Large-scale Evolutionary Multiobjective Optimization: An Experimental Study

Tser-Ru Tseng and Tsung-Che Chiang

Dept. of Computer Science and Information Engineering, National Taiwan Normal University, Taiwan (R.O.C.)

<< This paper is included in the Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC) 2024. >>

***Abstract*— Evolutionary multiobjective optimization (EMO) has been a subject of intensive study in the past two decades, owing to its research challenges and practical values. With the progress and development of multiobjective evolutionary algorithms (MOEAs), recent research efforts have shifted to addressing large-scale EMO, which refers to applying evolutionary algorithms to solve multiobjective optimization problems with 100 or more decision variables. In this study, we delve into the design of eight large-scale MOEAs and evaluate their performance under different problem scales and computational resource. Based on the experimental results, we identify suitable algorithms in different scenarios. We also present observations and findings on the relationships between algorithm design concepts and performance.**

I. Introduction

A (continuous) multiobjective optimization problem (MOP) is typically formulated as follows:

$$\begin{aligned} &minimize\ F(X) = \left(f_1(X), \cdots, f_m(X)\right) \\ &subject\ to\ X = (x_1, \cdots, x_D) \in \Omega, \end{aligned} \tag{1}$$

where we aim to search for solutions composed of D decision variables that can optimize (assuming minimization here) m objective functions f_1, f_2, …, and f_m. A solution X is said to dominate a solution Y if and only if X is not worse than Y in terms of all objective functions and is better in terms of at least one objective function. A solution X^* is said to be Pareto optimal if and only if no other solution can dominate it. The goal of solving an MOP is to find the set of Pareto optimal solutions.

Evolutionary algorithms are a class of optimization algorithms. They are recognized as promising approaches to solving MOPs. The so-called evolutionary multiobjective optimization (EMO) sub-domain keeps growing in the last two decades, and many well-known multiobjective evolutionary algorithms (MOEAs) such as NSGA-II [1], NSGA-III [2], MOEA/D [3], and so on, have been proposed. With the progress in the development of MOEAs, recently researchers started to apply MOEAs to solve MOPs with a larger number of decision variables. The MOPs with 100 ore more decision variables are generally regarded as large-scale MOPs [4]. The increase of problem dimensionality causes the exponential explosion of the search space and thus imposes difficulty and challenges to the classic MOEAs. Many useful concepts such as variable grouping and search space reduction have been applied to enable MOEAs to tackle large-scale MOPs. In this paper, we review eight MOEAs dedicated to large-scale MOPs and conduct a numerical study to examine their performance in different scenarios of problem dimensionality and computational budget. We aim to identify suitable algorithms in different scenarios and investigate the relationship between their algorithm design and performance.

The rest of this paper is organized as follows. Section II will give a review of related studies. In Section III, we will describe the core concepts of eight representative MOEAs for large-scale MOPs. Section IV will present the experiments, numerical results, and discussions. Finally, conclusions and future research directions will be given in Section V.

II. Literature Review

MOEAs dedicated to large-scale MOPs are usually categorized into three types: MOEAs based on variable grouping, MOEAs based on search space reduction, and MOEAs based on novel search strategies [4]. The following three sub-sections will present a brief review of these three types of MOEAs respectively.

A. Variable Grouping

The difficulty of large-scale MOPs comes from the exponential growth of the search space with the increasing number of decision variables. Thus, a useful way to tackle this difficulty is not to consider all variables at a time but consider only a subset of variables. More specifically, we divide the decision variables into several groups and solve each group separately. This is the core idea of cooperative coevolution (CC) [5], and it is utilized in many MOEAs to solve large-scale MOPs.

How to group variables is the key design issue of the variable-grouping-based MOEAs. CCGDE3 [6] is one of the very early representative of MOEAs based on variable grouping. It simply groups variables in a random manner and fixes the groups. MOEA/D-RDG [7] also groups variables randomly, but it changes the grouping periodically and adjusts the group size adaptively.

MOEA/DVA [8] conducts decision variable analysis and classifies variables into distance, position, and mixed variables. Then, it does interaction analysis on distance variables and then groups variables based on their interaction relationship. LMEA [9] uses the same method of variable grouping as MOEA/DVA does and improves the method of classifying variables. PCA-MOEA [10] adds a dimension reduction step and then does variable grouping. LERD [11] regards the variable grouping as an optimization problem and groups variables based on the optimization results.

B. Space Reduction

To deal with the difficulty of a large-scale MOP, the basic idea of space-reduction-based MOEAs is similar to that of variable-grouping-based MOEAs – solving a problem with a smaller scale. Variable-grouping-based MOEAs reduce the problem scale by considering only a subset of decision variables at a time, whereas space-reduction-based MOEAs consider all variables (not a subset) but only searches in a limited space.

WOF [12] is the first representative of space-reduction-based MOEAs. It groups the original decision variables, assigns a weight variable to each group, and transforms the original optimization problem of D decision variables into an optimization problem of $\lceil D/S \rceil$ weight variables. (Let S denote the number of decision variables in each group.) The decision variables x_i in the same group k change their values together only by multiplying the value of the corresponding weight variable w_k. In this way, the search is conducted in a limited space. WOF is a framework, and some variants such as WOF-MMOPSO-RDG [13] were proposed. Like WOF, LSMOF [14] considers all decision variables simultaneously and searches in a limited space. The difference is in that it defines the limited search space on the lines connecting selected solutions in the population and the boundary points composed of the minimum and maximum values in the decision space. The idea of directional sampling was also adopted but realized in different ways in LMOEA-DS [15], DGEA [16] and FLEA [17]. Details of these algorithms will be given in the next section.

C. Novel Search Strategy

This type of MOEAs do not attempt to reduce the problem scale; instead, they deal with the huge search space of large-scale MOPs by generating high quality solutions more effectively. LMOCSO [18] was developed based on competitive swarm optimization (CSO) [19]. The particle updating strategy of CSO considers the velocity by moving from a worse particle to a better particle. LMOCSO revises the updating strategy by adding a term representing the acceleration. The experimental results showed that the revised strategy generates solutions with higher diversity. CCSO [20] is another improved CSO for large-scale EMO. CCSO uses three types of competition. The environmental competition classifies solutions into two equal-size groups as winners and losers. Each loser learns from two winners, which are chosen by cognitive competition and social competition respectively. Finally, all winners and learned losers undergo mutation to form the next population. Hong et al. [21] identified the difficulty of large-scale MOPs with correlation between position and distance functions. They proposed DLS-MOEA based on a well-known indicator-based MOEA, SMS-EMOA [22]. The main contribution is to enhance the diversification ability of SMS-EMOA by maintaining an external archive and applying a local search procedure to the archived solutions. Yi et al. [23] examined the performance of three typical crossover operators in NSGA-III. They found that the uniform crossover outperforms the SBX and one-point crossover. Integration of neural networks into MOEAs could also be a promising research direction. Liu et al. [24] applied a three-layer neural network as a reproduction operator in their ALMOEA, which showed good performance.

III. Large-Scale Multiobjective Evolutionary Algorithms

A. Algorithm Concepts and Relationship

In this section we will present the core concepts of the eight algorithms that we choose to do numerical experiments and comparison. We identify the core concepts of these algorithms and illustrate their relationship in Fig. 1.

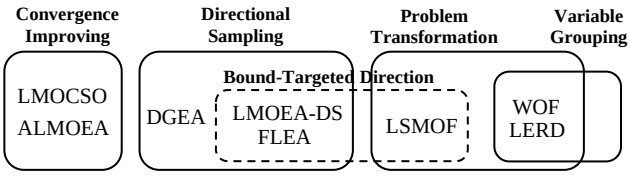

Fig. 1. Core concepts and relationship

WOF, LSMOF, and LERD are three algorithms that do problem transformation, i.e. building and solving a new problem besides the original problem. Among them, WOF and LERD also involve the concept of variable grouping. LMOEA-DS, DGEA, and FLEA are three algorithms that generate solutions through directional sampling. LMOEA-DS takes the boundary points in the decision space as end points of the direction lines, and this concept is also utilized by LSMOF and FLEA. LMOCSO and ALMOEA try some other ways to generate new solutions effectively.

B. WOF [12]

WOF is a representative of the space-reduction-based MOEAs. It alternates between the normal optimization and weighting optimization steps. The normal optimization step is a standard procedure where a typical MOEA such as NSGA-II is applied to solve the original problem. Space reduction is realized in the weighting optimization step, where a set of weight variables but not the set of the original decision variables is to be optimized. After the normal optimization step is carried out, a set of q good solutions are selected. Then, a transformed problem is created based on each of these selected solutions. Let $X = (x_1, x_2, \ldots, x_D)$ denote a selected solution composed of D decision variables. The decision variables are separated into $k = \lceil D/S \rceil$ groups, each group (except the last one) consisting of S variables. Each group is associated with a weight variable, and the vector of weight variables plays the role of sampling solutions in a reduced space. Specifically, a weight vector $W = (w_1, w_2, \ldots, w_k)$ corresponds to a solution

$$Y = (w_1x_1, \ldots, w_1x_s, w_2x_{s+1}, \ldots, w_2x_{2s}, \ldots, w_kx_D). \tag{2}$$

The weighting optimization step searches in the space of weight vectors directly and in the reduced space of the original decision vectors indirectly. The weighting optimization is carried out for each of the q good solutions selected from the normal population. One weight solution is chosen at the end of each weighting optimization and is applied to all solutions in the normal population. Finally, the best N (the population size) solutions are selected from the union of all (N+q·N) solutions to start the next round of normal optimization. Fig. 2 illustrates the case with two decision variables and one weight variable. By changing the value of the weight variable, WOF generates solutions on the lines connecting the selected solutions and the origin point in the decision space.

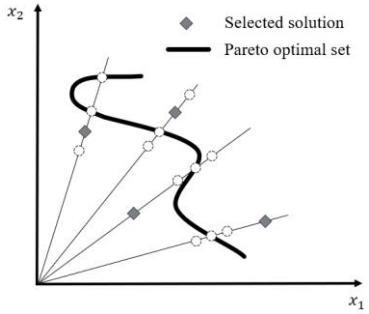

Fig. 2. Weight optimization in WOF (adapted from [12])

C. LERD [11]

The authors of LERD pointed out two disadvantages of existing variable-grouping-based methods – high computational cost and static grouping. LERD aims to identify the interaction relationship between variables dynamically through solving a reformulated problem. In the reformulated problem, each solution is a binary vector $B = \{b_1, b_2, \ldots b_D\}$ and represents a group of decision variables. The value of b_i as one implies that the corresponding decision variable x_i in the original problem is selected in the group. To evaluate a binary vector B, a solution X is randomly selected from the population of the original problem and then n_s new solutions are generated by perturbing the values of only variables belonging to the group represented by B. The sum of the normalized objective values of these n_s solutions is taken as the first objective of B, and the number of 1-bit in B (i.e. the group size) is taken as the second objective of B. The reformulated problem is actually a bi-objective optimization problem to find a set of variable groups that lead to quick convergence and have small group sizes. After solving the reformulated problem and obtaining the groups, LERD will do further convergence and diversity optimization procedures for each group of variables.

D. LSMOF [14]

LSMOF is another representative of the space-reduction-based MOEAs. Different from WOF, LSMOF does not need to do variable grouping. In addition, LSMOF carries out the optimization steps in the reduced space and in the original space sequentially, not alternatively. It reduces the search space by sampling only on the lines connecting selected solutions and the boundary points in the decision space. Let s_i denote a selected solution, o denote the lower boundary point (i.e. the point composed of the minimum values of all decision variables), and t denote the upper boundary point. LSMOF uses two weight variables λ_{i1} and λ_{i2} to sample solutions p_{i1} and p_{i2} on the lines, as formulated in (3) and (4). Fig. 3 illustrates the sampling process.

$$\begin{aligned} v_l &= s_i - o \\ v_u &= t - s_i \end{aligned} \tag{3}$$

$$\begin{aligned} p_{i1} &= o + \lambda_{i1} \frac{v_l}{\|v_l\|} \|t - o\| \\ p_{i2} &= t - \lambda_{i2} \frac{v_u}{\|v_u\|} \|t - o\| \end{aligned} \tag{4}$$

LSMOF selected r solutions from the population of the original problem. A weight vector $\Lambda = \{\lambda_{11}, \lambda_{12}, \ldots, \lambda_{r1}, \lambda_{r2}\}$ represents a set of $2 \cdot r$ solutions sampled by the method mentioned above. LSMOF attempts to find the optimal weight vector in terms of the hypervolume of the set of solutions sampled by the corresponding weight vector. In other words, LSMOF transforms the original optimization problem with D variables into a weight optimization problem with $2 \cdot r$ weight variables. It solves the weight optimization problem with half of the total computational budget and then solves the original problem with the other half.

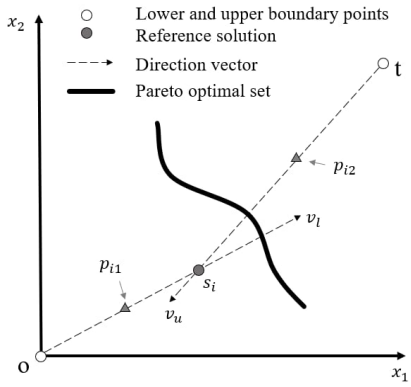

Fig. 3. Directional sampling in LSMOF (adapted from [14])

E. LMOEA-DS [15]

The core idea of LMOEA-DS is similar to that of LSMOF – sampling on the lines connecting the representative solutions and the boundary points in the decision space. The main difference is that LSMOF formulates the sampling process as an optimization problem but LMOEA-DS does not. LMOEA-DS calls these sampled solutions the guiding solutions and lets them participate into the reproduction process. In other words, there are two reproduction processes and two environmental selection in each generation. In the first reproduction process, the ordinary population P_t mate with the set of guiding solutions G_t to generate offspring solutions O_t'. The intermediate population P_t' is selected from the union of P_t, G_t, and O_t' by the authors' proposed complementary environmental selection. Then, the second round of reproduction and environmental selection are carried out on P_t' to generate the population P'_{t+1} for the next generation.

F. DGEA [16]

DGEA aims to develop an effective offspring generation method and also uses the concept of directional sampling of LSMOF and LMOEA-DS. Its method can be regarded as an improved mutation operator of differential evolution (DE). Given a base solution X and a direction V, a typical DE mutation operator generates a single new solution by $Y = X + F \cdot V$, where F is a scaling factor. In DGEA, the base solution is randomly chosen from the non-dominated solutions. For each base solution, DGEA generates multiple direction vectors by connecting the base solution and multiple reference solutions. Given a base solution X and a direction V, it generates multiple new solutions by $Y_j = X + r_j \cdot V$, where r_j is a random value following the Gaussian distribution $N(0, \sigma^2)$. The value of σ^2 is the variance of the distances of all non-dominated solutions to the base solution X along the direction V. DGEA generates direction vectors to enhance the diversity (resp. convergence) when there are few (resp. enough) non-dominated solutions. Fig. 4 illustrates the sampling process.

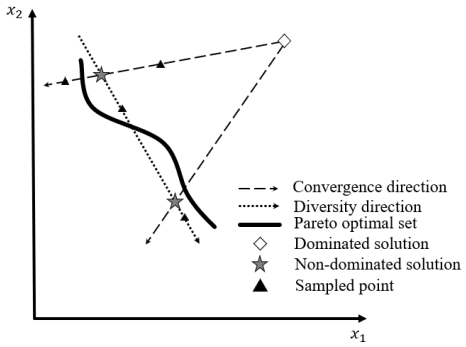

Fig. 4. Directional sampling in DGEA (adapted from [16])

G.

H. FLEA [17]

FLEA can be regarded as an improved version of DGEA. They share two key concepts: (1) building the search directions considering both convergence and diversity and (2) doing Gaussian sampling on the search directions. Comparing with DGEA, FLEA has three modifications. First, FLEA introduces the neighborhood when building the search directions. After a set of reference solutions are selected, each solution in the population is assigned to the neighborhood of these reference solutions in terms of Chebyshev distance. In addition, each reference solution has two types of neighborhoods, one in the decision space and the other in the objective space. When a reference solution dominates all its neighbor solutions, FLEA builds a search direction between the reference solution and the center of its neighbor solutions. Second, FLEA considers the lines connecting the reference solutions and the boundary points in the decision space as search directions. (LSMOF and LMOEA-DS also consider these directions, but DGEA does not.) These directions are built when all solutions in the neighborhood of a reference solution are non-dominated. Third, in each generation the reproduction process of FLEA uses all reference solutions as the starting points to sample solutions on the search directions but DGEA fixes to use only a single reference solution to be the starting point.

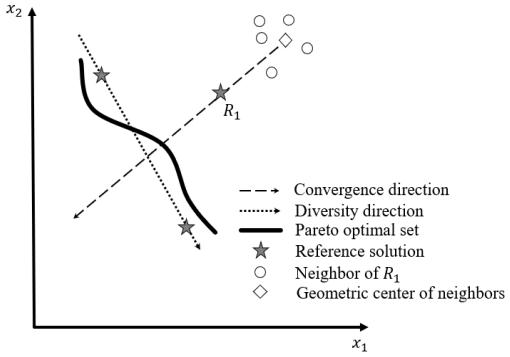

Fig. 5. Directional sampling in FLEA (adapted from [17])

I. LMOCSO [18]

LMOCSO does not use the concept of variable grouping or space reduction; instead, it focuses on improving the particle updating strategy of CSO. The authors of LMOCSO observed that the standard updating strategy of CSO moves the particle X_l toward the leading particle X_w slowly since X_l moves closer to the X_w by the attracting velocity $\vec{\Delta}(t)$ but may move away due to the original velocity $\overrightarrow{v_l}(t)$, as shown in (5)–(6). They improved the strategy by replacing $\overrightarrow{X_w}(t)$ and $\overrightarrow{X_l}(t)$ in (5) by $\overrightarrow{X_w}'(t)$ and $\overrightarrow{X_l}'(t)$ in (7)–(8). Simply speaking, the new strategy moves the particle by its original velocity first and then moves it toward the leading particle by $\vec{\Delta}(t)$ to reduce the negative effect of the original velocity.

$$\vec{\Delta}(t) = \overrightarrow{X_w}(t) - \overrightarrow{X_l}(t) \tag{5}$$

$$\overrightarrow{X_l}(t+1) = \overrightarrow{X_l}(t) + r_0\overrightarrow{v_l}(t) + r_1\vec{\Delta}(t) \tag{6}$$

$$\overrightarrow{X_w}'(t) = \overrightarrow{X_w}(t) + r_0\overrightarrow{v_w}(t) \tag{7}$$

$$\overrightarrow{X_l}'(t) = \overrightarrow{X_l}(t) + r_0\overrightarrow{v_l}(t) \tag{8}$$

J. ALMOEA [24]

The directions from worse solutions to better solutions are utilized in several algorithms such as DGEA, FLEA, and LMOCSO. Unlike DGEA and FLEA, which do Gaussian sampling on the search directions, ALMOEA uses the normal DE mutation operator. The novel design is the use of a neural network to learn the search directions. It sorts the solutions based on the non-domination rank and the crowding distance. The better half of the population is S_e, and the other half is S_p. Each solution X in S_p is paired with a solution Y in S_e with the minimal acute angle. A three-layer neural network is trained by these pairs of (X, Y) to learn the convergence directions from X to Y. In each generation, the neural network is trained with one epoch and is then used to generate the guiding solutions. More specifically, each solution X in the population generates a new solution by

$$X^{\text{new}} = X^{old} + r_1 \cdot (X^{old} - X^{\text{gdv}}) + r_2 \cdot (X^{d1} - X^{d2}), \tag{9}$$

where X^{gdv} is the output of the neural network with X^{old} as the input.

IV. Experiments and Results

A. Experimental Setting

We investigated the performance of the eight selected algorithms by the LSMOP benchmark set [25], which consists of nine functions. We set the number of decision variables (D) by 100, 500, 1000, and 2000 to examine the algorithm performance with respect to the problem scale. The number of objectives was set by two and three. The population size (N) was set by 100 and 105 when the number of objectives was two and three, respectively. The maximum number of fitness function evaluations (MaxFFE) was set by $500{\cdot}N$, $1000{\cdot}N$, $5000{\cdot}N$, and $10000{\cdot}N$ to examine the algorithm performance with respect to different levels of computational resource.

We carried out the experiments by using PlatEMO 4.2 [26]. The parameter values of all tested algorithms followed the setting in PlatEMO. The only modification was that we changed the base algorithm of LSMOF and FLEA from NSGA-II to NSGA-III since other compared algorithms are based on newer MOEAs. We also replaced the base algorithms NSGA-II and MOEA/D in LERD by NSGA-III. However, the new version did not provide better performance, and thus we kept the original implementation. The base algorithm in the original paper of ALMOEA is MOEA/D, but the implementation in PlatEMO is θ-DEA [27].

B. Performance Comparison

The performance indicator is the inverted generational distance (IGD). We used the IGD calculation function in PlatEMO. We ran each algorithm A_i to solve each LSMOP function f_j with D decision variables and m objectives under T fitness function evaluations for 20 times. We calculated the average IGD values and got the rank $r_i(j, D, m, T)$ of algorithm A_i. Then, we calculated the average rank $r_i(D, T)$ over test functions and the number of objectives. Fig. 6 shows the average ranks of eight algorithms for different problem scale D and computational resource T. A smaller rank and lighter color represent better performance.

Based on the results in Fig. 6, we first categorize the algorithms into two groups. The algorithms in the first group are more suitable for solving larger-scale problems with lower computational resource (lighter color at the bottom left corner in the heat maps). The algorithms in the second group are more suitable for solving smaller-scale problems with higher computational resource (lighter color at the upper right corner).

The top three algorithms on the left, WOF, LMOEA-DS, and LSMOF belong to the first group. The common design concept of these algorithms is that they all searched in a reduced space, either by transforming the original solution space into a weight space (WOF and LSMOF) or by restricting the space in the lines connecting current solutions and boundary points (LSMOF and LMOEA-DS). The results confirm that their designs are effective to make them perform better when the problem scale is larger and the computational resource is lower. ALMOEA also belongs to the first group. It utilizes a neural network to learn the mapping from worse solutions to better solutions and uses the mapping to improve solutions. The mapping could be regarded as a kind of reduction of the search space, which makes it have similar behaviors like the other three algorithms in the first category.

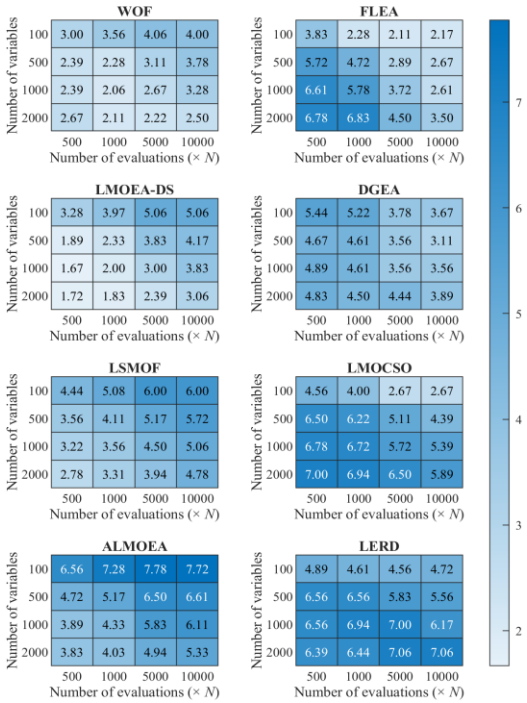

Fig. 6. Average ranks of eight algorithms for different problem scale and computational resource

The top three algorithms on the right, FLEA, DGEA, and LMOCSO belong to the second group. Comparing with the algorithms in the first group, these three algorithms allow larger search space. On one hand, the large search space causes difficulty of finding good solutions and thus these algorithms could not solve the problems well with low computational resource. On the other hand, the large search space contains more high-quality solutions, and these algorithms could find good solutions when they have high computational resource. LERD also belongs to the second group. Its core algorithm design is to do variable grouping, which requires a lot of computational resource. Therefore, its performance gets better when it is given higher computational resource.

Table I shows the results of Wilcoxon test with the significance level 0.05. We count the number of LSMOP functions that a tested algorithm is better than, equal to, and worse than WOF in each scenario of D and MaxFFE. WOF and LMOEA-DS are good options when the problem scale is large and the computational budget is low; FLEA and DGEA are good choices when the problem scale is small and the computational budget is high.

C. More Discussions

As we presented in Section III-A, there are some relationships and similarity between the tested algorithms. In the following we discuss why the algorithms have similar ideas have different performance results.

WOF vs. LSMOF: WOF and LSMOF are two representative algorithms that reduce the search space by problem transformation. In our experiments, LSMOF is outperformed by WOF. We think that LSMOF might impose too much reduction. First, LSMOF restricts all variables to change in the same direction together, while WOF allows different groups of variables change in different directions. Second, LSMOF fixes one end point of the direction to be the boundary point in the decision space, while WOF has no such restriction.

LMOEA-DS vs. LSMOF: LMOEA-DS and LSMOF both use the boundary points to guide the search directions. Specifically, they sample solutions on the lines connecting current solutions and the boundary points. In our experiments, LMOEA-DS performs better than LSMOF. The possible reason is that LMOEA-DS has double reproduction processes, where the sampled solutions participate in the crossover process to generate solutions with higher diversity.

LMOEA-DS & LSMOF vs. FLEA & DGEA: These four algorithms all use the idea of directional sampling, but the former two algorithms and the latter two algorithms belong to two different groups. The main difference of the algorithm design between the two groups is that the former two fix the boundary points as one end point of the search directions but the latter two do not. Sticking to searching toward the boundary points could help LMOEA-DS and LSMOF to perform better when the problem scale is larger and the computational resource is low. When the computational resource is high enough, however, the free search directions enable FLEA and DGEA to perform better.

FLEA vs. DGEA: FLEA and DGEA both use directional sampling to generate new solutions. We mentioned the difference between them in Section III-G. The use of multiple directions in FLEA gives it a larger search space, and thus FLEA performs better when the problem is smaller and the computational resource is higher.

V. Conclusions

Developing MOEAs to deal with large-scale MOPs is an emerging area in the domain of EMO, and many algorithms have been proposed. However, few studies investigated the algorithm performance with respect to both the problem scale and computational resource. There is also lack of discussions about the algorithmic similarity and performance difference. In this paper we selected eight representative algorithms, elaborated their core design concepts, and evaluated their performance through comprehensive experiments. Our findings can be summarized as follows:

- WOF and FLEA are two promising algorithms regarding different problem dimensionality and computational resource.
- Reduction of search space helps to find good solutions effectively when the problem scale is large and the computational resource is low, while it may hinder the algorithms to find good solutions when the problem is small and the computational resource is high.
- Directional sampling is an effective technique. When we use this technique, we need to think about the use of boundary points and the number of directions in difference scenarios of problem scale and computational resource.

Our future work will explore two directions. Firstly, we plan to validate our findings and observations by assessing the performance of these algorithms using other benchmark function sets such as LMF [28]. Secondly, we will develop some new algorithms by integrating the strengths and features of the existing algorithms.

TABLE I. COMPARISON RESULTS BETWEEN WOF AND SEVEN ALGORITHMS BY THE WILCOXON TEST

		LMOEA-DS	DGEA	FLEA	LMOCSO	LSMOF	LERD	ALMOEA
D	MaxFFE	+/=/−	+/=/−	+/=/−	+/=/−	+/=/−	+/=/−	+/=/−
100	500 × *N*	5/5/8	2/5/11	7/6/5	7/3/8	2/2/14	6/3/9	1/1/16
	1000 × *N*	6/4/8	5/3/10	11/4/3	8/5/5	2/3/13	6/5/7	0/2/16
	5000 × *N*	5/3/10	9/4/5	12/4/2	12/2/4	0/4/14	7/4/7	0/0/18
	10000 × *N*	6/2/10	9/3/6	14/2/2	13/1/4	1/3/14	6/5/7	0/0/18
500	500 × *N*	7/5/6	4/2/12	0/2/16	2/0/16	3/5/10	0/0/18	1/5/12
	1000 × *N*	4/6/8	4/3/11	2/3/13	3/1/14	2/4/12	0/1/17	1/2/15
	5000 × *N*	4/5/9	7/7/4	9/1/8	6/2/10	2/3/13	3/4/11	1/0/17
	10000 × *N*	5/3/10	10/4/4	8/6/4	8/3/7	1/4/13	6/2/10	0/1/17
1000	500 × *N*	8/7/3	3/3/12	0/0/18	1/2/15	6/3/9	0/0/18	2/3/13
	1000 × *N*	6/5/7	3/4/11	0/1/17	2/0/16	3/4/11	0/0/18	1/3/14
	5000 × *N*	6/2/10	5/2/11	4/6/8	3/1/14	2/5/11	0/1/17	2/0/16
	10000 × *N*	4/5/9	7/4/7	8/4/6	5/1/12	2/5/11	3/1/14	1/1/16
2000	500 × *N*	7/6/5	4/3/11	0/0/18	2/0/16	6/4/8	0/0/18	3/4/11
	1000 × *N*	6/5/7	4/3/11	0/0/18	1/1/16	4/2/12	0/0/18	1/4/13
	5000 × *N*	6/2/10	3/4/11	2/3/13	2/0/16	3/3/12	0/0/18	0/3/15
	10000 × *N*	5/3/10	4/5/9	5/5/8	2/1/15	2/3/13	0/1/17	2/2/14
Total		90/68/130	83/59/146	82/47/159	77/23/188	41/57/190	37/27/224	16/31/241

ACKNOWLEDGMENT

This research is supported by the National Science and Technology Council (NSTC), Taiwan, R.O.C. under Grant no. NSTC 112-2221-E-003-012.

REFERENCES

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

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

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

[4] Y. Tian, L. Si, X. Zhang, R. Cheng, C. He, K. C. Tan, and Y. Jin, "Evolutionary large-scale multi-objective optimization: A survey," *ACM Computing Surveys*, vol. 54, no. 8, pp. 1–34, 2021.

[5] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. International Conference on Parallel Problem Solving from Nature*, pp. 249–257, 1994.

[6] L. M. Antonio and C. A. C. Coello, "Use of cooperative coevolution for solving large scale multiobjective optimization problems," in *Proc. IEEE Congress on Evolutionary Computation*, pp. 2758–2765, 2013.

[7] A. Song, Q. Yang, W.-N. Chen, and J. Zhang, "A random-based dynamic grouping strategy for large scale multi-objective optimization," in *Proc. IEEE Congress on Evolutionary Computation*, pp. 468–475, 2016. [MOEA/D-RDG]

[8] X. Ma, et al. "A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with largescale variables," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 2, pp. 275–298, 2016. [DVA]

[9] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 97–112, 2018. [LMEA]

[10] R. Liu, R. Ren, J. Liu, and J. Liu, "A clustering and dimensionality reduction based evolutionary algorithm for large-scale multi-objective problems," *Applied Soft Computing*, vol. 89, Article 106120, 2020. [PCA-MOEA]

[11] C. He, R. Cheng, L. Li, K. C. Tan, and Y. Jin, "Large-scale multiobjective optimization via reformulated decision variable analysis." *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 1, pp. 47–61, 2024. [LERD]

[12] H. Zille, H. Ishibuchi, S. Mostaghim, and Y. Nojima, "A framework for large-scale multiobjective optimization based on problem transformation," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 260–275, 2018. [WOF]

[13] R. Liu, J. Liu, Y. Li, and J. Liu, "A random dynamic grouping based weight optimization framework for large-scale multi-objective optimization problems," *Swarm and Evolutionary Computation*, vol. 55, Article 100684, 2020. [WOF-MMOPSO-RDG]

[14] C. He, L. Li, Y. Tian, X. Zhang, R. Cheng, Y. Jin, and X. Yao, "Accelerating large-scale multiobjective optimization via problem reformulation," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 6, pp. 949–961, 2019. [LSMOF]

[15] S. Qin, C. Sun, Y. Jin, Y. Tan, and J. Fieldsend, "Large-scale evolutionary multiobjective optimization assisted by directed sampling," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 724–738, 2021. [LMOEA-DS]

[16] C. He, R. Cheng, and D. Yazdani, "Adaptive offspring generation for evolutionary large-scale multiobjective optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 2, pp. 786–798, 2020. [DGEA]

[17] L. Li, C. He, R. Cheng, H. Li, L. Pan, and Y. Jin, "A fast sampling based evolutionary algorithm for million-dimensional multiobjective optimization," *Swarm and Evolutionary Computation*, Article 101181, 2022. [FLEA]
[18] Y. Tian, X. Zheng, X. Zhang, and Y. Jin, "Efficient large-scale multiobjective optimization based on a competitive swarm optimizer," *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3696–3708, 2020. [LMOCSO]
[19] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 191–204, 2015. [CSO]
[20] S. Liu, Q. Lin, Q. Li, and K. C. Tan, "A comprehensive competitive swarm optimizer for large-scale multiobjective optimization," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 52, no. 9, pp. 5829–5841. [CCSO]
[21] W. Hong, K. Tang, A. Zhou, H. Ishibuchi, and X. Yao, "A scalable indicator-based evolutionary algorithm for large-scale multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 525–537, 2019. [DLS-MOEA]
[22] N. Beumea, 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. [SMS-EMOA]
[23] J.-H. Yi, L.-N. Xing, G.-G. Wang, J. Dong, A. V. Vasilakos, A. H. Alavi, and L. Wang, "Behavior of crossover operators in NSGA-III for large-scale optimization problems," *Information Sciences*, vol. 509, pp. 470–487, 2020. [USGA-III-UC]
[24] S. Liu, J. Li, Q. Lin, Y. Tian, and K. C. Tan, "Learning to accelerate evolutionary search for large-scale multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 1, pp. 67–81, 2023. [ALMOEA]
[25] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "Test problems for largescale multiobjective and many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4108–4121, 2017. [LSMOP]
[26] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: A MATLAB platform for evolutionary multi-objective optimization," *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, 2017. [PlatEMO]
[27] Y. Yuan, H. Xu, B. Wang, and X. Yao, "A new dominance relation-based evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 16–37, 2015. [θ-DEA]
[28] S. Liu, Q. Lin, K.-C. Wong, Q. Li, and K. C. Tan, "Evolutionary large-scale multiobjective optimization: Benchmarks and algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 3, pp. 401–415, 2023. [LMF]