Effects of Local Mating in Inter-task Crossover on the Performance of Decomposition-based Evolutionary Multiobjective Multitask Optimization Algorithms

Ryuichi Hashimoto
Department of Computer Science and Intelligent Systems
Graduate School of Engineering
Osaka Prefecture University
Osaka, Japan
ryuichi.hashimoto@ci.cs.osakafu-u.ac.jp

Toshiki Urita
Department of Computer Science and Intelligent Systems
Graduate School of Engineering
Osaka Prefecture University
Osaka, Japan
toshiki.urita@ci.cs.osakafu-u.ac.jp

Naoki Masuyama
Department of Computer Science and Intelligent Systems
Graduate School of Engineering
Osaka Prefecture University
Osaka, Japan
masuyama@cs.osakafu-u.ac.jp

Yusuke Nojima
Department of Computer Science and Intelligent Systems
Graduate School of Engineering
Osaka Prefecture University
Osaka, Japan
nojima@cs.osakafu-u.ac.jp

Hisao Ishibuchi
Department of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, China
hisao@sustech.edu.cn

***Abstract*—Recently, Evolutionary Multiobjective Multitask Optimization (EMMO) was proposed as a new research topic in the field of Evolutionary Multiobjective Optimization (EMO). In contrast to conventional EMO algorithms, EMMO algorithms solve multiple multiobjective optimization problems (multiple tasks) in their single run. Most EMMO algorithms have the same number of populations as the number of tasks to be solved simultaneously, and each population corresponds to a different task. The main feature of EMMO algorithms is that offspring solutions are generated by not only intra-task crossover but also inter-task crossover. Local mating in intra-task crossover improves the search performance of EMO algorithms that use uniformly distributed weight vectors during a search, such as MOEA/D. Therefore, local mating in inter-task crossover is a promising idea for EMMO algorithms. In this paper, we propose a simple extension of MOEA/D for EMMO algorithms and a local mating method in inter-task crossover based on uniformly distributed weight vectors. Through computational experiments, we examine the effects of local mating in inter-task crossover on the search performance of the proposed algorithm. Experimental results show that the local mating improves the search performance of the proposed algorithm.**

Keywords*— *local mating, inter-task crossover, decomposition-based evolutionary multiobjective multitask optimization algorithm

I. Introduction

Multiobjective Optimization Problems (MOPs) have more than one objective function to be optimized simultaneously [1]-[6]. When the objective functions conflict with each other, there exists a set of optimal solutions instead of a single optimal solution in an MOP. These optimal solutions are called Pareto-optimal solutions, and a set of Pareto-optimal solutions in the objective space is called the Pareto front. Evolutionary Multiobjective Optimization (EMO) algorithms are frequently-used approaches to solve MOPs [3]-[6]. This is because EMO algorithms can obtain a large number of solutions, which approximate the Pareto front by utilizing a population-based evolutionary search mechanism by their single run.

Evolutionary Multiobjective Multitask Optimization (EMMO) is a recently emerged research topic in the field of EMO [7]-[11]. In contrast to conventional EMO algorithms, EMMO algorithms solve multiple MOPs in a single run of evolutionary computation. In this research topic, each MOP is called a task. Most EMMO algorithms have the same number of populations as the number of tasks, and each population corresponds to a different task [12]. The main difference between EMMO algorithms and EMO algorithms is that solution information of each task is shared with other tasks.

One approach to share solution information among tasks is to apply crossover to solutions with different tasks. In this paper, this operation is denoted as "inter-task crossover." Offspring solutions generated by inter-task crossover have information on parent solutions for both tasks. By inserting the offspring solutions into the population for each task, solution information of each task is shared with the other tasks. In [8]-[11], the inter-task crossover improves the search performance of the EMMO algorithms. Thus, the inter-task crossover plays an important role in EMMO algorithms.

The Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D) [4] is one of the most frequently-used EMO algorithms, and good search performance by

This work was supported by National Natural Science Foundation of China (Grant No. 61876075), Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531), the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008)
Corresponding author: Hisao Ishibuchi (hisao@sustech.edu.cn).

MOEA/D is reported in [1], [2], [4]. One of the main features of MOEA/D is local mating, which applies crossover to neighboring solutions in the objective space. In [1], [2], [4], local mating has positive effects on the search performance of MOEA/D. In other words, local mating in crossover within a task has positive effects on the search performance of MOEA/D-based EMMO algorithms. In this paper, the crossover within a task is denoted as "intra-task crossover." Therefore, local mating in inter-task crossover may be a promising idea to improve the search performance of EMMO algorithms.

Although local mating in inter-task crossover is considered a promising idea, any EMMO algorithm which applies local mating in inter-task crossover has not been proposed yet as far as we know. One possible reason is its difficulty in the definition of a neighborhood relation across tasks. To perform local mating, MOEA/D utilizes neighborhood relations among uniformly distributed weight vectors in the objective space. However, since many EMMO algorithms solve each task by a dominance-based EMO algorithm (e.g., NSGA-II), it is difficult to define the neighborhood relation in the objective space.

In this paper, we propose a simple framework of an MOEA/D-based EMMO algorithm, which is named Multitask-MOEA/D (MT-MOEA/D). In MT-MOEA/D, the number of populations is equal to the number of tasks. Each population corresponds to a different task, and it is optimized for its own task by MOEA/D. The main difference between MT-MOEA/D and MOEA/D is that parent solutions are selected from the population not only for its own task but also for the other tasks with a pre-specified probability. We also propose a local mating method in inter-task crossover and apply it to MT-MOEA/D. We examine the effects of local mating in inter-task crossover by comparing MT-MOEA/D with/without local mating through computational experiments. Moreover, since we consider that parent solution selection methods in inter-task crossover affect the search performance of MT-MOEA/D with local mating, we examine the effects of parent solution selection methods on the search performance of MT-MOEA/D with local mating.

This paper is organized as follows. In Section II, we explain MOEA/D, which is the base algorithm of the proposed MT-MOEA/D. Next, we describe MT-MOEA/D and the local mating method in inter-task crossover in Section III. The effects of local mating in inter-task crossover are examined through computational experiments in Section IV. In Section V, we examine the effects of parent solution selection methods on the search performance of MT-MOEA/D with local mating in inter-task crossover. Finally, we summarize this paper in Section VI.

II. MOEA/D

MOEA/D decomposes an MOP into a number of single-objective optimization sub-problems by using uniformly distributed weight vectors in the objective space. In this paper, we use the weight vectors generated in a triangular simple lattice design method based on the following equations:

$$w_{k1} + w_{k2} + \ldots + w_{km} = 1 \tag{1}$$

$$w_{ki} \in \left\{0, \frac{1}{H}, \frac{2}{H}, \ldots, \frac{H}{H}\right\}, (i = 1, 2, \ldots, m), \tag{2}$$

where m is the number of objectives, w_{ki} is the ith weight value of the kth weight vector $\boldsymbol{w}_k$, and H is a user-specified positive integer. In MOEA/D, each weight vector has one solution, and one offspring solution is generated by performing local mating for each weight vector.

Local mating is a method for applying crossover to the parent solutions selected from the neighborhood of a weight vector. The neighborhood of the weight vector $\boldsymbol{w}_k$ is composed of T weight vectors close to $\boldsymbol{w}_k$ including $\boldsymbol{w}_k$. Fig. 1 illustrates the parent solution candidates when local mating is applied to the weight vector $\boldsymbol{w}_3$. In Fig. 1, the number of objectives m is two, the neighborhood size T is three, and H in (2) is four. In this setting, the neighborhood of $\boldsymbol{w}_3$ is $\{\boldsymbol{w}_2, \boldsymbol{w}_3, \boldsymbol{w}_4\}$. When local mating is applied to the weight vector $\boldsymbol{w}_3$, parent solutions are randomly selected from $\{\boldsymbol{w}_2, \boldsymbol{w}_3, \boldsymbol{w}_4\}$. Then, an offspring solution is generated by applying crossover to the selected parent solutions.

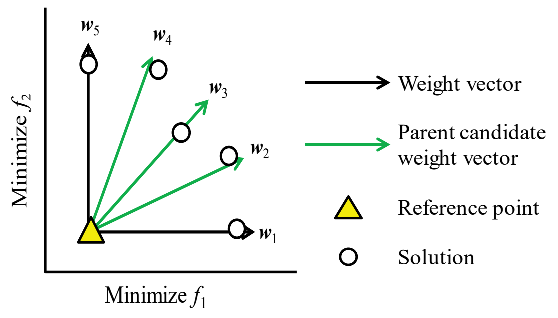

Fig. 1. Parent candidate weight vectors in MOEA/D. The number of objectives is two, the neighborhood size T is three, and H in (2) is four.

After an offspring solution of the weight vector $\boldsymbol{w}_3$ is generated, the offspring solution is compared with the current solution at each weight vector in the neighborhood of the weight vectors $\boldsymbol{w}_3$ (i.e., $\boldsymbol{w}_2$, $\boldsymbol{w}_3$, and $\boldsymbol{w}_4$). The sub-problem at each weight vector is used for this comparison. If the current solution is inferior to the offspring solution, it is replaced with the offspring solution.

III. Proposed Algorithm

A. Multitask MOEA/D

MOEA/D utilizes uniformly distributed weight vectors to obtain well-distributed solutions of an MOP. Since MOEA/D and its variants [13], [14] have shown excellent search performance in the field of EMO [1], [4], MOEA/D for EMMO is a promising approach to simultaneously solve multiple tasks. We propose a simple extension of MOEA/D for EMMO, which is named Multitask-MOEA/D (MT-MOEA/D). The proposed MT-MOEA/D is very similar to MOEA/D. The main difference between MOEA/D and MT-MOEA/D is a parent selection process. When an offspring solution of a task is generated, MOEA/D selects parent solutions only from the task while MT-MOEA/D selects parent solutions from the population not only its own task but also for the other tasks with a user-specified probability r. In this subsection, we explain the detail of MT-MOEA/D without local mating in inter-task crossover. For simplicity of explanations, let us assume that we have two tasks (task A and task B), and each task has two objective functions to be minimized.

1) Initialization of MT-MOEA/D: First, a set of uniformly distributed weight vectors is generated in the objective space of each task. We depict the objective space of each task in MT-MOEA/D in Fig. 2. As we mentioned in Section II, we use a triangular simple lattice design method to generate the weight vectors. Next, the neighborhood of each weight vector is defined as in MOEA/D. The neighborhood of the *k*th weight vector of the task *t* (i.e., $\boldsymbol{w}_{tk}$) is composed of *T* closest weight vectors to the weight vector $\boldsymbol{w}_{tk}$. Then, an initial solution for each weight vector is randomly generated and assigned to the corresponding weight vector. Finally, a reference point for each task is initialized. The initial reference point for each task is the same as the ideal point of the current population for the corresponding task.

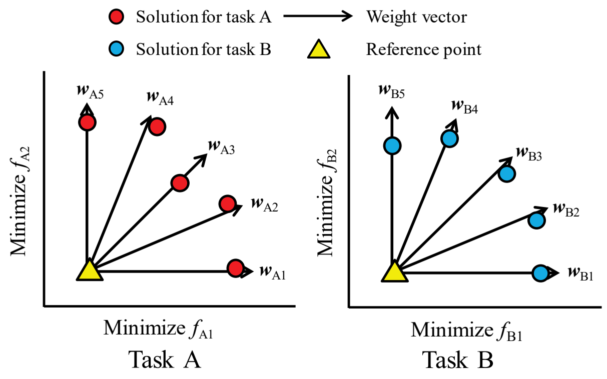

Fig. 2. A set of uniformly distributed weight vectors in the objective space of each task in MT-MOEA/D. The number of tasks is two, and each task is a two-objective optimization problem. *H* in (2) of each task is four.

2) Recombination in MT-MOEA/D: In each generation, the following procedure for generating an offspring solution is performed: (i) At first, one weight vector $\boldsymbol{w}_{tk}$ is randomly selected from both tasks. (ii) Then, with a user-specified probability *r*, two parent solutions are selected from the neighboring solutions of $\boldsymbol{w}_{tk}$, and one parent solution is selected from populations for the other tasks. Fig. 3 illustrates the parent solution candidates when $\boldsymbol{w}_{B3}$ is selected. In Fig. 3, only the green-colored weight vectors can be selected as the parent solutions. Otherwise (with a probability 1 − *r*), three parent solutions are randomly selected from the neighborhood of the weight vector $\boldsymbol{w}_{tk}$. (iii) After selecting three parent solutions, one offspring solution is generated by applying the Differential Evolution (DE) operator [15] and Polynomial Mutation [16] to the parent solutions. The generated offspring solution is evaluated in the task *t*. (iv) An update mechanism for a reference point is applied in the task *t*. The update mechanism is the same as the original MOEA/D. (v) Lastly, the offspring solution is compared with the current solution at each weight vector in the neighborhood of $\boldsymbol{w}_{tk}$ (i.e., as in Fig. 3, the offspring solution is compared with three green-colored weight vectors in the task B). The sub-problem of each weight vector is used for this comparison. If the current solution is inferior to the offspring solution in terms of the corresponding sub-problem, it is replaced with the offspring solution. In step (i), each weight vector is selected only once during a generation. This procedure for generating an offspring solution is repeated until all the weight vectors are selected.

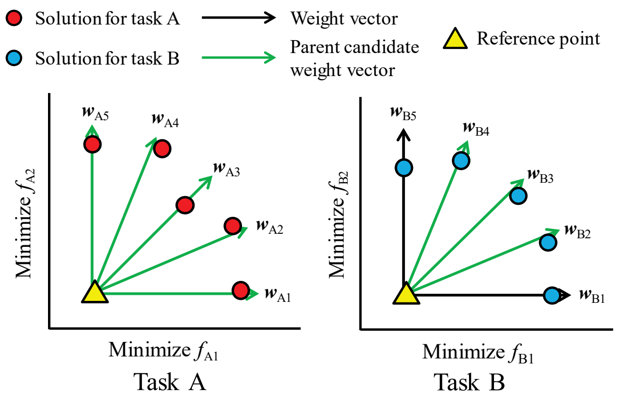

Fig. 3. Parent candidate weight vectors in MT-MOEA/D without local mating in inter-task crossover when the weight vector $\boldsymbol{w}_{B3}$ is selected. The number of objectives per task is two, *H* in (2) is four, and the neighborhood size *T* is three.

3) DE operator for MT-MOEA/D: The DE operator generates an offspring solution $\boldsymbol{y}$ by adding the weighted difference between two parent solutions ($\boldsymbol{x}_1$ and $\boldsymbol{x}_2$) to the other parent solution ($\boldsymbol{x}_3$) by using the following equation:

$$y_j = \begin{cases} x_{3j} + F \times (x_{2j} - x_{1j}), & \text{if } (rand \le CR) \text{ or } (j = j_{rand}), \\ x_{3j}, & \text{otherwise,} \end{cases} \tag{3}$$

where *CR* is a crossover probability in each decision variable, and *F* is a scaling factor. j_{rand} is a randomly selected decision variable index, and *rand* denotes a uniform random number between 0 and 1. Note that generated offspring solutions can be significantly different even when the same parents are used. This is because when the DE operator is applied for inter-task crossover, a selection order of parent solutions affects the generated offspring solution. Fig. 4 shows two examples of offspring solution generation of the task B by using the DE operator in inter-task crossover. The same parents are used in Fig. 4. In Fig 4 (a), an offspring solution is generated near the population for the task A. In contrast to Fig. 4 (a), the offspring solution is generated near the population for the task B in Fig. 4 (b). In order to generate an offspring near a population, two parent solutions $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are selected from the task B. Moreover, one parent solution $\boldsymbol{x}_3$ is selected from the task A. Therefore, by applying the DE operator for MT-MOEA/D in inter-task crossover to the parent solutions, an offspring solution for the task B is generated near the population for the task A. As a result, the generated offspring solution for the task B has information on the task A. In other words, solution information on the task A is transferred to the task B by applying the DE operator in inter-task crossover.

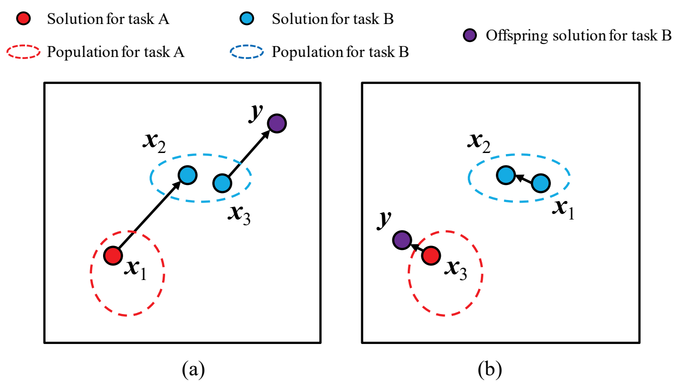

Fig. 4. An offspring solution of the task B generated by the DE operator in inter-task crossover in the decision space.

4) Additional parameter in MT-MOEA/D: In MT-MOEA/D, inter-task crossover is performed with a probability r. The probability r controls the amount of solution information sharing among tasks. If the probability r is close to 1, a large amount of solution information is shared. In contrast, if the probability r is close to 0, a less amount of solution information is shared. In the case of $r = 0$, MT-MOEA/D is the same as the independent execution of MOEA/D on each task since all offspring solutions are generated only by intra-task crossover.

B. Local mating in inter-task crossover for MT-MOEA/D

Local mating in MOEA/D is to apply crossover to neighboring parent solutions in the objective space. Since the parent solutions have similar objective function values, an offspring solution with good objective function values is likely to be generated near the parent solutions. This procedure improves the search performance of MOEA/D. Therefore, we consider that the search performance of MT-MOEA/D can be improved by applying local mating in inter-task crossover to parent solutions having similar objective function values.

If tasks have similar objective functions, solutions that are assigned to the same weight vector may have similar objective function values. For example, in Fig. 3, solutions assigned to $\boldsymbol{w}_{A1}$ and $\boldsymbol{w}_{B1}$ are better solutions for f_2 values. In such a case, we consider that the solution assigned to $\boldsymbol{w}_{B1}$ is suitable for a parent solution when an offspring solution of $\boldsymbol{w}_{A1}$ is generated by inter-task crossover. This is because $\boldsymbol{w}_{B1}$ prefers solutions that have a better f_{B2} value, and the solution assigned to $\boldsymbol{w}_{A1}$ usually has better f_{B2} value than the solution assigned to $\boldsymbol{w}_{A5}$.

From the above consideration, we propose a local mating method in inter-task crossover. Hereafter, the local mating method for MT-MOEA/D is presented in detail. Fig. 5 illustrates the parent solution candidates when local mating in inter-task crossover is applied to the weight vector $\boldsymbol{w}_{B3}$. When an offspring solution of the weight vector $\boldsymbol{w}_{B3}$ is generated by inter-task crossover, two parent solutions (i.e., $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ in (3)) are randomly selected from the neighborhood of the weight vector $\boldsymbol{w}_{B3}$. Then, one parent solution (i.e., $\boldsymbol{x}_3$ in (3)) is randomly selected from the neighborhood of a weight vector that has the same weight vector as $\boldsymbol{w}_{B3}$ excluding $\boldsymbol{w}_{B3}$ (i.e., the direction between the selected weight vector and $\boldsymbol{w}_{B3}$ is same). After selecting three parent solutions, an offspring solution is generated by applying the DE operator and Polynomial Mutation to the parent solutions.

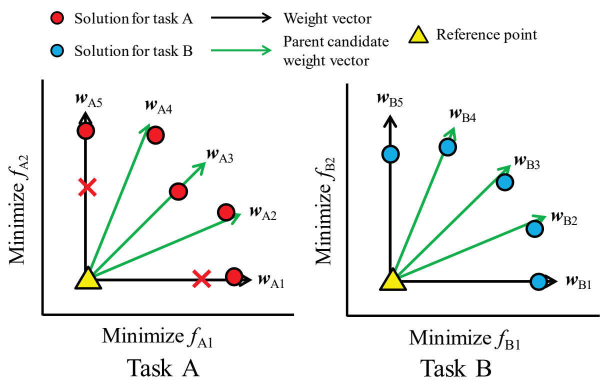

Fig. 5. Parent candidate weight vectors in MT-MOEA/D with local mating in inter-task crossover when the weight vector $\boldsymbol{w}_{B3}$ is selected. The number of objectives per task is two, H in (2) is four, and the neighborhood size is three.

In MT-MOEA/D with local mating, when an offspring solution of a weight vector $\boldsymbol{w}$ is generated, one parent solution is selected from the neighborhood of the weight vector whose direction is the same as the weight vector $\boldsymbol{w}$, not from populations for the other tasks. Thus, when an offspring solution of the weight vector $\boldsymbol{w}_{A1}$ is generated, solutions assigned to the weight vector whose direction is different from $\boldsymbol{w}_{A1}$ (e.g., $\boldsymbol{w}_{B4}$ and $\boldsymbol{w}_{B5}$) are not selected. Therefore, it is expected that similar effects to local mating in intra-task crossover on the search performance of EMMO algorithms are observed.

IV. Effects of Local Mating on the Search Performance of MT-MOEA/D

A. Experimental settings

In this section, in order to examine the effects of local mating in inter-task crossover, we apply MT-MOEA/D with/without local mating to benchmark problems. The benchmark problems used in this paper are based on the CEC 2018 Competition on Evolutionary Multi-task Multiobjective Optimization [17]. There are nine benchmark problems, and each benchmark problem has two tasks. Each task is a two-objective optimization problem defined as follows:

$$\text{Minimize } f_1(y, \boldsymbol{x}) = q(\boldsymbol{x}) \cos\left(\frac{\pi y}{2}\right), \tag{4}$$

$$f_2(y, \boldsymbol{x}) = q(\boldsymbol{x}) \sin\left(\frac{\pi y}{2}\right), \tag{5}$$

where $\boldsymbol{x}$ and y are a distance variable vector and a position variable, respectively. Each element in $\boldsymbol{x}$ takes a real number, and y takes a real number between 0 and 1. $q(\boldsymbol{x})$ is a distance function, which calculates the distance between the Pareto front and a solution. When a solution has the minimum distance function value, the solution is on the Pareto front. We use the same distance functions as those used in the CEC 2018 Competition [17]. Characteristics of the benchmark problems are summarized in Table I. These benchmark problems can be classified into three categories based on the intersection degree of the global optima of distance functions between two tasks: Complete Intersection (CI), Partial Intersection (PI), and No Intersection (NI). CI, PI, and NI mean that the global optima of the distance functions are the same, partially the same, completely different, respectively. Moreover, the benchmark problems can also be classified into three categories in terms of the similarity of the distance functions: High Similarity (HS), Medium Similarity (MS), and Low Similarity (LS). The similarity of two distance functions means the Spearman's rank correlation calculated by using randomly generated 1,000,000 solutions. The CI+HS problem in Table I is easy to be optimized in a multitasking manner. In contrast, the NI+LS problem is difficult to be optimized in a multitasking manner.

Table II shows the experimental settings used in this paper. We use the Inverted Generational Distance (IGD) [18] as a performance measure of each task since IGD is one of the most frequently-used performance measures in the EMO and EMMO communities. IGD is the average distance from each reference point to the closest solution in the obtained population. By using IGD, we can measure both the convergence and the spread of the obtained population in the objective space of each task. In these experiments, we use the same reference point set as one used in [17] to calculate IGD values.

TABLE I. CHARACTERISTICS OF BENCHMARK PROBLEMS.

Problem	Intersection degree of global optima between distance functions of two tasks	The similarity in the fitness landscape of distance functions of two tasks
CI+HS	Complete Intersection (CI)	High Similarity (HS)
CI+MS	Complete Intersection (CI)	Medium Similarity (MS)
CI+LS	Complete Intersection (CI)	Low Similarity (LS)
PI+HS	Partial Intersection (PI)	High Similarity (HS)
PI+MS	Partial Intersection (PI)	Medium Similarity (MS)
PI+LS	Partial Intersection (PI)	Low Similarity (LS)
NI+HS	No Intersection (NI)	High Similarity (HS)
NI+MS	No Intersection (NI)	Medium Similarity (MS)
NI+LS	No Intersection (NI)	Low Similarity (LS)

TABLE II. EXPERIMENTAL SETTINGS.

F in DE	Uniform random number in [0.2, 1.0]
CR in *DE*	Uniform random number in [0.2, 1.0]
Mutation	Polynomial Mutation (PM)
Mutation probability in PM	1 / (the number of decision variables)
The number of runs	101
Stopping condition	200,000 solution evaluations in total
Scalarizing function	Tchebycheff, PBI ($\theta = 5.0$)
Neighborhood size	10 for each task
H in (2)	99 for each task
r in MT-MOEA/D	0.1

B. Experimental results

Tables III and IV show the average IGD values of populations for each task by the two algorithms when we use the Tchebycheff and PBI (θ = 5.0) functions as a scalarizing function, respectively. We conduct Welch's *t*-test at the 0.05 significance level on the results. In Tables III and IV, a symbol * denotes that there exists a statistically significant difference between the results by the two algorithms. Moreover, we highlight the superior results to the other algorithm in bold. We also conduct the paired Wilcoxon test to compare the search performance of MT-MOEA/D with/without local mating through all the tasks in the nine benchmark problems. Here, the average IGD value of each task represents one observation. That is, there are 18 observations for each algorithm. We show the *p*-values calculated by using the Wilcoxon test in Table III and IV.

From Tables III and IV, we can observe that MT-MOEA/D with local mating obtained better results than MT-MOEA/D without local mating on many tasks regardless of scalarizing functions. Moreover, MT-MOEA/D without local mating did not outperform MT-MOEA/D with local mating on any tasks. In addition, the *p*-values shown in Tables III and IV are less than 0.05. These results indicate that MT-MOEA/D with local mating obtained better results than MT-MOEA/D without local mating. The only difference between the two algorithms is to apply local mating in inter-task crossover to MT-MOEA/D. Thus, this observation suggests that local mating in inter-task crossover improves the search performance of MT-MOEA/D.

TABLE III. AVERAGE IGD VALUES OF POPULATIONS FOR EACH TASK OBTAINED BY EACH ALGORITHM (TCHEBYCHEFF).

Problem	MT-MOEA/D with local mating		MT-MOEA/D without local mating	
	Task A	Task B	Task A	Task B
CI+HS	**0.000185**	**0.000887**	0.000218*	0.001230*
CI+MS	**0.006446**	**0.000960**	0.010156	0.002878*
CI+LS	**0.000245**	**0.000183**	0.000305*	0.000192*
PI+HS	**0.001004**	**0.301045**	0.001015	0.350438*
PI+MS	0.005847	**12.24740**	**0.005735**	12.79154
PI+LS	0.000286	**0.116000**	**0.000265**	0.133808
NI+HS	**1.497640**	**0.000396**	1.508565*	0.000536*
NI+MS	**0.607698**	0.007304	0.631342	**0.007271**
NI+LS	0.000365	0.000296	**0.000302**	**0.000270**
p-value	0.01387			

A symbol * denotes that the difference between the two algorithms is statistically significant. Better results than the other algorithm are highlighted in bold.

TABLE IV. AVERAGE IGD VALUES OF POPULATIONS FOR EACH TASK OBTAINED BY EACH ALGORITHM (PBI).

Problem	MT-MOEA/D with local mating		MT-MOEA/D without local mating	
	Task A	Task B	Task A	Task B
CI+HS	**0.000522**	**0.003110**	0.000883*	0.004071*
CI+MS	**0.000260**	**0.000520**	0.001196	0.000606
CI+LS	**0.001147**	**0.000258**	0.001415*	0.000313*
PI+HS	**0.002586**	**0.377532**	0.003081*	0.533442*
PI+MS	**0.008301**	**8.198671**	0.009413*	8.316388
PI+LS	0.001118	**0.558582**	**0.001114**	0.607121*
NI+HS	**1.550457**	**0.001159**	1.582770*	0.001718*
NI+MS	**0.626971**	**0.007810**	0.780259	0.008142
NI+LS	**0.000249**	**0.001031**	0.000277	0.001073
p-value	0.00023			

A symbol * denotes that the difference between the two algorithms is statistically significant. Better results than the other algorithm are highlighted in bold.

As described in Section III, it is expected that offspring solutions are generated near a weight vector by applying local mating in inter-task crossover. As a result, it is considered that MT-MOEA/D with local mating outperformed MT-MOEA/D without local mating on many tasks. In order to confirm that the proposed local mating method works as expected, we plot the distribution of offspring solutions generated by inter-task crossover with/without local mating in Fig. 6. In Fig. 6, offspring solutions are generated by adding a weighted difference between the two blue-colored solutions to the red-colored solution. If local mating is applied, a solution near the weight vector is selected as a parent solution. In this case, offspring solutions are generated near the weight vector as shown in Fig. 6 (a). In contrast, when local mating is not applied, parent solutions are not always positioned near the weight vector as shown in Fig. 6 (b). Thus, offspring solutions near the neighborhood of the weight vector are not always generated as shown in Fig. 6 (b). Therefore, the search performance of MT-MOEA/D with local mating outperformed MT-MOEA/D without local mating on many benchmark problems.

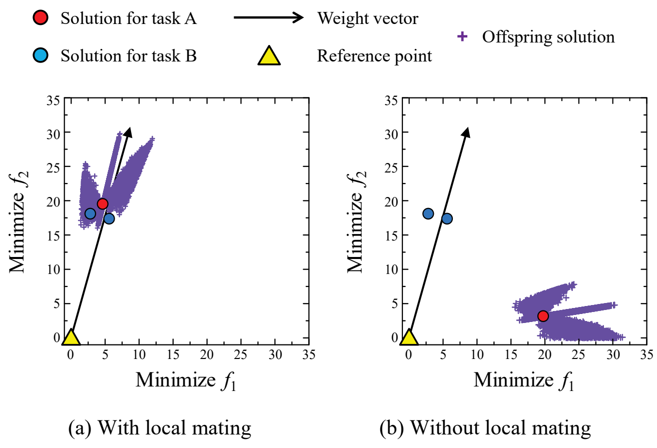

Fig. 6. Offspring solutions generated by the DE operator in inter-task crossover.

Table III shows that when we apply the Tchebycheff function to the two algorithms, MT-MOEA/D with local mating obtained better results on 8 out of the 18 tasks. On the other hand, Table IV shows that when we applied the PBI function (θ = 5.0), MT-MOEA/D with local mating obtained better results on 10 out of the 18 tasks. Moreover, the p-values obtained when we use the PBI function (θ= 5.0) is smaller than one obtained when we used the Tchebycheff function. From these results, we can observe that the search performance of MT-MOEA/D with local search was easier to be improved with the PBI function than the Tchebycheff function. This may be because the area where a solution replacement mechanism, as described in Section II, is performed of the PBI function (θ= 5.0) is smaller than the replacement area of the Tchebycheff function. In this paper, this area is denoted as a "replacement area." Fig. 7 shows an example of the replacement area of each scalarizing function. From Fig. 7, we can observe that the replacement area of the PBI function (θ = 5.0) is smaller than that of the Tchebycheff function. In other words, when we use the PBI function (θ= 5.0), in order to update the current population, it is needed to generate offspring solutions near the corresponding weight vector, as compared to the Tchebycheff function. By applying local mating in inter-task crossover, offspring solutions are more likely to be generated near the weight vector. Thus, we consider that the difference between the search performance of MT-MOEA/D with/without local mating was large when we used the PBI function (θ= 5.0), as compared to the Tchebycheff function.

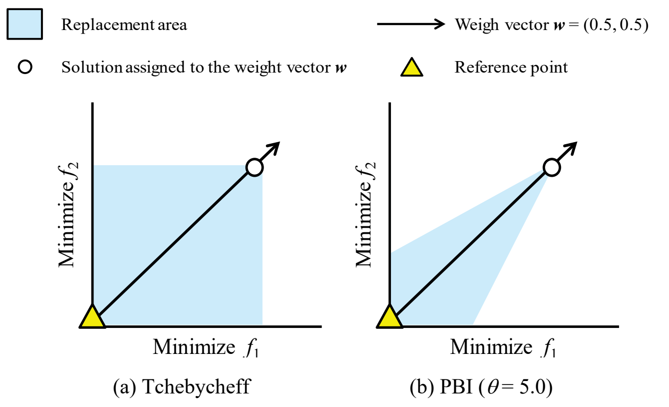
Replacement area
Weigh vector $\boldsymbol{w} = (0.5, 0.5)$
Solution assigned to the weight vector $\boldsymbol{w}$
Reference point
Minimize f_2
Minimize f_1
(a) Tchebycheff
(b) PBI (θ= 5.0)

Fig. 7. A replacement area for the solution of each scalarizing function. A replacement area means the area where the replacement mechanism is performed in MT-MOEA/D.

V. Effects of Parent Solution Selection Methods in Local Mating in Inter-task Crossover

The DE operator generates an offspring solution by adding the weighted difference between two solutions (i.e., $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in (3)) to one solution (i.e., $\boldsymbol{x}_3$ in (3)). As we mentioned in Section III, we consider that it is better to select $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ from the own task and $\boldsymbol{x}_3$ from the other task when the DE operator is applied in inter-task crossover. To confirm this hypothesis, we examine the effects of parent solution selection methods on the search performance of MT-MOEA/D with local mating.

A. Experimental settings

We use the same experimental settings and benchmark problems as in Tables I and II, respectively. In the experiments, we prepare four combinations of selecting the two solutions ($\boldsymbol{x}_1$ and $\boldsymbol{x}_2$) and the one solution ($\boldsymbol{x}_3$). The combinations are summarized in Table V.

TABLE V. Combinations of Parent Selection Methods in the DE Operator in Inter-task Crossover.

	$\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in (3)	$\boldsymbol{x}_3$ in (3)
Type 1	Select from the own task	Select from the other task
Type 2	Select from the other task	Select from the other task
Type 3	Select from the own task	Select from the own task
Type 4	Select from the other task	Select from the own task

MT-MOEA/D with the Type 3 is the same as the independent execution of MOEA/D for each task. This is because inter-task crossover is not actually performed in MT-MOEA/D with the Type 3. However, we also prepare the Type 3 setting in order to examine the effects of parent solution selection methods on the search performance of MT-MOAE/D. We use IGD as a performance measure. We conduct Welch's t-test at the 0.05 significance level to examine whether there exists a statistically significant difference between the results obtained by MT-MOEA/D with Type 1 and each of MT-MOEA/D with Types 2, 3, and 4, respectively.

TABLE VI AVERAGE IGD VALUES BY MT-MOEA/D WITH EACH PARENT SOLUTION SELECTION METHOD IN THE DE OPERATOR WHEN THE TCHEBYCHEFF FUNCTION IS APPLIED TO EACH OF THEM.

Problem	MT-MOEA/D with Type 1		MT-MOEA/D with Type 2		MT-MOEA/D with Type 3		MT-MOEA/D with Type 4	
	Task A	Task B	Task A	Task B	Task A	Task B	Task A	Task B
CI+HS	0.000185	**0.000887**	**0.000185**	0.000967*	0.000576*	0.001036*	0.000818*	0.001335*
CI+MS	0.006446	**0.000960**	**0.001875**	0.001280	0.060006*	0.002628*	0.052114*	0.002688*
CI+LS	**0.000245**	0.000183	0.000248	**0.000181**	1.479076*	0.000250*	1.550047*	0.000250*
PI+HS	0.001004	**0.301045**	0.001071	0.340109	**0.000803***	0.328730	0.000818*	0.464372*
PI+MS	**0.005847**	**12.24740**	0.006010	13.32073	0.006594*	13.73497*	0.006257	12.73055
PI+LS	0.000286	0.116000	0.000268	**0.020217**	0.000273	0.581291*	**0.000243***	0.556183*
NI+HS	**1.497640**	**0.000396**	1.505439*	0.000437*	6.902160*	0.000427*	5.703716*	0.000652*
NI+MS	0.607698	0.007304	**0.595333**	**0.007248**	0.865920*	0.026140*	0.760834	0.022248*
NI+LS	0.000365	0.000296	**0.000324**	0.000260	0.000325	0.000259	0.000425	**0.000245**
p-value	-		0.52773		0.00214		0.00073	

A symbol * denotes that the difference between the result by MT-MOEA/D with Type 1 and that by the corresponding algorithm is statistically significant. The underline denotes that corresponding algorithms obtained better results than MT-MOEA/D with Type 1. The best results among algorithms are highlighted in bold.

TABLE VII AVERAGE IGD VALUES BY MT-MOEA/D WITH EACH PARENT SOLUTION SELECTION METHOD IN THE DE OPERATOR WHEN THE PBI FUNCTION IS APPLIED TO EACH OF THEM.

Problem	MT-MOEA/D with Type 1		MT-MOEA/D with Type 2		MT-MOEA/D with Type 3		MT-MOEA/D with Type 4	
	Task A	Task B	Task A	Task B	Task A	Task B	Task A	Task B
CI+HS	0.000522	**0.00311**	**0.000494***	0.003196	0.001889*	0.004291*	0.002724*	0.004264*
CI+MS	**0.000260**	**0.00052**	0.001468	0.000655	0.062775*	0.000872*	0.050476*	0.000633
CI+LS	0.001147	0.000258	**0.001072***	**0.000224***	1.865537*	0.000597*	1.853896*	0.000574*
PI+HS	**0.002586**	0.377532	0.002684	**0.359856**	0.002741	1.356255*	0.002724	1.693200*
PI+MS	**0.008301**	**8.198671**	0.009311*	8.920801	0.013538*	9.447309*	0.013079*	8.715194
PI+LS	0.001118	0.558582	0.001085	**0.544412**	0.000968*	0.606862*	**0.000807***	0.596866
NI+HS	**1.550457**	**0.001159**	1.554078	0.001161	7.839226*	0.001829*	7.468387*	0.002120*
NI+MS	0.626971	0.00781	**0.536021**	**0.005737**	0.703931	0.029121*	0.618158	0.026560*
NI+LS	0.000249	0.001031	0.000288	0.000966	0.000255	0.000960*	**0.000232**	**0.000784***
p-value	-		0.91330		0.00046		0.00497	

A symbol * denotes that the difference between the result by MT-MOEA/D with Type 1 and that by the corresponding algorithm is statistically significant. The underline denotes that corresponding algorithms obtained better results than MT-MOEA/D with Type 1. The best results among algorithms are highlighted in bold.

B. *Experimental results*

Tables VI and VII show the average IGD values over 101 runs on the benchmark problems. Here, a symbol * denotes that there exists a statistically significant difference between the results by MT-MOEA/D with Type 1 and MT-MOEA/D with the corresponding parent solution selection method. The best results among the parent solution selection methods are highlighted in bold. Moreover, an underline in Tables VI and VII denotes that MT-MOEA/D with the corresponding parent solution selection method obtained better results than MT-MOEA/D with Type 1. We also conduct the paired Wilcoxon test to compare the search performance of MT-MOEA/D with Type 1 and each of the others (i.e., Types 2, 3, and 4) through all the tasks in the nine benchmark problems. Here, the average IGD value of each task represents one observation. That is, there are 18 observations for each algorithm. We show the *p*-values calculated by using the Wilcoxon test in Table VI and VII.

From Tables VI and VII, we can observe that MT-MOEA/D with Type 1 or Type 2 obtained the best results among the parent solution selection methods on many tasks regardless of scalarizing functions. In contrast, there exist few tasks where MT-MOEA/D with Types 3 or 4 outperformed MT-MOEA/D with Type 1. Moreover, the *p*-values between MT-MOEA/D with Type 1 and each of Types 3 and 4 are less than 0.05, as shown in Tables VI and VII. Different from MT-MOEA/D with Types 1 and 2, a solution $\boldsymbol{x}_3$ is selected from its own task in Types 3 and 4. In these cases, an offspring solution is generated near the population for its own task. It is difficult to share solution information between two tasks in Types 3 and 4. Therefore, the effects of solution information sharing on the performance of MT-MOEA/D with Types 3 and 4 are smaller than that of MT-MOEA/D with Types 1 and 2. As a result, MT-MOEA/D with Types 1 and 2 outperformed MT-MOEA/D with Types 3 and 4. This observation suggests that a method for selecting a solution $\boldsymbol{x}_3$ in (3) affects the search performance of MT-MOEA/D.

MT-MOEA/D with Type 1 is the proposed algorithm. In addition, MT-MOEA/D with Type 3 is the same as the independent execution of MOEA/D for each task. Therefore, we can observe whether our proposed algorithm works well or not by comparing MT-MOEA/D with Types 1 and 3. From Tables VI and VII, we can observe that MT-MOEA/D with Type 1 obtained better results than MT-MOEA/D with Type 3 (i.e., MOEA/D) on 13 out of 18 tasks regardless of scalarizing functions. Moreover, the *p*-values between MT-MOEA/D with Types 1 and 3 are less than 0.05. These observations suggest that the search performance of MT-MOEA/D with Type 1 and 3 is statistically different. This may be because the convergence of solution in each population towards the Pareto front is accelerated by solution information sharing. Similar effects are observed in the literature [11]. Since MT-MOEA/D works well compared to MOEA/D, MT-MOEA/D with local mating may be a promising approach for EMMO.

VI. Conclusion

In this paper, first, we proposed MT-MOEA/D, which is a simple extension of MOEA/D for EMMO algorithms and a local mating method in inter-task crossover using uniformly distributed weight vectors in MT-MOEA/D. Next, to examine the effects of the proposed local mating method in inter-task crossover on the search performance of EMMO algorithms, we applied MT-MOEA/D with/without our local mating method to nine benchmark problems. From the experimental results, the following findings were obtained.

1. Local mating in inter-task crossover improved the search performance of MT-MOEA/D regardless of scalarizing functions.
2. When the PBI function is applied, the difference of the search performance between MT-MOEA/D with/without local mating in inter-task crossover was larger as compared to the Tchebycheff function.

We used the DE operator in inter-task crossover. In the DE operator, an offspring solution is generated by adding a weighted difference between two solutions $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ to one solution $\boldsymbol{x}_3$. We examined the effects of parent solution selection methods on the search performance of MT-MOEA/D. From the experimental results, the following findings were obtained.

3. By selecting a solution $\boldsymbol{x}_3$ from the population for the other tasks, the search performance of MT-MOEA/D was improved.
4. MT-MOEA/D with local mating obtained better results than MOEA/D (i.e., MT-MOEA/D in which $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$ are selected from the own task).

One future research topic is to consider a new local mating method for MT-MOEA/D. The local mating method proposed in this paper assumes that the distributions of weight vectors are the same among tasks. This is because our local mating method cannot be applied to MT-MOEA/D when the distribution of weight vectors differs among tasks. Another future topic is to examine the effects of our local mating method in inter-task crossover when we use other crossover methods (e.g., SBX [19] and UNDX [20]) as inter-task crossover. It may show positive effects on the search performance of MT-MOEA/D.

References

[1] H. Ishibuchi, N. Akedo, and Y. Nojima, "Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems," *IEEE Trans. on Evolutionary Computation*, vol. 19, no. 2, pp. 264-283, 2015.

[2] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh, "A survey of multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. on Evolutionary Computation*, vol. 21, no. 3, pp. 440-459, 2017.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

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

[5] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653-1669, 2007.

[6] 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. on Evolutionary Computation*, vol. 18, no. 4, pp. 577-601, 2014.

[7] L. Feng, L. Zhou, J. Zhong, A. Gupta, Y. S. Ong, K. C. Tan, and A. K. Qin, "Evolutionary multitasking via explicit autoencoding," *IEEE Trans. on Cybernetics*, vol. 49, pp. 3457-3470, 2019.

[8] A. Gupta, Y. S. Ong, L. Feng, and K. C. Tan, "Multiobjective multifactorial optimization in evolutionary multitasking," *IEEE Trans. on Cybernetics*, vol. 47, no. 7, pp. 1652-1665, 2017.

[9] J. Mo, Z. Fan, W. Li, Y. Fang, Y. You, and X. Cai, "Multi-factorial evolutionary algorithm based on M2M decomposition," *Lecture Notes Computer Science*, vol. 10593, pp. 134-144, 2017.

[10] C. Yang, J. Ding, K. C. Tan, and Y. Jin, "Two-stage assortative mating for multi-objective multifactorial evolutionary optimization," *Proc. of Annual Conference on Decision and Control*, 2017, pp. 76-81.

[11] R. Hashimoto, N. Masuyama, Y. Nojima, and H. Ishibuchi, "Effect of solution information sharing between tasks on the search ability of evolutionary multiobjective multitasking algorithms," *Proc. of 2019 IEEE Symposium Series on Computational Intelligence*, pp. 2681-2688, 2019.

[12] R. Hashimoto, H. Ishibuchi, N. Masuyama, and Y. Nojima, "Analysis of evolutionary multi-tasking as an island model," *Proc. of the Genetic and Evolutionary Computation Conference Companion*, pp. 1894-1897, 2018.

[13] K. Li, Q. Zhang, S. Kwong, M. Li, and R. Wang, "Stable matching-based selection in evolutionary multiobjective optimization," *IEEE Trans. on Evolutionary Computation*, vol. 18, no. 6, pp. 909-923, 2014.

[14] K. Li, K. Deb, Q. Zhang, and S. Knong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Trans. on Evolutionary Computation*, vol. 19, no. 5, pp. 694-716, 2018.

[15] R. Storn and K. Price, "Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341-359, 1997.

[16] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Computer Science and Informatics*, vol. 26, no. 1, pp. 30–45, 1996.

[17] Y. Yuan, Y. S. Ong, L. Feng, A. K. Qin, A. Gupta, B. Da, Q. Zhang, K. C. Tan, Y. Jin, and H. Ishibuchi, "Evolutionary multitasking for multiobjective continuous optimization: Benchmark problems, performance metrics and baseline results," arXiv: 1706.02766v1[cs.NE], 2017.

[18] S. Jiang, Y. S. Ong, J. Zhang, and L. Feng, "Consistencies and contradictions of performance metrics in multiobjective optimization," *IEEE Trans. on Cybernetics*, vol. 44, no. 12, pp. 2391-2404, 2014.

[19] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, no. 2, pp. 115-148, 1995.

[20] I. Ono and S. Kobayashi, "A real coded genetic algorithm for function optimization using unimodal normal distributed crossover," *Proc. of the Genetic and Evolutionary Computation Conference Companion,* pp. 246-253, 1997.