly equal IGD-values than `VI` in all objectives.

Figure 2 shows boxplots of VI-values of the variables by `VI` at various generations. Results are shown for 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 boxplot 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 after 500 generations, distance-related variables have larger variable importance than position-related variables. This shows that the regression model in random forest 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 recombine. Therefore, we can find solutions which have better convergence in objective space by giving high recombination opportunity to distance-related variables.

Figure 3 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 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 4 shows the GD- and IGD-values obtained by the three algorithms on DTLZ2. We can see that `VI` has worse or same GD- and IGD-values than `orig`, but `ideal` obtained significantly better GD-values. Figure 5 shows boxplots of VI-values for each variable by `VI`. Note that there is no difference

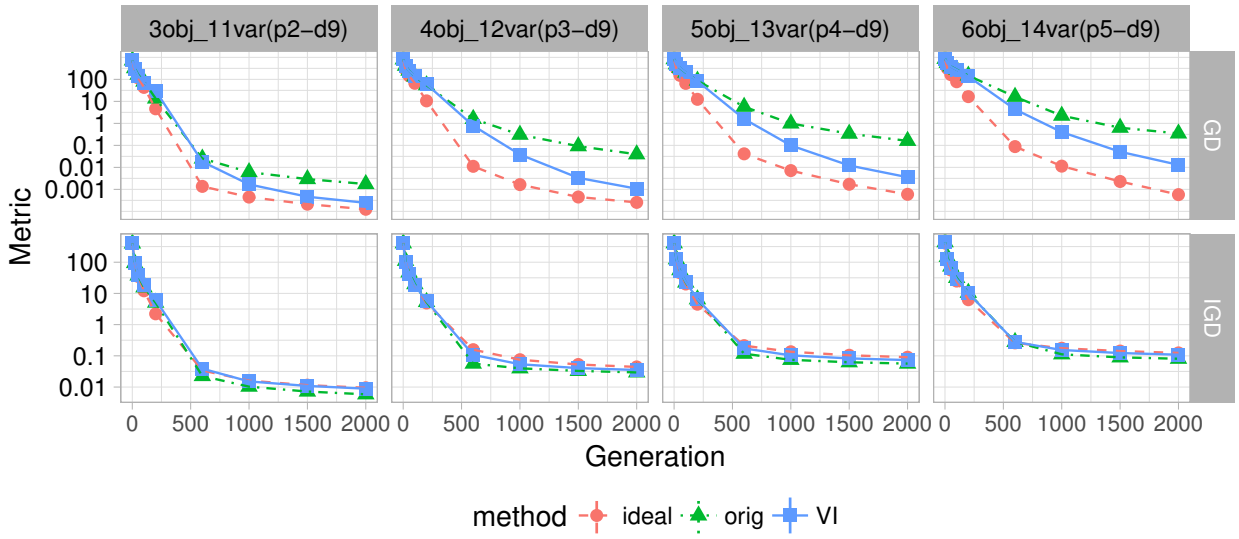

Fig. 1: GD(top) and IGD(bottom) on DTLZ3.

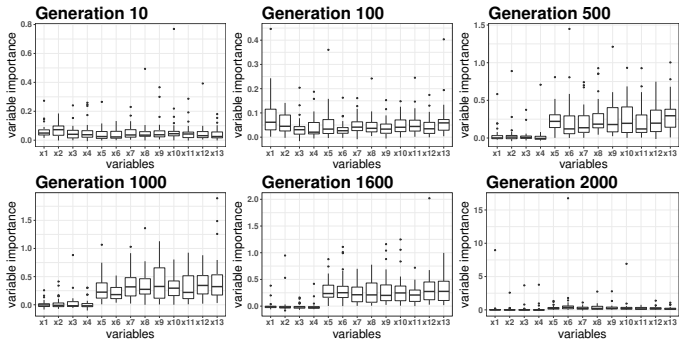

Fig. 2: VI on DTLZ3, 5 objectives.

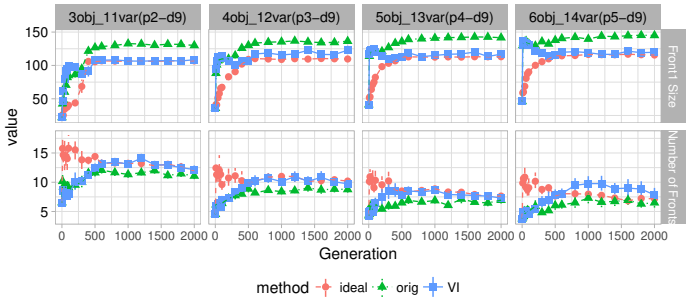

Fig. 3: Size of Front 1 (top) and number of fronts (bottom) on DTLZ3.

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 6, we can see that the size of the first front by `VI` is above 150 and greater than `orig` since early generations. Likewise, the number of fronts in the combined population is around 5 in three objectives and around 2 in 6 objectives, similar or smaller than `orig`. On the other hand, the size of front 1 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 described in Section IV. Figure 7 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 on all number of objectives and approaches `ideal` as evolution advances on 3-, 4- and 5-objectives problems. On 6-objectives, GD-values by `VI` are better than `orig` but do not approach `ideal`.

Figure 8 shows boxplots of VI-value for each variable by `VI` on 5-objectives Modified-DTLZ2 problem. Note that after 500 generations `VI` of distance-related variables $x_5 \sim x_{13}$ is higher than position-related variables $x_1 \sim x_4$. Therefore, similar to DTLZ3, in this problem distance-related variables get the chance to recombine more often and convergence improves.

Figure 9 shows the size of the first front and the number of fronts in the combined population. Looking at Figure 9, we can see that the size of front 1 by `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 properly VI and identify distance related variables for recombination.

VI. Conclusions

In this work, we investigated the ability of a machine learning-enhanced method to learn variables that favor Pareto

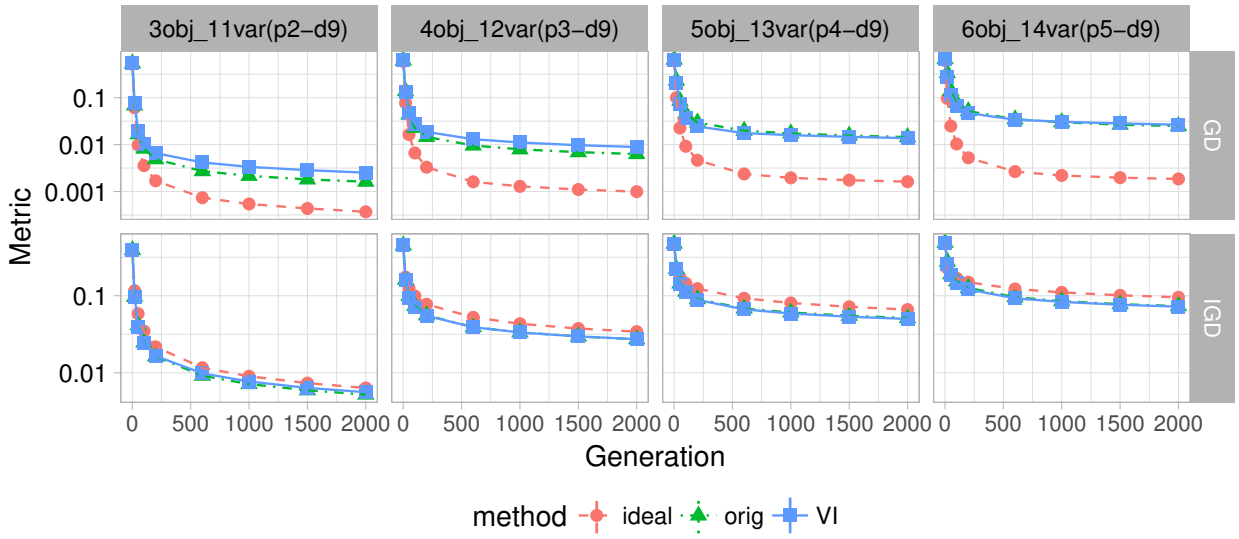

Fig. 4: GD(top) and IGD(bottom) on DTLZ2.

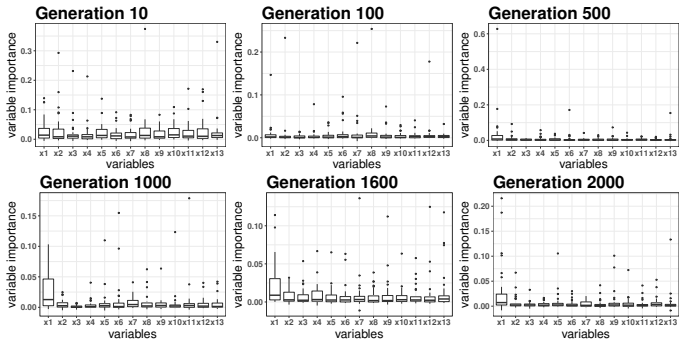

Fig. 5: VI on DTLZ2, 5 objectives.

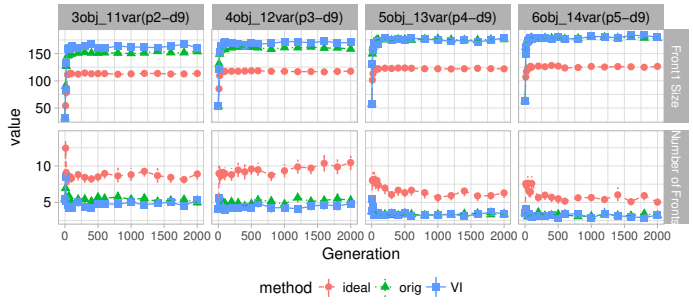

Fig. 6: Size of Front 1 (top) and number of fronts (bottom) on DTLZ2.

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 variable importance 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 DTLZ2 and DTLZ3 test problems. In addition to DTLZ2, we also use Modified-DTLZ2 to show clearly the ability of the method. We showed that the machine learning-enhanced algorithm achieves a significantly better convergence using GD and IGD metric on DTLZ3 and modified-DTLZ2. We verified that the regression model is able to distinguish correctly 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. Ours results revealed that, in order to correctly distinguish between distance- and position-related variables, we should pay attention to the population that is submitted to random forest. If the number of ranks in the population is too small, the importance of the variable for the rank cannot be predicted accurately.

In the future, 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 like also to look into alternative ways to rank the population. Moreover we need to apply the method to other algorithm because this method can be applied easily to any evolutionary 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. [Online]. Available: http://dx.doi.org/10.1109/4235.996017

[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.

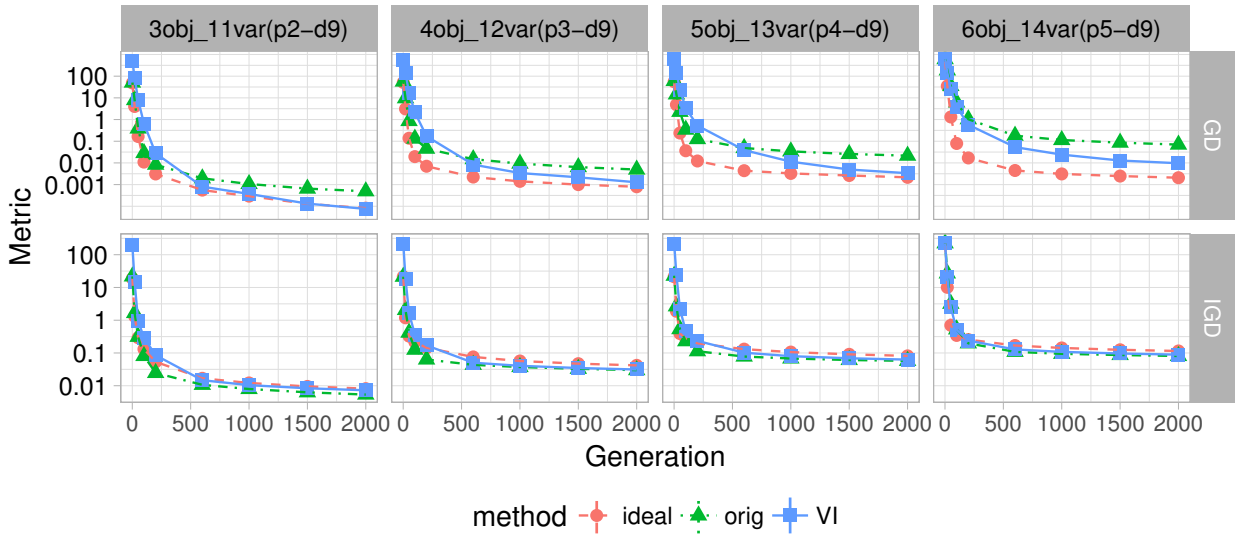

Fig. 7: GD(top) and IGD(bottom) on modified DTLZ2.

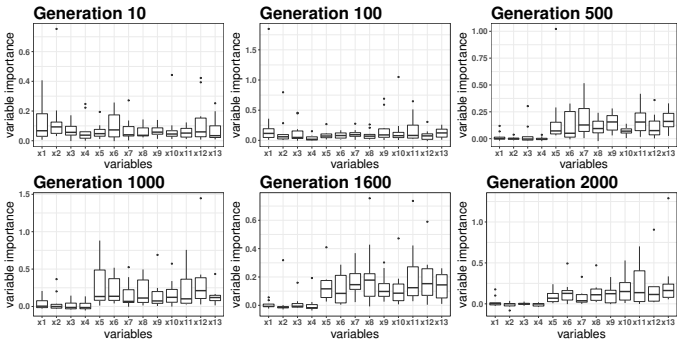

Fig. 8: VI on modified DTLZ2, 5 objectives.

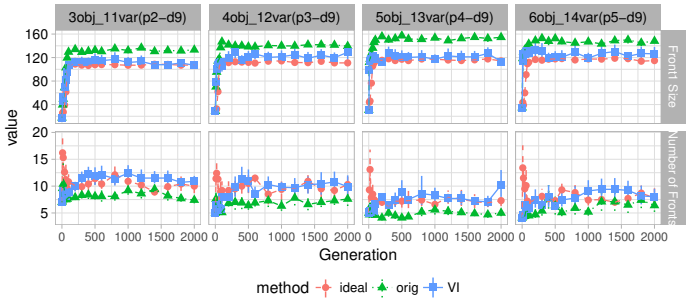

Fig. 9: Size of Front 1 (top) and number of fronts (bottom) on modified DTLZ2.

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

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

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

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

[11] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002. [Online]. Available: http://CRAN.R-project.org/doc/Rnews/

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

[13] 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.

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

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

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

[17] 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.