This paper has been published in Proceeding of International Conference on Parallel Problem Solving from Nature, 2022.

New Solution Creation Operator in MOEA/D for Faster Convergence

Longcan Chen, Lie Meng Pang, and Hisao Ishibuchi*

Guangdong Provincial Key Laboratory of Brain-Inspired Intelligent Computation
Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China
11813009@mail.sustech.edu.cn, {panglm,hisao}@sustech.edu.cn

Abstract. This paper introduces a novel solution generation strategy for MOEA/D. MOEA/D decomposes a multi/many-objective optimization problem into several single-objective sub-problems using a set of weight vectors and a scalarizing function. When a better solution is generated for one sub-problem, it is likely that a further better solution will appear in the improving direction. Examination of such a promising solution may improve the convergence speed of MOEA/D. Our idea is to use the improved directions in the current and previous populations to generate new solutions in addition to the standard genetic operators. To assess the usefulness of the proposed idea, we integrate it into MOEA/D-PBI and use a distance minimization problem to visually examine its behavior. Furthermore, the proposed idea is evaluated on some large-scale multi-objective optimization problems. It is demonstrated that the proposed idea drastically improves the convergence ability of MOEA/D.

Keywords: Evolutionary multi-objective optimization · Large-scale multi-objective optimization · MOEA/D · solution generation strategy.

1 Introduction

Many real-world applications involve multi-objective optimization problems that have conflicting objectives [1]. Without loss of generality, multi-objective optimization problems can be represented as follows:

$$\begin{aligned} &\text{Minimize } \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_m(\boldsymbol{x}))^T, \\ &\text{subject to } \boldsymbol{x} \in \Omega \end{aligned} \tag{1}$$

where $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_d)^T$ is a d-dimensional vector of decision variables, Ω is the feasible region, and $f_i(\boldsymbol{x})$ is the i-th objective to be minimized ($i = 1, 2, \ldots, m$). Since the objectives are conflicting with each other, there is no solution that can optimize all objectives simultaneously. In multi-objective optimization, the final goal is to find a set of Pareto optimal (PO) solutions. Population-based approaches are useful for discovering a set of well-distributed and well-converged solutions [1, 2], and evolutionary multi-objective optimization (EMO) is one of the effective approaches.

* Corresponding author.

MOEA/D [3] is one of the most popular decomposition-based EMO algorithms. MOEA/D uses a set of weight vectors $\boldsymbol{W} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{|P|})^T$ (where $|P|$ is the population size) and a scalarizing function to decompose a multi-objective optimization problem into a set of single-objective sub-problems. Each weight vector $\boldsymbol{w}_i = (\boldsymbol{w}_{i_1}, \boldsymbol{w}_{i_2}, \ldots, \boldsymbol{w}_{i_m})^T$ corresponds to a sub-problem. For a given sub-problem, the scalarizing function is used to calculate the fitness value of a solution. Each weight vector $\boldsymbol{w}_i$ (i-th weight vector, $i = 1, 2, ..., |P|$) has a current solution ${\boldsymbol{x}_i}^{Current}$. When an offspring solution ${\boldsymbol{x}_i}^{New}$ is better than the current solution ${\boldsymbol{x}_i}^{Current}$, ${\boldsymbol{x}_i}^{Current}$ is replaced with ${\boldsymbol{x}_i}^{New}$. Let us denote the neighborhood of $\boldsymbol{w}_i$ by S_i and its size by $|S_i|$. In each generation, the current solution ${\boldsymbol{x}_i}^{Current}$ is compared with $|S_i|$ offspring solutions (one by one) generated in the neighborhood S_i. Thus, the current solution can be updated $|S_i|$ times in each generation. The current solution is not updated if there are no better solutions than the current solution.

In MOEA/D, when a better solution is generated for one sub-problem, it is likely that a further better solution will appear in the improving direction. Examination of such a promising solution may improve the convergence speed of MOEA/D. Based on this idea, we use the solutions in the current and previous generations to generate improving directions for sub-problem. By using the improving directions, promising solutions can be generated and used for accelerating the convergence speed of MOEA/D.

The idea of using the information obtained from the current and previous generations to generate new solutions is not entirely new. In the literature, many studies focus on online innovization approaches [4–10]. Online innovization approaches attempt to learn from the current and previous generations. By extracting the patterns or relationships among variables in the decision space, online innovization approaches can accelerate the convergence speed of EMO algorithms. Mittal et al. [4] proposed a learning-based innovized progress operator for EMO algorithms. It uses a machine learning (ML) model to capture the patterns of the variables in the decision space and uses the learned ML model to improve offspring solutions. Ghosh et al. [5] proposed a method that combines user-supplied and machine-learnable patterns and rules to accelerate the convergence speed of multi-objective optimization algorithms. Mittal et al. [6] proposed an innovized repair operator which uses an ML model to repair the offspring solutions.

In this paper, we propose a solution creation method for MOEA/D by using the improving move of the current and previous solutions corresponding to each sub-problem. This paper is organized as follows. In Section 2, we explain the proposed strategy and its implementation. Next, we use computational experiments to demonstrate and validate the usefulness of the proposed strategy in Section 3. Finally, we conclude this paper and give some future research directions in Section 4.

2 Proposed Strategy and Implementations

In this section, we first explain our idea using a distance minimization problem. In this problem, as Fig. 1 shows, we need to minimize four objectives, which are f_1: Distance to P_1, f_2: Distance to P_2, f_3: Distance to P_3, and f_4: Distance to P_4. When a better solution (i.e., Offspring 4 in Fig. 1) is generated for a weight vector $\boldsymbol{w}$ with the current solution $\boldsymbol{x}$ (i.e., the red point in the figure), it is likely that we will be able to find a further better solution in the improving direction (i.e., a candidate solution as shown by the yellow circle along the red line). Examination of such a promising solution may improve the convergence speed of MOEA/D.

We assume that the current solution is replaced with the candidate solution in Fig. 1. Then as Fig. 2 shows, we also assume that a better solution (i.e., Offspring 5) is found. In this case, we can examine a candidate solution along the improving direction (e.g., Candidate Solution A on the red line). We can also generate another Candidate Solution B by considering both the current improving direction and the previous improving direction.

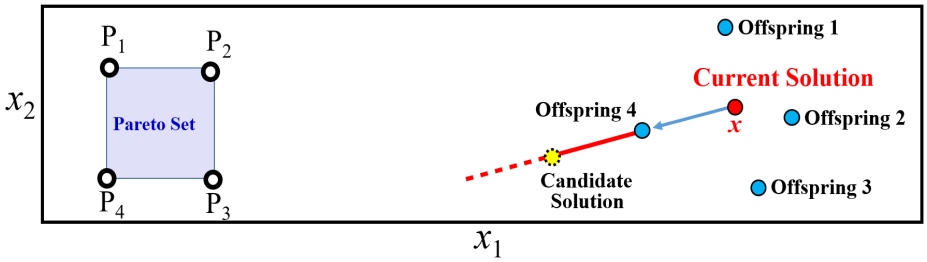

Fig. 1. Illustration of the proposed idea (use of the moves in the current generation).

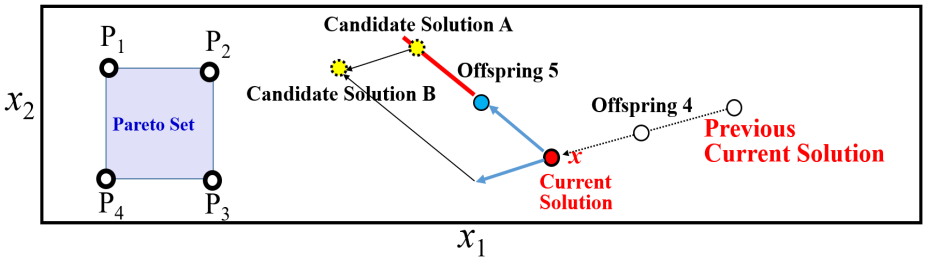

Fig. 2. Illustration of the proposed idea (use of the moves in the current and previous generations).

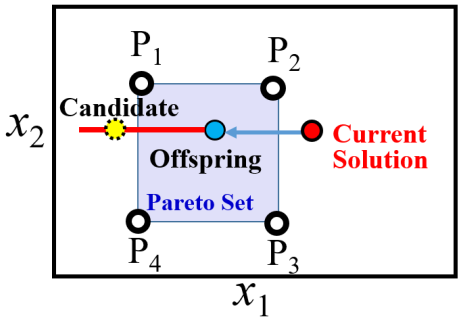

Fig. 3. Illustration of the proposed idea where the candidate is not better than the current solution.

However, the candidate solutions (e.g., the yellow circle) are not always better than the offspring solution (e.g., the blue circle) as shown in Fig. 3. In this case, the current solution (e.g., the red circle) is replaced with the offspring solution (e.g., the blue circle), not with the candidate solution (the yellow circle).

In this paper, we implement this idea for MOEA/D with the PBI function ($\theta = 5$). We propose three different implementations (i.e., Type1, Type2, and Type3) to generate candidate solutions. Type1 implementation uses the moves of the solution for the current sub-problem. Type2 implementation uses the moves of the solution for the current sub-problem and the moves of its neighboring solutions. Type3 implementation uses the moves of all solutions in the population. Additionally, each implementation can be further subdivided into two sub-types. The first sub-type considers only the current improving direction, and the second sub-type considers both the current and previous improving directions.

It should be noted that in the standard MOEA/D implementation, the initial population is randomly generated and assigned to each weight vector, which may affect the performance of the proposed strategy. An example is shown in Fig. 4. Fig. 4 illustrates the improving moves of the current solution $\boldsymbol{x}_4$ for the weight vector $\boldsymbol{w}_4$. In the figure, the pink curve is the Pareto front, ${\boldsymbol{x}_4}^{Initial}$ is a randomly generated initial solution, ${\boldsymbol{x}_4}^{(2)}$ is the current solution after the 2^{nd} generation (which is the best solution among the generated $|S_i|$ offspring solutions in the neighborhood during the 2^{nd} generation), and ${\boldsymbol{x}_4}^{(3)}$ is the current solution after the 3^{rd} generation. In many cases, the move from ${\boldsymbol{x}_4}^{Initial}$ to ${\boldsymbol{x}_4}^{(2)}$ is not a good direction (the red arrow) while the move from ${\boldsymbol{x}_4}^{(2)}$ to ${\boldsymbol{x}_4}^{(3)}$ is usually a good direction (the blue arrow).

The following are the details of the implementation of our strategy in each type.

Type1: Independent Formulation for Each Sub-problem. We denote the current solution for the weight vector $\boldsymbol{w}_i$ at the end of the t^{th} generation by $\boldsymbol{x}_i(t)$. In Type1 implementation, when $\boldsymbol{x}_i(t)$ is better than $\boldsymbol{x}_i(t\text{-}1)$, a new candidate solution is generated by the proposed strategy with a probability of 0.5. The

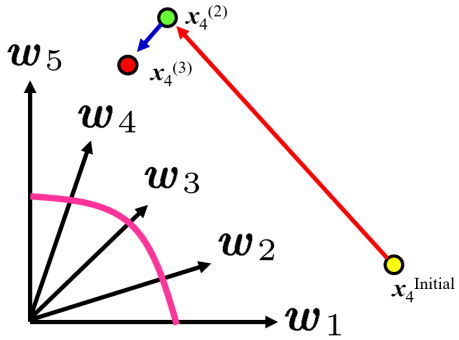

Fig. 4. Illustration of the improvement of the current solution $\boldsymbol{x}_4$ for the weight vector $\boldsymbol{w}_4$.

total move during the t^{th} generation is defined as $\Delta\boldsymbol{x}_i(t) = \boldsymbol{x}_i(t) - \boldsymbol{x}_i(t\text{-}1)$. Since the improving direction in the second generation is not reliable, $\Delta\boldsymbol{x}_i(t)$ is defined as $\Delta\boldsymbol{x}_i(t) = 0$ for $t = 2$. A new candidate solution (i.e., $\boldsymbol{x}_i^{Candidate}$) is generated from $\boldsymbol{x}_i(t)$ and $\Delta\boldsymbol{x}_i(t)$ as explained latter in detail. If $\boldsymbol{x}_i^{Candidate}$ is better than the current solution $\boldsymbol{x}_i(t)$, $\boldsymbol{x}_i(t)$ is replaced with $\boldsymbol{x}_i^{Candidate}$. $\boldsymbol{x}_i^{Candidate}$ is also compared with the neighboring solutions.

When no solution is improved during the t^{th} generation (i.e.,$\boldsymbol{x}_i(t) = \boldsymbol{x}_i(t\text{-}1)$), no candidate solution is generated. Even in this case, the current solution can be updated by a candidate solution generated for a neighboring weight vector.

Type1-1: Use of the Current Move. In Type1-1 MOEA/D, we only use the current move to generate candidates. The candidate solution is generated as $\boldsymbol{x}_i^{Candidate} = \boldsymbol{x}_i(t) + \eta\Delta\boldsymbol{x}_i(t)$ where η is a non-negative constant parameter.

Type 1-2: Use of the Current and Previous Moves. In Type1-2 MOEA/D, we consider both the current and previous improving directions. The problem is how to define the previous improving direction since the current solution was not always improved during the $(t\text{-}1)^{th}$ generation. Thus, we define the previous improving direction using the latest improved generation k before the t^{th} generation as $\Delta\boldsymbol{x}_i(k) = \boldsymbol{x}_i(k) - \boldsymbol{x}_i(k\text{-}1)$, where generation k is the latest improved generation $(1 < k < t)$ before the t^{th} generation. If there is no improved generation before the t^{th} generation, we define $\Delta\boldsymbol{x}_i(k)$ as $\Delta\boldsymbol{x}_i(k) = 0$. When $k = 2$, we define $\Delta\boldsymbol{x}_i(k)$ as $\Delta\boldsymbol{x}_i(k) = 0$ since the initial solution is randomly assigned. Then, a candidate solution can be generated as $\boldsymbol{x}_i^{Candidate} = \boldsymbol{x}_i(t) + \eta\Delta\boldsymbol{x}_i(t) + \alpha\Delta\boldsymbol{x}_i(k)$ where η and α are non-negative constant parameters.

Type1-2: One Variant of Type1-2.* In Type1-2* MOEA/D, a simplified version of the definition of the candidate solution is to use the current and previous moves as $\boldsymbol{x}_i^{Candidate} = \boldsymbol{x}_i(t) + \eta\Delta\boldsymbol{x}_i(t) + \alpha\Delta\boldsymbol{x}_i(t\text{-}1)$. In this variant, the move in the $(t\text{-}1)^{th}$ generation is used even if $\Delta\boldsymbol{x}_i(t\text{-}1) = 0$. In early generations, it is likely that the current solution is frequently improved. Thus, this variant is similar to

Type1-2 MOEA/D. However, in late generations, the current solution is not frequently improved. As a result, this variant is similar to Type1-1 MOEA/D.

Type2: Use of the Moves of Neighboring Solutions. In Type2, when at least one solution in the neighborhood S_i is improved during the t^{th} generation, the candidate solution is generated with a small probability (in our experiment, the probability is set as $5/|S_i|$). The total move during the t^{th} generation is defined as $\Delta \boldsymbol{x}_i(t) = \boldsymbol{x}_i(t) - \boldsymbol{x}_i(t\text{-}1)$, and the total move during the t^{th} generation in the neighborhood S_i is defined as $\Delta_{S_i} \boldsymbol{x}_i(t) = \sum_{j \in S_i} (\boldsymbol{x}_j(t) - \boldsymbol{x}_j(t\text{-}1))$ where $\boldsymbol{x}_i$ is included in S_i. Since the improving direction in the second generation is not reliable, $\Delta \boldsymbol{x}_i(t)$ and $\Delta_{S_i} \boldsymbol{x}_i(t)$ are defined as $\Delta \boldsymbol{x}_i(t) = 0$ and $\Delta_{S_i} \boldsymbol{x}_i(t) = 0$ for $t = 2$. Then, $\boldsymbol{x}_i^{Candidate}$ is generated as explained below. If $\boldsymbol{x}_i^{Candidate}$ is better than the current solution $\boldsymbol{x}_i(t)$, $\boldsymbol{x}_i(t)$ is replaced with $\boldsymbol{x}_i^{Candidate}$. $\boldsymbol{x}_i^{Candidate}$ is also compared with the neighboring solutions.

When no solution in the neighborhood S_i is improved during the t^{th} generation (i.e., $\boldsymbol{x}_j(t) = \boldsymbol{x}_j(t\text{-}1)$), no candidate solution is generated. Even in this case, the current solution can be updated by a candidate solution generated for a neighboring weight vector.

Type2-1: Use of the Current Move. In Type2-1 MOEA/D, we use the total move in the neighborhood S_i to generate candidates. The candidate solution is generated as $\boldsymbol{x}_i^{Candidate} = \boldsymbol{x}_i(t) + \eta \Delta \boldsymbol{x}_i(t) + \eta_{S_i} \Delta_{S_i} \boldsymbol{x}_i(t)$ where η and η_{S_i} are non-negative constant parameters. In this formulation, $\Delta \boldsymbol{x}_i(t)$ equals to 0 in many cases. However, $\Delta_{S_i} \boldsymbol{x}_i(t)$ is not zero in many cases since all the moves in the neighborhood are summed up.

Type2-2: Use of the Current and Previous Moves. In Type2-2 MOEA/D, we use the current and previous moves in the neighborhood S_i to generate candidates. We define the previous improving direction using the latest improved generation k before the t^{th} generation as $\Delta \boldsymbol{x}_i(k)$ and $\Delta_{S_i} \boldsymbol{x}_i(k)$, where k is the latest improved generation ($1 < k < t$) where at least one solution in the neighborhood S_i is improved before the t^{th} generation. If there is no improved generation before the t^{th} generation, we define $\Delta \boldsymbol{x}_i(k)$ and $\Delta_{S_i} \boldsymbol{x}_i(k)$ as 0. We also define $\Delta \boldsymbol{x}_i(k)$ and $\Delta_{S_i} \boldsymbol{x}_i(k)$ as 0 when $k = 2$ since the initial solution is randomly assigned. Then, a candidate solution can be generated as $\boldsymbol{x}_i^{Candidate} = \boldsymbol{x}_i(t) + \eta \Delta \boldsymbol{x}_i(t) + \alpha \Delta \boldsymbol{x}_i(k) + \eta_{S_i} \Delta_{S_i} \boldsymbol{x}_i(t) + \alpha_{S_i} \Delta_{S_i} \boldsymbol{x}_i(k)$, where η, η_{S_i}, α and α_{S_i} are non-negative constant parameters.

Type3: Use of the moves of All Solutions in the Population. In Type3, when at least one solution in the population is improved during the t^{th} generation, the candidate solution is generated with a small probability (in our experiment, the probability is set as $5/|S|$). The total move during the t^{th} generation is defined as $\Delta \boldsymbol{x}_i(t) = \boldsymbol{x}_i(t) - \boldsymbol{x}_i(t\text{-}1)$, and the total move during the t^{th} generation in the population S is defined as $\Delta_S x_i(t) = \sum_{j \in S} (\boldsymbol{x}_j(t) - \boldsymbol{x}_j(t\text{-}1))$. For $t = 2$, $\Delta \boldsymbol{x}_i(t)$ and $\Delta_S \boldsymbol{x}_i(t)$ are defined as $\Delta \boldsymbol{x}_i(t) = 0$ and $\Delta_S \boldsymbol{x}_i(t) = 0$.

Then, $\boldsymbol{x}_i^{Candidate}$ is generated. When no solution in the population S is improved during the t^{th} generation, no candidate solution is generated.

Type3-1: Use of the Current Move. In Type3-1 MOEA/D, we use the total move in the population S to generate candidates. The candidate solution is generated as $\boldsymbol{x}_i^{Candidate} = \boldsymbol{x}_i(t) + \eta\Delta\boldsymbol{x}_i(t) + \eta_S\Delta_S\boldsymbol{x}_i(t)$, where η and η_S are non-negative constant parameters. It should be noted that all solutions in the population have the same value of $\Delta_S\boldsymbol{x}_i(t)$.

Type3-2: Use of the Current and Previous Moves. In Type3-2 MOEA/D, we use the current and previous moves in the population S to generate candidates. We define the previous improving direction using the latest improved generation k before the t^{th} generation as $\Delta\boldsymbol{x}_i(k)$ and $\Delta_S\boldsymbol{x}_i(k)$, where k is the latest improved generation ($1<k<t$) where at least one solution in the population is improved before the t^{th} generation. The candidate solution is defined as $\boldsymbol{x}_i^{Candidate} = \boldsymbol{x}_i(t)+\eta\Delta\boldsymbol{x}_i(t)+\alpha\Delta\boldsymbol{x}_i(k)+\eta_S\Delta_S\boldsymbol{x}_i(t)+\alpha_S\Delta_S\boldsymbol{x}_i(k)$, where η, η_S, α and α_S are non-negative constant parameters.

To speed up the convergence speed, we try to find a candidate solution in the improving direction. Parameter values decide the position of a candidate solution in the improving direction. For simplicity, in this paper, we set η and α as 1 since we consider that the total move of the solution during each generation has the same weight. When using the total move in the neighborhood or population, we add all moves in the neighborhood or population together. Since the neighborhood and population size may affect the position in the improving direction, we set η_{S_i} and α_{S_i} as $1/|S_i|$, and set η_S and α_S as $1/|S|$.

3 Experimental Study

To examine the usefulness of the proposed strategy (its seven implementations), we use a multi-objective distance minimization problem (MDMP) in the 2-dimensional space [11, 12]. The effect can be visually examined by drawing the trajectory of the current solutions. The distance minimization problem is generated by using the following four points in the 2-dimensional space [1, 1001]×[1, 1001]: (2, 6), (6, 2), (2, 2), (6, 6). The four points are intentionally placed in a small region around the corner (1, 1) of the 2-dimensional space in order to examine the effect of the proposed strategy in comparison with the standard implementation of MOEA/D. In this problem, we need to minimize four objectives, which are f_1: Distance to P_1, f_2: Distance to P_2, f_3: Distance to P_3, and f_4: Distance to P_4.

Experiment settings for the 2-dimensional MDMP problem are as follows:

Software Platform. We use PlatEMO [13] as the experiment platform. PlatEMO is an open-source platform based on MATLAB for evolutionary multi-objective optimization.

Parameter Settings. Population size N is set to 56. This setting is based on the number of weight vectors generated by the Das and Dennis method [14].

The termination condition is set to 560 solution evaluations. Each algorithm is applied to each test problem for 31 independent runs.

Performance Metrics. The IGD [15] and IGD$^+$ [16] indicators are used to evaluate the performance of each algorithm.

The experiment results are shown in Table 1. The average values of IGD and IGD$^+$ over 31 runs are summarized in the table. Each algorithm is compared with the standard MOEA/D using the Wilcoxon rank sum test with the significance level of 0.05, in which the symbol "+" means that the compared algorithm is significantly better than the standard MOEA/D, the symbol "−" means that the compared algorithm is significantly worse than the standard MOEA/D, and the symbol "=" means that there is no statistically significant difference between the compared algorithm and the standard MOEA/D. The statistical test results are summarized at the bottom of each table. The best result is highlighted by blue font, and the worst result is highlighted by red font.

As Table 1 shows, almost all algorithms perform well on MDMP. Although Type2-2 MOEA/D performs the worst among all algorithms, there is no statistically significant difference between it and the standard MOEA/D.

Table 1. Average IGD$^+$ and IGD Values on MDMP ($d = 2$) obtained by MOEA/D with the proposed strategy and the standard MOEA/D.

Problem	Indicator	Type1-1 MOEA/D	Type1-2 MOEA/D	Type1-2* MOEA/D	Type2-1 MOEA/D	Type2-2 MOEA/D	Type3-1 MOEA/D	Type3-2 MOEA/D	MOEA/D
MDMP	IGD$^+$	6.9620e-1 +	7.2649e-1 =	6.4160e-1 +	8.3517e-1 =	9.3748e-1 =	7.5255e-1 =	7.9250e-1 =	8.0039e-1
	IGD	1.0345e+0 +	1.0950e+0 =	9.9320e-1 +	1.2195e+0 =	1.3138e+0 =	1.1216e+0 =	1.1941e+0 =	1.2015e+0
+/−/=		2/0/0	0/0/2	2/0/0	0/0/2	0/0/2	0/0/2	0/0/2	

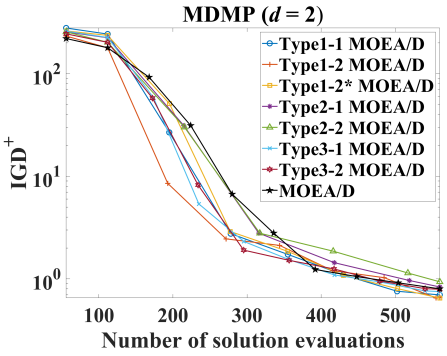

Fig. 5. Average IGD$^+$ value of the current population at each generation over 31 runs on MDMP ($d = 2$).

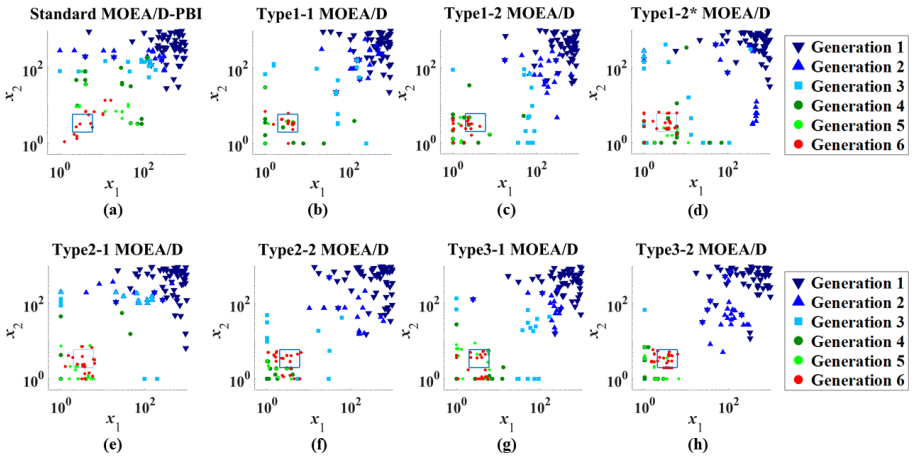

Fig. 6. The current population in the decision space of each algorithm at the first six generations on MDMP ($d = 2$).

To clearly show the convergence ability of MOEA/D with the proposed strategy and the standard MOEA/D, we use Fig.5 to show the relation between the average IGD^+ value (y-axis) and the number of examined solutions (x-axis) for each algorithm. As shown in Fig. 5, MOEA/D with the proposed strategy clearly converges faster than the standard MOEA/D before 300 solution evaluations.

To show the convergence trajectory of each algorithm, we choose a single run with the median IGD^+ value among the 31 runs and plot the population in the decision space at each of the first 6 generations in Fig. 6 (a)-(h). The blue square frame represents the Pareto set. The triangles, squares and dots represent the solutions in the decision space. In Fig. 6, more red dots in the blue square means faster convergence of the algorithm. In Fig. 6 (a), only a small number of red dots are in the blue square frame. However, in Fig. 6 (b)-(h), more red dots are in the blue square frame, which indicates that the proposed strategy can clearly speed up the convergence of MOEA/D in MDMP ($d = 2$).

To further examine the usefulness of the proposed strategy, we use four large-scale MDMPs [17, 18] to test the performance of the MOEA/D with the proposed strategy (its seven implementations). Their decision spaces are 10-, 100-, 500-, and 1000-dimensional, respectively. The decision space of each problem is [0, 100]×[0, 100]× ... ×[0, 100]. Each problem uses the following four points P_1 (1, 1, 0, ..., 0), P_2 (5, 1, 0, ..., 0), P_3 (1, 5, 0, ..., 0), P_4 (5, 5, 0, ..., 0). The four points are intentionally placed in a small region around the corner (0, 0, ..., 0) in order to examine the effect of the proposed strategy in comparison with the standard implementation of MOEA/D. Furthermore, the usefulness of the proposed strategy is also examined on the large-scale three-objective DTLZ1-4 test problems with $d = 500$ and 1000 where d is the number of decision variables.

Table 2. Average IGD^+ and IGD Values on MDMP (d = 10, 100, 500, 1000) obtained by the MOEA/D with the proposed strategy and the standard MOEA/D.

Problem	Indicator	Type1-1 MOEA/D	Type1-2 MOEA/D	Type1-2* MOEA/D	Type2-1 MOEA/D	Type2-2 MOEA/D	Type3-1 MOEA/D	Type3-2 MOEA/D	MOEA/D
MDMP	IGD^+	1.9645e-1 +	1.9320e-1 +	1.9351e-1 +	1.9973e-1 +	1.9738e-1 +	2.0323e-1 +	1.9888e-1 +	2.1986e-1
d=10	IGD	3.2504e-1 +	3.1739e-1 +	3.1753e-1 +	3.3500e-1 +	3.2549e-1 +	3.4067e-1 +	3.3254e-1 +	3.7101e-1
MDMP	IGD^+	8.0770e+1 +	6.1398e+0 +	1.6572e+1 +	6.1579e+1 +	3.4040e+0 +	1.3828e+1 +	2.8008e+0 +	2.6249e+2
d=100	IGD	8.0770e+1 +	6.2429e+0 +	1.6639e+1 +	6.1579e+1 +	3.6002e+0 +	1.3840e+1 +	2.9816e+0 +	2.6249e+2
MDMP	IGD^+	2.4575e+2 +	2.3197e+1 +	7.0410e+1 +	1.9420e+2 +	2.4354e+1 +	8.5562e+1 +	1.4220e+1 +	7.8203e+2
d=500	IGD	2.4575e+2 +	2.3247e+1 +	7.0419e+1 +	1.9420e+2 +	2.4397e+1 +	8.5562e+1 +	1.4221e+1 +	7.8203e+2
MDMP	IGD^+	3.7112e+2 +	5.9208e+1 +	1.3932e+2 +	2.9512e+2 +	4.3213e+1 +	1.3603e+2 +	2.4485e+1 +	1.1599e+3
d=1000	IGD	3.7112e+2 +	5.9219e+1 +	1.3932e+2 +	2.9512e+2 +	4.3230e+1 +	1.3603e+2 +	2.4485e+1 +	1.1599e+3
+/−/=		8/0/0	8/0/0	8/0/0	8/0/0	8/0/0	8/0/0	8/0/0	

Our experimental settings are as follow. Population size N is set to 120 on MDMP (d = 10, 100, 500, 1000) and 91 on DTLZ1-4 (d = 500, 1000). This setting is based on the number of weight vectors generated by the Das and Dennis method [14]. The termination condition is set to 6000, 12000, 60000, and 120000 solution evaluations for MDMP with d = 10, 100, 500 and 1000, respectively, and 10000 solution evaluations for DTLZ1-4 with d =500 and d = 1000. Each algorithm is applied to each test problem for 31 independent runs.

Experimental results on MDMP (d = 10, 100, 500, 1000) and DTLZ1-4 (d = 500, 1000) are summarized in Table 2 and Table 3, respectively.

In Table 2, MOEA/D with any implementation of the proposed strategy performs clearly better than the standard MOEA/D on the large-scale MDMP. Especially, Type3-2 MOEA/D performs clearly the best among all algorithms. In Table 3, MOEA/D with the proposed strategy performs clearly better than the standard MOEA/D on the large-scale DTLZ1 and DTLZ3. However, Type1 and Type2 MOEA/D are slightly worse than the standard MOEA/D on DTLZ2 and DTLZ4.

Table 3. Average IGD^+ and IGD Values on DTLZ1-4 (d = 500, 1000) obtained by the MOEA/D with proposed strategy and the standard MOEA/D.

Problem	Indicator	Type1-1 MOEA/D	Type1-2 MOEA/D	Type1-2* MOEA/D	Type2-1 MOEA/D	Type2-2 MOEA/D	Type3-1 MOEA/D	Type3-2 MOEA/D	MOEA/D
DTLZ1	IGD^+	3.9936e+3 +	4.1167e+3 +	4.0188e+3 +	4.1254e+3 +	4.1009e+3 +	6.0997e+3 +	5.3836e+3 +	8.8888e+3
d=500	IGD	3.9936e+3 +	4.1167e+3 +	4.0188e+3 +	4.1254e+3 +	4.1009e+3 +	6.0997e+3 +	5.3836e+3 +	8.8888e+3
DTLZ1	IGD^+	8.3988e+3 +	8.5284e+3 +	8.3412e+3 +	8.5624e+3 +	8.4580e+3 +	1.3750e+4 +	1.2297e+4 +	2.2017e+4
d=1000	IGD	8.3988e+3 +	8.5284e+3 +	8.3412e+3 +	8.5624e+3 +	8.4580e+3 +	1.3750e+4 +	1.2297e+4 +	2.2017e+4
DTLZ2	IGD^+	1.7238e+1 -	1.7315e+1 -	1.7948e+1 -	1.9439e+1 -	2.0212e+1 -	1.5447e+1 =	1.5937e+1 -	1.4910e+1
d=500	IGD	1.7238e+1 -	1.7316e+1 -	1.7949e+1 -	1.9440e+1 -	2.0213e+1 -	1.5448e+1 =	1.5937e+1 -	1.4911e+1
DTLZ2	IGD^+	4.8929e+1 -	4.9394e+1 -	4.8678e+1 -	5.1909e+1 -	5.2703e+1 -	4.5575e+1 -	4.5847e+1 -	4.4273e+1
d=1000	IGD	4.8929e+1 -	4.9395e+1 -	4.8678e+1 -	5.1910e+1 -	5.2703e+1 -	4.5575e+1 -	4.5848e+1 -	4.4274e+1
DTLZ3	IGD^+	1.2993e+4 +	1.3134e+4 +	1.2957e+4 +	1.3304e+4 +	1.3051e+4 +	2.0050e+4 +	1.8446e+4 +	2.9554e+4
d=500	IGD	1.2993e+4 +	1.3134e+4 +	1.2957e+4 +	1.3304e+4 +	1.3051e+4 +	2.0050e+4 +	1.8446e+4 +	2.9554e+4
DTLZ3	IGD^+	2.7139e+4 +	2.7667e+4 +	2.6864e+4 +	2.7382e+4 +	2.7132e+4 +	4.5599e+4 +	3.9454e+4 +	7.4475e+4
d=1000	IGD	2.7139e+4 +	2.7667e+4 +	2.6864e+4 +	2.7382e+4 +	2.7132e+4 +	4.5599e+4 +	3.9454e+4 +	7.4475e+4
DTLZ4	IGD^+	2.1359e+1 -	1.9885e+1 -	2.0306e+1 -	2.0505e+1 -	2.2361e+1 -	1.6608e+1 =	1.6949e+1 =	1.7311e+1
d=500	IGD	2.1366e+1 -	1.9894e+1 -	2.0314e+1 -	2.0513e+1 -	2.2367e+1 -	1.6622e+1 =	1.6960e+1 =	1.7324e+1
DTLZ4	IGD^+	5.4717e+1 -	5.6761e+1 -	5.7274e+1 -	5.8057e+1 -	5.7015e+1 -	5.2648e+1 -	5.3349e+1 -	5.0683e+1
d=1000	IGD	5.4720e+1 -	5.6764e+1 -	5.7276e+1 -	5.8060e+1 -	5.7018e+1 -	5.2651e+1 -	5.3352e+1 -	5.0687e+1
+/−/=		8/8/0	8/8/0	8/8/0	8/8/0	8/8/0	8/4/4	8/6/2	

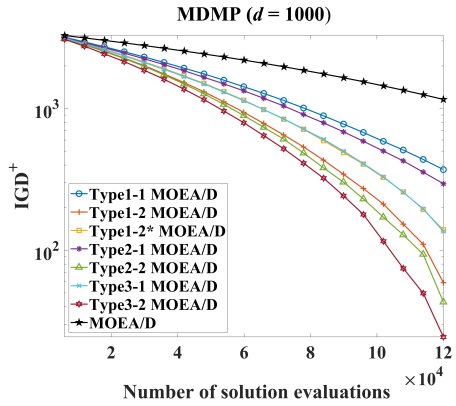

Fig. 7. Average IGD$^+$ value of the current population at each generation over 31 runs on MDMP ($d = 1000$).

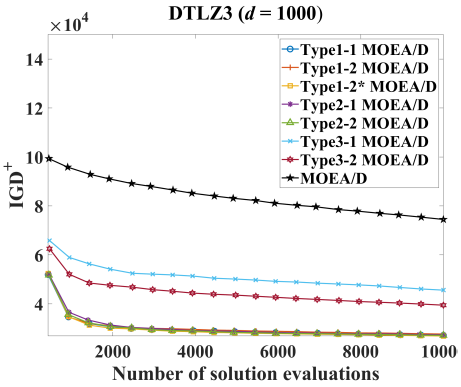

Fig. 8. Average IGD$^+$ value of the current population at each generation over 31 runs on DTLZ3 ($d = 1000$).

Figs. 7 and 8 show the relation between the average IGD$^+$ value (y-axis) and the number of examined solutions (x-axis) obtained by the standard MOEA/D and MOEA/D with the proposed strategy on MDMP (d=1000) and DTLZ3 (d=1000). As shown in Fig. 7 and Fig. 8, MOEA/D with any implementation of the proposed strategy converges much faster than the standard MOEA/D. Type3-2 MOEA/D clearly converges the fastest on MDMP ($d = 1000$). However, on DTLZ3 ($d = 1000$), Type3 MOEA/D performs not as well as the MOEA/D with the other implementations (whereas Type3 MOEA/D is much faster than the standard MOEA/D). By comparing between Type1-1 and Type1-2 (and

comparing between Type2-1 and Type2-2, and between Type3-1 and Type3-2), we can conclude that the use of the current and previous moves can help MOEA/D converge faster than the use of only the current move. By comparing Type1-2 with Type1-2*, we can conclude that using the move in the $(t\text{-}1)^{th}$ generation even if $\Delta x_i(t\text{-}1) = 0$ is not as efficient as using the move in the latest improved generation before the t^{th} generation.

4 Conclusion and Future Work

In this paper, we proposed a novel solution generation operator for MOEA/D. By using the moves of solutions in the current and previous generations, we can generate promising candidate solutions. The experimental studies showed that the proposed strategy significantly speeds up the convergence speed of MOEA/D.

One future research topic is to investigate the sensitivity of the proposed strategy to parameter settings. In the current implementation, the total move during each generation is given the same weight and each parameter value is fixed. One possibility is to investigate the use of adaptive parameter controls in the implementation. For example, the control parameters could be decreased throughout the evolutionary process. Another possibility is to use a random number as a parameter value instead of a fixed value.

References

1. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. Wiley, Chichester (2001)
2. Coello, C.A.C., Lamont, G.B., and Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-Objective Problems. Springer, New York (2007)
3. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE Transactions on Evolutionary Computation **11**(6), 712–731 (2007)
4. Mittal, S., Saxena, D.K., Deb, K., Goodman, E.D.: A Learning-based Innovized Progress Operator for Faster Convergence in Evolutionary Multi-objective Optimization. ACM Transactions on Evolutionary Learning and Optimization **2**(1), 1-29 (2021)
5. Ghosh, A., Deb, K., Averill, R., Goodman, E.D.: Combining User Knowledge and Online Innovization for Faster Solution to Multi-objective Design Optimization Problems. In: Ishibuchi, H., et al. (eds) EMO 2021. LNCS, vol. 12654, pp. 102–114. Springer, Cham (2021)
6. Mittal, S., Saxena, D.K., Deb, K., Goodman, E.D.: Enhanced Innovized Repair Operator for Evolutionary Multi- and Many-objective Optimization. arXiv preprint arXiv:2011.10760 (2020)
7. Ghosh, A., Goodman, E.D., Deb, K., Averill, R., Diaz, A.: A Large-scale Bi-objective Optimization of Solid Rocket Motors Using Innovization. In: 2020 IEEE Congress on Evolutionary Computation (CEC 2020), pp. 1–8 (2020)
8. Mittal, S., Saxena, D.K., Deb, K.: A Unified Automated Innovization Framework Using Threshold-based Clustering. In: 2020 IEEE Congress on Evolutionary Computation (CEC 2020), pp. 1–8 (2020)

9. Mittal, S., Saxena, D.K., Deb, K.: Learning-based multi-objective optimization through ANN-assisted online Innovization. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion (GECCO 2020), pp. 171–172 (2020)
10. Garg, K., Mukherjee, A., Mittal, S., Saxena, D.K., Deb, K.: A generic and computationally efficient automated innovization method for power-law design rules. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion (GECCO 2020), pp. 161–162 (2020)
11. Ishibuchi, H., Hitotsuyanagi, Y., Tsukamoto, N., Nojima, Y.: Many-Objective Test Problems to Visually Examine the Behavior of Multiobjective Evolution in a Decision Space. In: Parallel Problem Solving from Nature, PPSN XI., pp. 91–100 (2010)
12. Ishibuchi, H., Akedo, N., Nojima, Y.: A many-objective test problem for visually examining diversity maintenance behavior in a decision space. In: Proceedings of the 2011 Genetic and Evolutionary Computation Conference Companion (GECCO 2011), pp. 649–656 (2011)
13. Tian, Y., Cheng, R., Zhang, X., Jin, Y..: PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization [Educational Forum]. IEEE Computational Intelligence Magazine **12**(4), 73–87 (2017)
14. Das, I., Dennis, J.E.: Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems. SIAM Journal on Optimization **8**(3), 631–657 (1998)
15. Coello, C.A.C., Cortés, N.C.: Solving Multiobjective Optimization Problems Using an Artificial Immune System. Genetic Programming and Evolvable Machines **6**(2), 163–190 (2015)
16. Ishibuchi, H., Masuda, H., Tanigaki, Y., Nojima, Y.: Modified Distance Calculation in Generational Distance and Inverted Generational Distance. In: Gaspar-Cunha, A., et al. (eds) EMO 2015. LNCS,vol. 9019, pp. 110–125. Springer, Cham (2015)
17. Ishibuchi, H., Yamane, M., Akedo, N., Nojima, Y.: Many-objective and many-variable test problems for visual examination of multiobjective search. In: 2013 IEEE Congress on Evolutionary Computation (CEC 2013), pp. 1491–1498 (2013)
18. Masuda, H., Nojima, Y., Ishibuchi, H.: Visual examination of the behavior of EMO algorithms for many-objective optimization with many decision variables. In: 2014 IEEE Congress on Evolutionary Computation (CEC 2014), pp. 2633–2640 (2014)