Article

Enhancing Multi-Objective Optimization with Automatic Construction of Parallel Algorithm Portfolios

Xiasheng Ma [1,2], Shengcai Liu [3,*] and Wenjing Hong [4]

1 School of Mechanical Engineering, University of Science and Technology Beijing, Beijing 100083, China; b20190270@xs.ustb.edu.cn
2 Beijing Key Laboratory of Research and Application for Robotic Intelligence of Hand-Eye-Brain Interaction, Beijing 100190, China
3 Centre for Frontier AI Research, Agency for Science, Technology and Research, Singapore 138632, Singapore
4 Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China; hongwj@sustech.edu.cn
* Correspondence: liu_shengcai@cfar.a-star.edu.sg

Abstract: It has been widely observed that there exists no universal best Multi-Objective Evolutionary Algorithm (MOEA) dominating all other MOEAs on all possible Multi-Objective Optimization Problems (MOPs). In this work, we advocate using the Parallel Algorithm Portfolio (PAP), which runs multiple MOEAs independently in parallel and gets the best out of them, to combine the advantages of different MOEAs. Since the manual construction of PAPs is non-trivial and tedious, we propose to automatically construct high-performance PAPs for solving MOPs. Specifically, we first propose a variant of PAPs, namely MOEAs/PAP, which can better determine the output solution set for MOPs than conventional PAPs. Then, we present an automatic construction approach for MOEAs/PAP with a novel performance metric for evaluating the performance of MOEAs across multiple MOPs. Finally, we use the proposed approach to construct an MOEAs/PAP based on a training set of MOPs and an algorithm configuration space defined by several variants of NSGA-II. Experimental results show that the automatically constructed MOEAs/PAP can even rival the state-of-the-art multi-operator-based MOEAs designed by human experts, demonstrating the huge potential of the automatic construction of PAPs in multi-objective optimization.

Keywords: multi-objective optimization; evolutionary algorithm; algorithm portfolio

1. Introduction

Multi-Objective Optimization Problems (MOPs) are a class of classical optimization problems, which widely exist in many real-world applications such as finance [1], vehicle routing [2], and adversarial attack [3]. Multi-Objective Evolutionary Algorithms (MOEAs), as a kind of algorithm for solving MOPs, have attracted great attention over the past few decades and have shown a good performance [4–10]. Generally, MOEAs follow the generational framework where a population of solutions evolve from one generation to the next. In the evolution process, evolutionary operators (e.g., mutation and crossover) are used to generate offspring through the parents, while environmental selection operators are used to select elite individuals for the next-generation evolution.

Despite the tremendous success achieved by MOEAs, it has been widely observed [11–14] that there exists no universal best MOEA that dominates over all other MOEAs on all possible MOPs. Instead, different MOEAs are good at solving different MOPs due to their abilities on exploitation and exploration [15]. For example, MOEAs equipped with polynomial-based mutation (PM) [16] are good at searching in a local area, while those using simulated binary crossover (SBX) [17] and differential evolution (DE) [18,19] are capable of exploring the search space globally. Hence, in order to achieve a better overall performance for a diverse range of MOPs, it is natural and intuitive to combine

the advantages of different MOEAs. One notable series of research efforts following this idea are multi-operator-based MOEAs [11–14], which adaptively allocate computational resources to MOEAs equipped with different operators when solving a MOP.

Apart from multi-operator-based MOEAs, and from a more general perspective of problem solving, there is an effective technique that exploits the complementarity between different algorithms by including them into a so-called algorithm portfolio (AP). To utilize an AP to solve a problem, Tang et al. [20,21] proposed a simple but effective strategy, called a parallel algorithm portfolio (PAP), that runs all member algorithms in the portfolio independently in parallel to obtain multiple solutions. Then, the best solution will be taken as the final output of the PAP. Although a PAP would consume more computational resources than a single algorithm, it has three important advantages. First, PAPs are easy-to-implement because they do not necessarily require any resource allocation since each member algorithm is simply assigned with the same amount of resource. Second, the performance of a PAP on any problem is the best performance achieved among its member algorithms on the problem. In other words, a PAP could achieve a much better overall performance than any of its member algorithms. Third, considering the tremendous growth of parallel computing architectures [22] (e.g., multi-core CPUs) over the last few decades, leveraging parallelism has become very important in designing effective solvers for hard optimization problems [23–27]. PAPs employ parallel solution strategies and thus allow for the use of modern computing facilities in an extremely simple way.

It is conceivable that any PAP's effectiveness relies heavily on the diversity and complementarity among its member algorithms. In other words, the manual construction of high-quality PAPs is generally a challenging task, requiring domain experts (with a deep understanding of both algorithms and problems) to explore the vast design space of PAPs, which cannot be carried out manually with ease [28–30]. As an alternative, Tang and Liu [25,26] proposed a general framework, called automatic construction of PAPs, that seeks to automatically build PAPs by selecting the member algorithms from an algorithm configuration space, with the goal of optimizing the performance of the resulting PAP on a given problem set (called training set). Such a framework has been shown to be effective in building high-performance PAPs for combinatorial problems such as the Boolean Satisfiability Problem (SAT) [25], the Traveling Salesman Problem (TSP) [27,31], and the Vehicle Routing Problem (VRP) [26].

However, to the best of our knowledge, the potential of the automatic construction of PAPs has not been investigated in the area of multi-objective optimization. Considering its excellent performance on the above-mentioned problems and the practical significance of MOPs, studying how to utilize it to solve MOPs is thus valuable. In this work, we focus on automatically building PAPs for continuous MOPs. On the other hand, as a general framework, appropriately instantiating automatic PAP construction for a specific problem domain is non-trivial. Specifically, it requires careful designs of the algorithm configuration space and the performance metrics used in the construction process [26].

The main contributions of this work can be summarized as follows.

1. Taking the characteristics of MOPs into account, we propose a novel variant form of PAP for MOPs, dubbed MOEAs/PAP. Its main difference from conventional PAPs lies in the method for determining the final output. MOEAs/PAP would compare the solution sets found by member algorithms and the solution set generated based on all the solutions found by member algorithms, and finally output the best solution set.
2. We present an automatic construction approach for MOEAs/PAP with a novel metric that evaluates the performance of MOEAs/PAPs across multiple MOPs.
3. Based on a training set of MOPs and an algorithm configuration space defined by several variants of NSGA-II, we use the proposed approach to construct an MOEAs/PAP, namely NSGA-II/PAP. Experimental results show that NSGA-II/PAP significantly outperforms existing single-operator-based MOEAs and the state-of-the-art multi-operator-based MOEAs designed by human experts. Such promising results indicate the huge potential of automatic construction of PAPs in multi-objective optimization.

The remainder of this paper is organized as follows. Section 2 presents the preliminaries and briefly reviews the related works. Section 3 gives the variant form of PAP for MOPs. The automatic construction approach, as well as the algorithm configuration space, the training set, and the performance metric used in construction, are presented in Section 4. Section 5 presents the experimental study. Finally, Section 6 concludes the paper.

2. Preliminaries and Related Work

2.1. Multi-Objective Optimization Problems

Without loss of generality, in this work we assume optimization problems take the minimization form. MOPs are optimization problems with multiple objectives, defined as follows:

$$
\begin{aligned}
\min F(\mathbf{x}) &= [f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_m(\mathbf{x})] \in \Pi \\
\mathbf{x} &= [x_1, x_2, \cdots, x_n] \in D
\end{aligned}
\tag{1}
$$

where $\mathbf{x}$ is the decision vector, which consists of n decision variables. The objective vector $F : D \to \Pi$ consists of m objectives; $D \subseteq R^n$ and $\Pi \subseteq R^m$ denote the decision space and the objective space, respectively. The objectives in Equation (1) are often in conflict with each other. That is, the improvement of one objective may lead to the deterioration of another. This gives rise to a set of optimal solutions (largely known as Pareto-optimal solutions), instead of a single optimal solution. The concept of Pareto-optimal is defined as follows.

Definition 1. *Given two decision vectors $\mathbf{u}, \mathbf{v}$ and their corresponding objective vectors $F(\mathbf{u}), F(\mathbf{v})$. $\mathbf{u}$ dominates $\mathbf{v}$ (denoted as $\mathbf{u} \prec \mathbf{v}$), if and only if $\forall i \in \{1, \cdots, m\}, f_i(\mathbf{u}) \leq f_i(\mathbf{v})$ and $\exists i \in \{1, \cdots, m\}, f_i(\mathbf{u}) < f_i(\mathbf{v})$.*

Definition 2. *A solution $\mathbf{x}^*$ is Pareto-optimal if and only if there exists no $\mathbf{u} \in \Omega$ such that $\mathbf{u} \prec \mathbf{x}^*$. The set of all Pareto-optimal solutions is called the Pareto set. The corresponding objective vector set of the Pareto set is called the Pareto front.*

There exists a large body of benchmark sets for MOPs. Among them, Zizler–Deb–Thiele (ZDT) [32], Deb–Thiele–Laumanns–Zizler (DTLZ) [33], walking-fish-group (WFG) [34], and CEC 2009 competitions (UF) [35] are commonly used in the literature. These problems are characterized by multimodality, co-replicated Pareto sets and Pareto fronts, separability between decision variables, and correlation between decision variables and objectives, deception, and epistasis [34]. These are of varying difficulty and complexity and cover a variety of real-world problems. Moreover, the commonly used metrics for evaluating the solutions on MOPs include Inverted Generational Distance (IGD) [36] and Hypervolume (HV) [37,38].

2.2. Multi-Objective Evolutionary Algorithms

There is a large body of MOEAs, such as NSGA-II [4], MOEA/D [5], SPEA2 [8], PAES [10], and EMOCA [39]. This article will mainly introduce Genetic Algorithms (GAs) [4,5] and Differential Evolution (DE) Algorithms [40], and propose our algorithm based on these. GA is based on biological inspiration and generates new offspring by imitating the crossover and mutation process in biological genetic evolution. One of the most classic and widely used variation strategies is composed of a simulated binary crossover (SBX) operator and a polynomial mutation (PM) operator [41]. Specifically, supposing the two parent individuals are $\mathbf{x}^1 = (x_1^1, x_2^1, \cdots, x_n^1)$ and $\mathbf{x}^2 = (x_1^2, x_2^2, \cdots, x_n^2)$, and the generated offspring are $\mathbf{c}^1 = (c_1^1, c_2^1, \cdots, c_n^1)$ and $\mathbf{c}^2 = (c_1^2, c_2^2, \cdots, c_n^2)$. SBX and PM operate as follows.

SBX:

$$
\begin{aligned}
c_i^1 &= 0.5 \times [(1 + \beta) \cdot x_i^1 + (1 - \beta) \cdot x_i^2] \\
c_i^2 &= 0.5 \times [(1 - \beta) \cdot x_i^1 + (1 + \beta) \cdot x_i^2]'
\end{aligned}
\tag{2}
$$

where β is a parameter, defined as follows:

$$\beta = \begin{cases} (r \times 2)^{1/(1+\eta)} & \text{if } r \leq 0.5 \\ (1/(2 - r \times 2))^{1/(1+\eta)} & \text{otherwise} \end{cases}, \tag{3}$$

where r is a random value within $[0, 1]$ and η is a parameter representing the similarity between the offspring individual and the parent individual. The larger the value, the higher the similarity.

PM:

$$c_i^1 = x_i^1 + \Delta \cdot (u_i - l_i), \tag{4}$$

Δ is defined as follows:

$$\Delta = \begin{cases} (2r + (1 - 2r)\delta_1^{\eta+1})^{1/(\eta+1)-1} - 1 & \text{if } r \leq 0.5 \\ ((4r - 1) + (2r - 1)\delta_2^{\eta+1})^{1/(\eta+1)} & \text{otherwise} \end{cases}, \tag{5}$$

where $\delta_1 = (u_i - x_i^1)/(u_i - l_i)$ and $\delta_2 = (x_i^1 - l_i)/(u_i - l_i)$, r is a random parameter within $[0, 1]$, u_i and l_i are the upper and lower bounds of the i-th dimension variable, respectively. η is a parameter that representing the similarity between the offspring individual and the parent individual. The larger the value, the higher the similarity.

rand/p:

$$c_j^i = \begin{cases} x_j^{r_3} + F \cdot \sum_{l=1}^{p}(x_j^{r_1,l} - x_j^{r_2,l}) & \text{if } r \leq CR \text{ or } j = j_r \\ x_j^i, & \text{otherwise} \end{cases}, \tag{6}$$

best/p:

$$c_j^i = \begin{cases} x_j^{best} + F \cdot \sum_{l=1}^{p}(x_j^{r_1,l} - x_j^{r_2,l}), & \text{if } r \leq CR \text{ or } j = j_r \\ x_j^i, & \text{otherwise} \end{cases}, \tag{7}$$

current-to-rand/p:

$$c_j^i = \begin{cases} x_j^i + K \cdot (x_j^{r_3} - x_j^i) + F \cdot \sum_{l=1}^{p}(x_j^{r_1,l} - x_j^{r_2,l}) & \text{if } r \leq CR \text{ or } j = j_r \\ x_j^i & \text{otherwise} \end{cases}, \tag{8}$$

current-to-best/p:

$$c_j^i = \begin{cases} x_j^i + K \cdot (x_j^{best} - x_j^i) + F \cdot \sum_{l=1}^{p}(x_j^{r_1,l} - x_j^{r_2,l}), & \text{if } r \leq CR \text{ or } j = j_r \\ x_j^i, & \text{otherwise} \end{cases}. \tag{9}$$

Different from GAs, DEs mainly perform gradient estimation through differential mutation operators to generate offspring. Over the past few decades, various types of differential mutation operators have been proposed. Mezura-Montes et al. [40] summarized them in Equations (6)–(9):

In Equations (6)–(9), p is the number of pairs of solutions used to compute the differences in the mutation operator, c_j^i is the j-th dimension variable of the i-th offspring, $x_j^{r_3}$ is the j-th dimension variable of the donor solution chosen at random, x_j^{best} is the j-th dimension variable of the best solution in the population as the donor solution, x_j^i is the j-th dimension variable of the current parent, $x_j^{r_1,l}$ and $x_j^{r_2,l}$ are the j-th dimension variables of the p-th pair to compute the mutation differential, and F, K, and CR are parameters in the DE algorithm.

2.3. Multi-Operator-Based MOEAs

Multi-Operator MOEAs mainly adaptively allocate the computation resources (population sizes) among different operators. The allocation strategy could be designed by human experts [11,12] or learned automatically [13,14]. Specifically, motivated by the well-known ensemble learning approach AdaBoost [42], Wang et al. [11] proposed to use the accumulated data of previous generations to adjust the population size of each operator in the next generation, such that the operators that produce better solutions in the previous generation will occupy more computational resources. Gao et al. [12] took the solutions generated by multiple operators as a whole and determined the local search strategy, global search direction, and final convergence direction by analyzing the features in the entire target space. Elsayed et al. [13] constructed a solver that includes two complementary algorithms. The inferior algorithm optimizes itself by learning the strategy of the superior algorithm and gives feedback to finetune the superior algorithm. Finally, the desired result is output from the high-quality algorithm. Sun et al. [14] regarded the parameter adaptation process as a Markov Decision Process (MDP). Through reinforcement learning, different parameter controllers are learned for different problems, such that they have different adaptive parameter selection methods for these problems. All of the above methods need a well-designed adaptive method to allocate computational resources to the member algorithms.

2.4. PAPs and Automatic Construction of PAPs

Essentially, a PAP [27,31,43] is a set of member algorithms that run in parallel:

$$P = \{\theta_1, \theta_2, \cdots, \theta_k\}, \tag{10}$$

where P is the PAP solver, θ_i is the i-th member algorithm of P, and k is the number of member algorithms.

Let $Metric(\theta, z)$ denote the performance of an algorithm θ on a problem z (e.g., an MOP) in terms of the performance metric $Metric$ (e.g., HV for MOPs). Then, the performance of P on a problem z, denoted as $\Omega(P, z)$, is the best performance achieved among the member algorithms of P on z:

$$\Omega(P, z) = \max_{\theta \in P} Metric(\theta, z). \tag{11}$$

Without loss of generality, we assume for M, the larger the better. Let Z denote a set of problems. The performance of P on Z, denoted as $\Omega(P, Z)$, is an aggregated value of the performance of P on all the problems in Z:

$$\Omega(P, Z) = \frac{1}{|Z|} \cdot \sum_{z \in Z} \Omega(P, z). \tag{12}$$

The conventional way to construct P is to manually choose member algorithms for it, as shown in [20,21], which is, however, non-trivial and tedious. To address this issue, Tang and Liu [25,26] proposed the framework of the automatic construction of PAPs. Specifically, given a performance metric $Metric$, a training set Z, and an algorithm configuration space Θ, the goal is to find at most k algorithms from Θ to form a PAP P^*, which has the best performance on Z in terms of M:

$$P^* = \arg\max_{P \subseteq \Theta \& |P| \le k} \Omega(P, Z), \tag{13}$$

where $\Omega(P, Z)$ is defined in Equation (12). The results in [25–27,31] have shown that the PAPs automatically built by solving the problem in Equation (13) could achieve excellent performance on problems including SAT, TSPs, and VRPs, which are even better than the algorithms designed by human experts.

3. MOEAs/PAP for MOPs

As stated in Equation (11), a PAP would return the best solution among the ones found by its member algorithms. However, this could be problematic in multi-objective optimization since an MOEA outputs a set of Pareto-optimal solutions, instead of a single one, for a given MOP. One way to mitigate this issue is to compare the solution sets found by the member algorithms in terms of the performance metric (e.g., HV), and finally returns the best solution set.

On the other hand, it is likely that the solution sets found by member algorithms contain solutions that do not dominate each other. Thus, we propose a novel procedure called Restructure to obtain the final solution set. Specifically, when all member algorithms have completed their execution, the Pareto-optimal solution sets obtained by them are first combined. Assuming each member algorithm has a population size of P, the size of this set is $N \cdot P$ (N is the number of member algorithms). Then, Restructure employs the non-dominated sorting mechanism of NSGA-II [4] on these solutions and finally selects the top-P solutions to form the output solution set. Finally, all the solution sets found by member algorithms, as well as the one generated by Restructure, are compared, and the best set is returned as the output of MOEAs/PAP, as illustrated in Figure 1.

Formally, for an MOEAs/PAP, denoted as P, its performance on an MOP z can be described as follows:

$$\Omega(P, z) = \max\{\max_{\theta \in P} Metric(\theta, z), Metric(\bar{\theta}, z)\}, \tag{14}$$

where $\bar{\theta}$ represents the Restructure procedure. Note Equation (14) is slightly different from Equation (11) (i.e., the performance of conventional PAPs) due to the Restructure procedure.
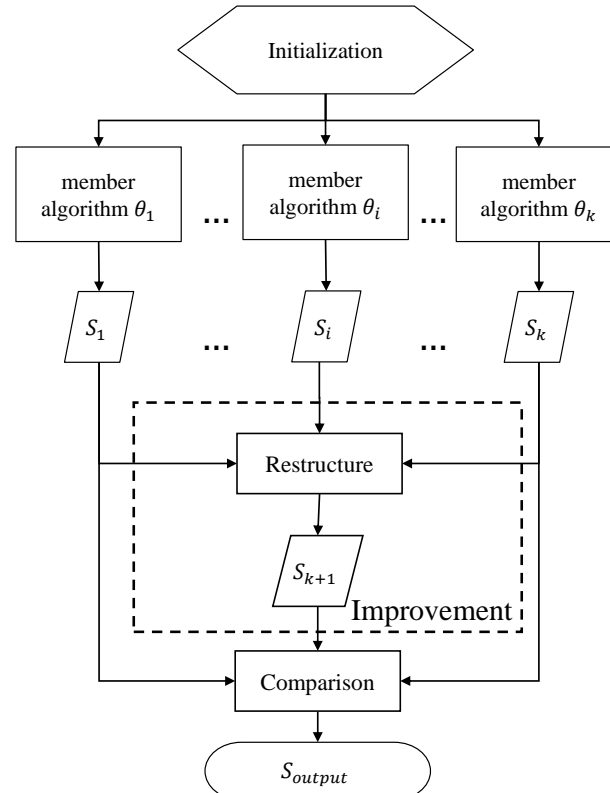

Figure 1. Illustrations of MOEAs/PAP, the variant form of PAP for MOPs. S_i represents the solution set found by member algorithm θ_i, and S_{output} represents the solution set finally returned by MOEAs/PAP. The main difference of MOEAs/PAP from conventional PAPs lies in the way of determining the final output, as indicated by the dash box.

4. Automatic Construction of MOEAs/PAP

As aforementioned, the member algorithms of a PAP could be automatically determined by solving the problem defined in Equation (13). Below, we first introduce the algorithm configuration space Θ, the training set Z, and the performance metric *Metric*; then we present the automatic construction approach for MOEAs/PAP.

4.1. Algorithm Configuration Space Θ and Training Set Z

The algorithm configuration space Θ is defined by a number of parameterized algorithms $B_1, B_2, \ldots, B_c$ (called foundation algorithms). Each foundation algorithm B_i has a number of parameters, whose values control the behavior of B_i and thus can largely affect its performance. Therefore, B_i taking different parameter values can actually be considered as different algorithms. Let Θ_i denote the set of all unique algorithms obtained by taking all possible values of the parameters of B_i. Then, the algorithm configuration space Θ is the union of $\Theta_1, \Theta_2, \ldots, \Theta_c$, i.e., $\Theta = \Theta_1 \cup \Theta_2 \cup \ldots \cup \Theta_c$. In this work, the foundation algorithms are parameterized MOEAs, i.e., NSGA-II with different evolutionary operators. The training set Z is a set of problems (MOPs), and should be representative of the target problems to which the constructed MOEAs/PAP is expected to be applied. The details of the algorithm configuration space Θ and the training set Z are given in the experiments (see Sections 5.1 and 5.2).

4.2. Performance Metric

As stated in Equation (12), during the construction process, the performance of the PAP on the training set Z is an aggregated value of its performance on all the problems in Z. However, for MOPs, the commonly used IGD and HV cannot be directly used as the performance metric *Metric* here. The main reason is that, for these two metrics, the performance of an MOEA on different MOPs cannot be directly aggregated (or compared) due to the different scales.

Hence, we propose to re-scale HV values within $[0, 1]$, by using the following HV Ratio (HVR):

$$hvr = \frac{hv}{hv^*}, \tag{15}$$

where hv is the HV value of the solution set found by an MOEA on an MOP, and hv^* is the HV value of the optimal solution set of the problem. Both hv and hv^* are calculated based on the same reference point in the objective space, which is obtained by taking the maximum on each objective achieved among the solution set found by the MOEA. However, there is still an issue in the actual use of HVR. During the construction process of MOEAs/PAP, there is a large body of MOEAs, which would be evaluated. Since the reference point is up to the solution set found by the evaluated MOEA, given an MOP, hv^* needs to be recalculated for every evaluated MOEA. Considering the size of the optimal solution set is usually very large (typically larger than 1000), thus in the construction process it requires a large amount of calculations for hv^*.

To address the above issue, we propose to use a fixed reference point for a given MOP, such that hv^* needs to be calculated for the problem for only once. One way is to use the the reference point taking the upper bounds (note that we assume MOPs take the minimization form) of the objective values. However, this will cause the gap between hv and hv^* to be very small, in which case hvr tends to be 1 and loses its evaluation ability. Therefore, we calculate the HVs of the undominated space for the found solution set and the optimal solution set, and finally use the ratio of them to assess the performance of an MOEA on a MOP.

The new performance metric is named Inverted Hypervolume Ratio (IHVR):

$$ihvr = \frac{hv_{all} - hv^*}{hv_{all} - hv}, \tag{16}$$

where hv and hv^* are calculated based on the reference point taking the upper bounds of the objective values, and hv_{all} is the HV of the cube formed by the ranges of objective values of the MOP. Thus, $hv_{all} - hv$ and $hv_{all} - hv^*$ are the HVs of the undominated space for the solution set found by the evaluated MOEA and the optimal solution set, respectively. Note that $ihvr \in (0, 1]$, and a larger value of $ihvr$ is better. IHVR can distinguish well between MOEAs with a different performance, and can also save a lot of calculations. In this work, IHVR is used as the performance metric *Metric*.

4.3. Automatic Construction Approach

Algorithm 1 presents the automatic construction approach for MOEAs/PAP. Starting from an empty set (line 2), the approach constructs the PAP (denoted as P) iteratively. Specifically, each iteration of the approach (lines 3–15) consists of two subsequent phases. In the first phase, an existing automatic algorithm configuration tool, namely SMAC 3 [44], is used to search in Θ to find the algorithm that can improve the performance of the current PAP to the largest extent (line 5), and then this algorithm is inserted into P (line 6). This phase is similar to the commonly-used greedy approach in the automatic construction of PAPs [27,31]. Additionally, we introduce a new phase, namely simplification, as the second phase in Algorithm 1. In this phase (lines 7–14), P would be simplified by removing the member algorithms that do not contribute at all to its performance (meaning removing these algorithms has no effect on the performance of P on the training set). Considering the size of P is bounded (line 3), removing the redundant algorithms from P is meaningful because this will leave space for new member algorithms that can improve the performance of P. Finally, the approach would be terminated if the maximum number of member algorithms is reached (line 3), or the performance of P does not improve with the inclusion of the found algorithm (line 6).

Algorithm 1: Automatic construction of MOEAs/PAP.

input : Algorithm configuration space Θ, training set Z, maximum number of member algorithms k

output: the final PAP P

1 /* ------Initialization------ */
2 $P \leftarrow \varnothing$;
3 **while** $|P| < k$ **do**
4 　/* ------Search in Θ------ */
5 　$\theta \leftarrow$ Find θ from Θ that maximizes $\Omega(P \cup \theta, Z)$;
6 　**if** $\Omega(P \cup \theta, Z) \leq \Omega(P, Z)$ **then** break **else** $P \leftarrow P \cup \{\theta\}$;
7 　/* -----Simplification of P--- */
8 　**while** *true* **do**
9 　　$P_{tmp} \leftarrow P$;
10 　　**for** $\theta \in P$ **do**
11 　　　**if** $\Omega(P \setminus \{\theta\}, Z) \geq \Omega(P, Z)$ **then** $P \leftarrow P \setminus \{\theta\}$;
12 　　**end**
13 　　**if** $P_{tmp} = P$ **then** break;
14 　**end**
15 **end**
16 **return** P

---

5. Experiments

In the experiments, we first used the proposed automatic construction approach to build a MOEAs/PAP with a training set, and then compared it with other MOEAs on the testing set to verify its performance. Finally, we analyzed the performance of each member algorithm of the MOEAs/PAP, as well as the impact of the Restructure procedure.

5.1. Benchmark Sets

We collected the commonly-used MOP benchmarks in the literature, including ZDT [32], DTLZ [33], WFG [34], and UF [35]. These benchmark sets contain 32 MOPs in total, as summarized in Table 1. From each of these benchmark sets, several problems were chosen as the training problems, and the remaining problems were used for testing. Specifically, the training set includes 17 MOPs in total, i.e., ZDT3-6, DTLZ2-5, WFG1, WFG5-6, WFG9, UF3-4, UF6-7, and UF10; the testing set contains 15 MOPs in total, i.e., ZDT1-2, DTLZ1, DTLZ6-7, WFG2-4, WFG7-8, UF1-2, UF5, and UF8-9.

Table 1. The problem benchmark sets used in the experiments.

Problem	Decision Vector Dimension (n)	Objective Vector Dimension (m)
ZDT1-3	30	2
ZDT4	10	2
ZDT5	11	2
ZDT6	10	2
DTLZ1-7	11	2
WFG1-9	12	3
UF1-7	30	2
UF8-10	30	3

5.2. Construction of NSGA-II/PAP

To use the approach in Algorithm 1 to construct an MOEAs/PAP, one needs to provide a training set of MOPs (detailed in Section 5.1) and an algorithm configuration space. Here, the algorithm configuration space is defined based on five variants of NSGA-II [4], each of which adopts a different parameterized evolutionary operator as defined in Equations (2)–(9). All the foundation algorithms, as well as their evolutionary operators and parameters, are summarized in Table 2. Finally, the maximum number of member algorithms, i.e., k, was set to 10, considering that 10-core machines are widely available now.

Table 2. The five foundation algorithms (variants of NSGA-II), as well as their evolutionary operators and parameters, that define the algorithm configuration space. For brevity, each foundation algorithm is denoted by the used evolutionary operator. n represents the decision vector dimension of the problem.

Foundation Algorithm	Parameter	Value Range
SBX + PM	η of SBX	$\{1, 2, \cdots, 100\}$
	p_c	$\{1\}$
	η of PM	$\{1, 2, \cdots, 100\}$
	p_m	$\{1/n\}$
rand/p	F	$(0, 2]$
	p	$\{1, 2\}$
	CR	$(0, 1]$
best/p	F	$(0, 2]$
	p	$\{1, 2\}$
	CR	$(0, 1]$
current-to-rand/p	F	$(0, 2]$
	K	$(0, 1]$
	p	$\{1\}$
	CR	$(0, 1]$
current-to-best/p	F	$(0, 2]$
	K	$(0, 1]$
	p	$\{1\}$
	CR	$(0, 1]$

Given the training set and the algorithm configuration space, we used the proposed approach to construct an MOEAs/PAP. Since all the foundation algorithms are variants of NSGA-II, the constructed MOEAs/PAP is named NSGA-II/PAP. Specifically, NSGA-II/PAP contains six member algorithms, which are detailed in Table 3.

Table 3. Member algorithms of the constructed NSGA-II/PAP. For brevity, each member algorithm is denoted by the used evolutionary operator.

Member Algorithm	Parameter
SBX + PM	η of SBX = 81, η of PM = 18 $p_c = 1, p_m = 1/n$
best/p	$p = 2, F = 1.16198$ $CR = 0.07724$
rand/p	$p = 1, F = 0.07578$ $CR = 0.41910$
best/p	$p = 2, F = 0.61077$ $CR = 0.57930$
SBX + PM	η of SBX = 67, η of PM = 54 $p_c = 1, p_m = 1/n$
SBX + PM	η of SBX = 45, η of PM = 23 $p_c = 1, p_m = 1/n$

5.3. Compared Algorithms and Experimental Protocol

We compare NSGA-II/PAP with the NSGA-II-based MOEA including the original NSGA-II [4], its variant using the DE operator, namely NSGA-II-DE [18], and the state-of-the-art multi-operator-based MOEAs, dubbed NSGA-II/MOE [11]. Also, we included other types of MOEAs, MOPSO [9] and MOEA/D [5], in the comparison. The experimental results in [11] have shown that NSGA-II/MOE could consistently achieve a better performance than conventional MOEAs on a large body of benchmark sets.

For the compared algorithms, the recommended parameter settings in the original publication were used, which are listed in Table 4. For all the tested algorithms, the population sizes and the number of fitness evaluations (FEs) were kept the same. Specifically, for UF1-7, population size was set to 100 and the max generation number was set to 500; for UF8-10, population size was set to 500 and the max generation number was set to 600; for WFG1-9, population size was set to 500 and the max generation number was set to 250; for all DTLZ and ZDT, population size was set to 100 and the max generation number was set to 250.

Table 4. Parameter settings of the compared algorithms.

Algorithm	Operator	Parameter
NSGA-II	SBX + PM	η of SBX = 20, η of PM = 20 $p_m = 1/n$
NSGA-II-DE	rand/1	$F = 0.5, CR = 0.5$
NSGA-II/MOE	rand/1	$F = 0.5, CR = 1.0$
	rand/2	$F = 0.5, CR = 1.0$
	current-to-rand/1	$F = 0.5, CR = 1.0, K = 0.5$
	SBX + PM	η of SBX = 20, η of PM = 20 $p_m = 1/n$
MOEA/D	SBX + PM	η of SBX = 20, η of PM = 20 $p_m = 1/n, T = 10$
MOEA/D	-	$C1 = 20, C2 = 20$ $p_m = 1/n, W = 0.4$

Since NSGA-II/PAP runs its member algorithms in parallel, it would perform more FEs than the compared algorithms. Hence, in the experiments, for each compared algorithm, we also considered two variants of it that performed the same number of FEs as NSGA-II/PAP. Let N be the number of member algorithms of NSGA-II/PAP (here, $N = 6$). Specifically, the first variant, indicated by Ngen, increased the maximum generation number of the algorithm to N times the original number. The second variant, indicated by Nsize, increased the population size to N times the original population size, and used the Restructure procedure (see Section 3) to choose the final Pareto-optimal solution set of the original population size from the final population. We implemented all the tested algorithms with Python (version 3.8) library Geatpy (version 2.6.0 (https://github.com/geatpy-dev/geatpy) (accessed on 31 October 2022)). All the experiments were conducted on an Intel Xeon machine with 96 GB RAM and 16 cores (2.10 GHz, 20 MB Cache), running Ubuntu 16.04.

For each testing problem, each tested algorithm would be independently applied for 30 times; the average and standard deviation of the achieved HV and IGD would be reported.

5.4. Testing Results and Analysis

The testing results in terms of HV are presented in Tables 5 and 6. And the results in terms of IGD are presented in Tables 7 and 8. For each problem, the best performance is indicated in bold, and a Wilcoxon's bilateral rank-sum test was conducted to judge whether the difference between the performance of the compared algorithm and the performance of NSGA-II/PAP was significant (with $p = 0.05$).

Table 5. The mean $\pm$ standard deviation of the HV values across the 30 runs over the testing set. For each testing problem, the best performance is indicated in bold (note for HV, the larger, the better). † indicates that the performance of the algorithm is significantly different from the performance of NSGA-II/PAP (according to a Wilcoxon's bilateral rank sum test with $p = 0.05$). The significance test of NSGA-II/PAP against the compared algorithm is summarized in the win-draw-loss (W-D-L) counts.

Problem	NSGA-II/PAP	NSGA-II (Nsize)	NSGA-II (Ngen)	NSGA-II-DE (Nsize)	NSGA-II-DE (Ngen)	NSGA-II/MOE (Nsize)	NSGA-II/MOE (Ngen)
UF1	$0.6770 \pm 4.50 \times 10^{-4}$	$0.6147 \pm 1.01 \times 10^{-4}$ †	$0.6173 \pm 3.64 \times 10^{-5}$ †	$0.6283 \pm 2.44 \times 10^{-5}$ †	$0.6469 \pm 1.83 \times 10^{-4}$ †	$0.6574 \pm 2.11 \times 10^{-4}$ †	$\mathbf{0.6920 \pm 5.54 \times 10^{-4}}$ †
UF2	$0.6957 \pm 1.09 \times 10^{-5}$	$0.6814 \pm 3.80 \times 10^{-5}$ †	$0.6889 \pm 5.04 \times 10^{-5}$ †	$0.6749 \pm 8.27 \times 10^{-5}$ †	$0.6947 \pm 2.74 \times 10^{-5}$	$0.6814 \pm 1.68 \times 10^{-4}$ †	$\mathbf{0.7003 \pm 2.90 \times 10^{-5}}$ †
UF5	$\mathbf{0.2503 \pm 2.17 \times 10^{-3}}$	$0.1944 \pm 7.70 \times 10^{-4}$ †	$0.1755 \pm 2.08 \times 10^{-3}$ †	$0.2129 \pm 1.20 \times 10^{-13}$ †	$0.1979 \pm 1.32 \times 10^{-3}$ †	$0.2041 \pm 1.45 \times 10^{-3}$ †	$0.1576 \pm 4.19 \times 10^{-3}$ †
UF8	$\mathbf{0.4394 \pm 1.08 \times 10^{-3}}$	$0.2827 \pm 1.67 \times 10^{-4}$ †	$0.3142 \pm 9.92 \times 10^{-4}$ †	$0.2992 \pm 1.19 \times 10^{-4}$ †	$0.2794 \pm 1.34 \times 10^{-4}$ †	$0.4157 \pm 1.19 \times 10^{-3}$ †	$0.4252 \pm 4.08 \times 10^{-3}$
UF9	$\mathbf{0.7619 \pm 1.92 \times 10^{-4}}$	$0.7014 \pm 9.52 \times 10^{-4}$ †	$0.6452 \pm 5.52 \times 10^{-3}$ †	$0.7254 \pm 1.92 \times 10^{-7}$ †	$0.7127 \pm 5.61 \times 10^{-6}$ †	$0.5276 \pm 1.31 \times 10^{-2}$ †	$0.4190 \pm 1.59 \times 10^{-2}$ †
WFG2	$\mathbf{0.9425 \pm 1.13 \times 10^{-7}}$	$0.9393 \pm 6.09 \times 10^{-7}$ †	$0.9410 \pm 1.17 \times 10^{-7}$ †	$0.9390 \pm 4.45 \times 10^{-7}$ †	$0.9392 \pm 2.12 \times 10^{-7}$ †	$0.9406 \pm 3.33 \times 10^{-7}$ †	$0.9408 \pm 1.49 \times 10^{-7}$ †
WFG3	$\mathbf{0.6539 \pm 5.95 \times 10^{-6}}$	$0.6439 \pm 6.25 \times 10^{-6}$ †	$0.6395 \pm 8.55 \times 10^{-6}$ †	$0.6332 \pm 4.57 \times 10^{-6}$ †	$0.6119 \pm 1.58 \times 10^{-5}$ †	$0.6439 \pm 3.06 \times 10^{-6}$ †	$0.6350 \pm 1.05 \times 10^{-5}$ †
WFG4	$\mathbf{0.5745 \pm 1.43 \times 10^{-6}}$	$0.5714 \pm 6.99 \times 10^{-7}$ †	$0.5651 \pm 3.08 \times 10^{-6}$ †	$0.5490 \pm 1.73 \times 10^{-6}$ †	$0.5372 \pm 3.02 \times 10^{-6}$ †	$0.5687 \pm 1.10 \times 10^{-6}$ †	$0.5583 \pm 4.28 \times 10^{-6}$ †
WFG7	$0.1077 \pm 3.61 \times 10^{-6}$	$0.1027 \pm 2.88 \times 10^{-6}$ †	$\mathbf{0.1114 \pm 4.64 \times 10^{-7}}$ †	$0.0276 \pm 2.07 \times 10^{-5}$ †	$0.0731 \pm 3.85 \times 10^{-5}$ †	$0.1025 \pm 2.66 \times 10^{-6}$ †	$0.1038 \pm 8.77 \times 10^{-6}$ †
WFG8	$0.2121 \pm 5.88 \times 10^{-6}$	$0.2261 \pm 4.72 \times 10^{-7}$ †	$\mathbf{0.2450 \pm 2.21 \times 10^{-7}}$ †	$0.0775 \pm 1.92 \times 10^{-5}$ †	$0.1360 \pm 9.81 \times 10^{-5}$ †	$0.2172 \pm 5.12 \times 10^{-7}$ †	$0.2378 \pm 1.02 \times 10^{-5}$ †
DTLZ1	$0.5556 \pm 5.79 \times 10^{-4}$	$0.5773 \pm 9.67 \times 10^{-7}$ †	$0.5812 \pm 1.36 \times 10^{-7}$ †	$0.0000 \pm 0.00 \times 10^{0}$ †	$\mathbf{0.5816 \pm 3.20 \times 10^{-8}}$ †	$0.5581 \pm 1.02 \times 10^{-2}$ †	$0.4668 \pm 5.18 \times 10^{-2}$ †
DTLZ6	$0.3465 \pm 2.52 \times 10^{-8}$	$0.3451 \pm 2.09 \times 10^{-7}$ †	$0.3462 \pm 6.18 \times 10^{-8}$ †	$0.3451 \pm 2.74 \times 10^{-7}$ †	$0.3463 \pm 5.14 \times 10^{-8}$ †	$0.3448 \pm 3.00 \times 10^{-7}$ †	$\mathbf{0.3468 \pm 6.28 \times 10^{-8}}$ †
DTLZ7	$\mathbf{0.2428 \pm 1.23 \times 10^{-9}}$	$0.2420 \pm 4.04 \times 10^{-8}$ †	$0.2405 \pm 1.43 \times 10^{-4}$ †	$0.2420 \pm 6.57 \times 10^{-8}$ †	$0.2425 \pm 2.94 \times 10^{-9}$ †	$0.2418 \pm 8.32 \times 10^{-8}$ †	$0.2383 \pm 2.77 \times 10^{-4}$ †
ZDT1	$\mathbf{0.7198 \pm 3.84 \times 10^{-8}}$	$0.7164 \pm 1.02 \times 10^{-6}$ †	$0.7195 \pm 5.31 \times 10^{-8}$ †	$0.7164 \pm 6.66 \times 10^{-7}$ †	$0.7198 \pm 2.83 \times 10^{-8}$	$0.7152 \pm 1.61 \times 10^{-6}$ †	$0.7188 \pm 1.70 \times 10^{-7}$ †
ZDT2	$0.4442 \pm 3.29 \times 10^{-8}$	$0.4414 \pm 3.66 \times 10^{-7}$ †	$0.4436 \pm 6.48 \times 10^{-8}$ †	$0.4409 \pm 6.24 \times 10^{-7}$ †	$\mathbf{0.4444 \pm 1.52 \times 10^{-8}}$ †	$0.4403 \pm 1.19 \times 10^{-6}$ †	$0.4438 \pm 1.27 \times 10^{-7}$ †
W-D-L	-	13-0-2	12-0-3	15-0-0	11-2-2	13-0-2	10-1-4

Table 6. The mean ± standard deviation of the HV values across the 30 runs over the testing set. For each testing problem, the best performance is indicated in bold (note for HV, the larger, the better). † indicates that the performance of the algorithm is significantly different from the performance of NSGA-II/PAP (according to a Wilcoxon's bilateral rank sum test with $p = 0.05$). The significance test of NSGA-II/PAP against the compared algorithm is summarized in the win-draw-loss (W-D-L) counts.

Problem	NSGA-II/PAP	MOEA/D (Ngen)	MOPSO (Ngen)	MOEA/D (Nsize)	MOPSO (Nsize)
UF1	$0.6770 \pm 4.50 \times 10^{-4}$	$0.6662 \pm 4.52 \times 10^{-4}$ †	$0.6708 \pm 5.96 \times 10^{-4}$ †	$\mathbf{0.6873 \pm 1.91 \times 10^{-4}}$ †	$0.6784 \pm 6.74 \times 10^{-4}$
UF2	$0.6957 \pm 1.09 \times 10^{-5}$	$0.6972 \pm 1.30 \times 10^{-4}$	$0.6997 \pm 3.13 \times 10^{-6}$ †	$\mathbf{0.7130 \pm 8.17 \times 10^{-5}}$ †	$0.7111 \pm 9.78 \times 10^{-7}$ †
UF5	$\mathbf{0.2503 \pm 2.17 \times 10^{-3}}$	$0.1610 \pm 4.70 \times 10^{-3}$ †	$0.1088 \pm 4.64 \times 10^{-3}$ †	$0.2223 \pm 2.45 \times 10^{-3}$ †	$0.0900 \pm 2.69 \times 10^{-3}$ †
UF8	$0.4394 \pm 1.08 \times 10^{-3}$	$0.4545 \pm 2.75 \times 10^{-3}$ †	$0.4602 \pm 2.97 \times 10^{-3}$ †	$0.4847 \pm 9.02 \times 10^{-4}$ †	$\mathbf{0.5074 \pm 7.02 \times 10^{-4}}$ †
UF9	$\mathbf{0.7619 \pm 1.92 \times 10^{-4}}$	$0.7504 \pm 2.65 \times 10^{-3}$ †	$0.5970 \pm 1.17 \times 10^{-2}$ †	$0.7554 \pm 2.07 \times 10^{-4}$ †	$0.7416 \pm 3.71 \times 10^{-3}$ †
WFG2	$\mathbf{0.9425 \pm 1.13 \times 10^{-7}}$	$0.9364 \pm 1.86 \times 10^{-6}$ †	$0.9287 \pm 4.76 \times 10^{-7}$ †	$0.9411 \pm 1.44 \times 10^{-7}$ †	$\mathbf{0.9427 \pm 4.49 \times 10^{-8}}$
WFG3	$\mathbf{0.6539 \pm 5.95 \times 10^{-6}}$	$0.6417 \pm 6.74 \times 10^{-6}$ †	$0.5362 \pm 1.05 \times 10^{-4}$ †	$0.6490 \pm 5.13 \times 10^{-7}$ †	$0.6290 \pm 1.57 \times 10^{-5}$ †
WFG4	$0.5745 \pm 1.43 \times 10^{-6}$	$0.5768 \pm 1.06 \times 10^{-6}$	$0.5046 \pm 1.02 \times 10^{-5}$ †	$\mathbf{0.5905 \pm 2.02 \times 10^{-7}}$ †	$0.5622 \pm 9.02 \times 10^{-7}$ †
WFG7	$\mathbf{0.2121 \pm 5.88 \times 10^{-6}}$	$0.0976 \pm 5.07 \times 10^{-7}$ †	$0.0106 \pm 1.37 \times 10^{-5}$ †	$0.1089 \pm 1.74 \times 10^{-7}$ †	$0.0160 \pm 1.51 \times 10^{-5}$ †
WFG8	$\mathbf{0.5556 \pm 5.79 \times 10^{-4}}$	$0.2373 \pm 2.12 \times 10^{-7}$ †	$0.0627 \pm 1.89 \times 10^{-7}$ †	$0.2384 \pm 1.84 \times 10^{-7}$ †	$0.0659 \pm 1.32 \times 10^{-7}$ †
DTLZ1	$0.5556 \pm 5.79 \times 10^{-4}$	$0.5787 \pm 6.39 \times 10^{-7}$ †	$0.0000 \pm 0.00 \times 10^{0}$ †	$\mathbf{0.5854 \pm 1.75 \times 10^{-8}}$ †	$0.0000 \pm 0.00 \times 10^{0}$ †
DTLZ6	$0.3465 \pm 2.52 \times 10^{-8}$	$0.3438 \pm 1.12 \times 10^{-6}$ †	$0.3460 \pm 2.34 \times 10^{-7}$	$0.3497 \pm 4.04 \times 10^{-8}$ †	$\mathbf{0.3498 \pm 2.74 \times 10^{-9}}$ †
DTLZ7	$\mathbf{0.2428 \pm 1.23 \times 10^{-9}}$	$0.2412 \pm 7.92 \times 10^{-7}$ †	$0.2419 \pm 1.61 \times 10^{-7}$ †	$\mathbf{0.2433 \pm 2.07 \times 10^{-8}}$ †	$\mathbf{0.2430 \pm 5.57 \times 10^{-10}}$ †
ZDT1	$\mathbf{0.7198 \pm 3.84 \times 10^{-8}}$	$0.7167 \pm 6.31 \times 10^{-7}$ †	$0.7149 \pm 6.57 \times 10^{-7}$ †	$0.7132 \pm 5.01 \times 10^{-8}$ †	$0.7129 \pm 7.33 \times 10^{-9}$ †
ZDT2	$\mathbf{0.4442 \pm 3.29 \times 10^{-8}}$	$0.4413 \pm 6.20 \times 10^{-7}$ †	$0.4422 \pm 4.95 \times 10^{-7}$ †	$0.4427 \pm 2.48 \times 10^{-8}$ †	$0.4435 \pm 6.31 \times 10^{-8}$ †
W-D-L	-	11-2-2	12-1-2	8-1-6	9-3-3

In summary, NSGA-II/PAP not only shows a superior performance to conventional single-operator-based MOEAs such as NSGA-II, but also outperforms the human-designed multi-operator-based NSGA-II/MOE. Moreover, the performance advantage of NSGA-II is still significant even when the compared algorithms consume the same number of FEs (indicated by Ngen and Nsize as NSGA-II/PAP. We further compared the average runtime of NSGA-II/PAP and the two variants of NSGA-II/MOE for solving the testing problems. In addition, we also compared the running time of MOEAs/PAP in parallel and serial cases. The results in terms of speed-up ratios are illustrated in Figure 2. It can be observed that the speed-up ratios achieved by running NSGA-II/PAP in parallel are worse than the ideal ratios, which are equal to the number of member algorithms. The reason for this is that the current naive parallel-running strategy in NSGA-II/PAP only runs member algorithms in parallel and does not parallelize the Restructure procedure. We note that parallelizing the Restructure procedure can further improve the speed-up ratio, which is an important direction for future research.

Overall, the above results indicate that PAPs are, in general, a powerful tool for solving computationally hard problems, and the automatic construction of PAPs has huge potential in developing powerful MOP solvers that can even rival human-designed state-of-the-art MOEAs.

Table 7. The mean $\pm$ standard deviation of the IGD values across the 30 runs over the testing set. For each testing problem, the best performance is indicated in bold (note for IGD, the smaller, the better). $\dagger$ indicates that the performance of the algorithm is significantly different from the performance of NSGA-II/PAP (according to a Wilcoxon's bilateral rank sum test with $p = 0.05$). The significance test of NSGA-II/PAP against the compared algorithm is summarized in the win-draw-loss (W-D-L) counts.

Problem	NSGA-II/PAP	NSGA-II (Nsize)	NSGA-II (Ngen)	NSGA-II-DE (Nsize)	NSGA-II-DE (Ngen)	NSGA-II/MOE (Nsize)	NSGA-II/MOE (Ngen)
UF1	$3.690 \times 10^{-2} \pm 2.88 \times 10^{-4}$	$8.938 \times 10^{-2} \pm 1.01 \times 10^{-4}$ $\dagger$	$8.911 \times 10^{-2} \pm 7.59 \times 10^{-4}$ $\dagger$	$7.501 \times 10^{-2} \pm 1.80 \times 10^{-5}$ $\dagger$	$5.993 \times 10^{-2} \pm 9.98 \times 10^{-5}$ $\dagger$	$4.614 \times 10^{-2} \pm 7.27 \times 10^{-5}$ $\dagger$	$\mathbf{2.234 \times 10^{-2} \pm 1.99 \times 10^{-4}}$ $\dagger$
UF2	$2.180 \times 10^{-2} \pm 8.39 \times 10^{-6}$	$3.202 \times 10^{-2} \pm 1.68 \times 10^{-5}$ $\dagger$	$3.086 \times 10^{-2} \pm 1.42 \times 10^{-4}$ $\dagger$	$3.694 \times 10^{-2} \pm 4.02 \times 10^{-5}$ $\dagger$	$2.525 \times 10^{-2} \pm 4.63 \times 10^{-5}$ $\dagger$	$3.771 \times 10^{-2} \pm 3.65 \times 10^{-4}$	$\mathbf{1.998 \times 10^{-2} \pm 3.84 \times 10^{-5}}$ $\dagger$
UF5	$\mathbf{2.677 \times 10^{-1} \pm 2.68 \times 10^{-3}}$	$2.978 \times 10^{-1} \pm 6.01 \times 10^{-4}$ $\dagger$	$3.481 \times 10^{-1} \pm 6.97 \times 10^{-3}$ $\dagger$	$3.384 \times 10^{-1} \pm 2.41 \times 10^{-11}$ $\dagger$	$3.439 \times 10^{-1} \pm 7.40 \times 10^{-4}$ $\dagger$	$3.090 \times 10^{-1} \pm 3.29 \times 10^{-3}$ $\dagger$	$3.772 \times 10^{-1} \pm 8.76 \times 10^{-3}$ $\dagger$
UF8	$\mathbf{1.142 \times 10^{-1} \pm 2.93 \times 10^{-4}}$	$2.408 \times 10^{-1} \pm 4.00 \times 10^{-4}$ $\dagger$	$1.900 \times 10^{-1} \pm 5.01 \times 10^{-4}$ $\dagger$	$1.959 \times 10^{-1} \pm 1.38 \times 10^{-4}$ $\dagger$	$2.037 \times 10^{-1} \pm 6.38 \times 10^{-5}$ $\dagger$	$1.733 \times 10^{-1} \pm 9.14 \times 10^{-4}$ $\dagger$	$1.530 \times 10^{-1} \pm 1.03 \times 10^{-3}$ $\dagger$
UF9	$\mathbf{6.188 \times 10^{-2} \pm 3.61 \times 10^{-5}}$	$1.024 \times 10^{-1} \pm 1.40 \times 10^{-3}$ $\dagger$	$1.465 \times 10^{-1} \pm 5.17 \times 10^{-3}$ $\dagger$	$7.863 \times 10^{-2} \pm 1.22 \times 10^{-6}$ $\dagger$	$8.271 \times 10^{-2} \pm 2.41 \times 10^{-6}$ $\dagger$	$2.498 \times 10^{-1} \pm 8.84 \times 10^{-3}$ $\dagger$	$3.536 \times 10^{-1} \pm 1.29 \times 10^{-2}$ $\dagger$
WFG2	$9.415 \times 10^{-2} \pm 5.97 \times 10^{-6}$	$1.035 \times 10^{-1} \pm 1.84 \times 10^{-5}$ $\dagger$	$9.365 \times 10^{-2} \pm 5.39 \times 10^{-6}$	$1.017 \times 10^{-1} \pm 1.36 \times 10^{-5}$ $\dagger$	$9.319 \times 10^{-2} \pm 4.90 \times 10^{-6}$	$9.496 \times 10^{-2} \pm 1.20 \times 10^{-5}$	$\mathbf{9.308 \times 10^{-2} \pm 8.05 \times 10^{-6}}$
WFG3	$\mathbf{8.358 \times 10^{-2} \pm 4.31 \times 10^{-5}}$	$8.584 \times 10^{-2} \pm 2.33 \times 10^{-5}$	$9.104 \times 10^{-2} \pm 3.87 \times 10^{-5}$ $\dagger$	$9.339 \times 10^{-2} \pm 5.40 \times 10^{-5}$ $\dagger$	$1.146 \times 10^{-1} \pm 2.33 \times 10^{-5}$ $\dagger$	$8.391 \times 10^{-2} \pm 4.48 \times 10^{-5}$	$8.785 \times 10^{-2} \pm 3.88 \times 10^{-5}$ $\dagger$
WFG4	$1.157 \times 10^{-1} \pm 5.75 \times 10^{-6}$	$\mathbf{1.146 \times 10^{-1} \pm 2.93 \times 10^{-6}}$	$1.242 \times 10^{-1} \pm 6.40 \times 10^{-6}$ $\dagger$	$1.347 \times 10^{-1} \pm 5.02 \times 10^{-6}$ $\dagger$	$1.669 \times 10^{-1} \pm 2.03 \times 10^{-5}$ $\dagger$	$1.156 \times 10^{-1} \pm 3.90 \times 10^{-6}$	$1.344 \times 10^{-1} \pm 4.64 \times 10^{-6}$ $\dagger$
WFG7	$1.147 \times 10^{0} \pm 6.21 \times 10^{-6}$	$1.156 \times 10^{0} \pm 6.93 \times 10^{-6}$ $\dagger$	$\mathbf{1.145 \times 10^{0} \pm 5.47 \times 10^{-6}}$ $\dagger$	$1.244 \times 10^{0} \pm 1.55 \times 10^{-4}$ $\dagger$	$1.176 \times 10^{0} \pm 6.13 \times 10^{-5}$ $\dagger$	$1.158 \times 10^{0} \pm 6.41 \times 10^{-6}$ $\dagger$	$1.157 \times 10^{0} \pm 6.12 \times 10^{-6}$ $\dagger$
WFG8	$2.216 \times 10^{0} \pm 5.05 \times 10^{-5}$	$2.182 \times 10^{0} \pm 5.23 \times 10^{-6}$ $\dagger$	$\mathbf{2.149 \times 10^{0} \pm 5.86 \times 10^{-7}}$ $\dagger$	$2.796 \times 10^{0} \pm 1.03 \times 10^{-3}$ $\dagger$	$2.475 \times 10^{0} \pm 1.80 \times 10^{-3}$ $\dagger$	$2.204 \times 10^{0} \pm 5.52 \times 10^{-6}$ $\dagger$	$2.159 \times 10^{0} \pm 3.14 \times 10^{-5}$ $\dagger$
DTLZ1	$1.083 \times 10^{-2} \pm 6.97 \times 10^{-5}$	$3.900 \times 10^{-3} \pm 1.69 \times 10^{-7}$ $\dagger$	$\mathbf{2.250 \times 10^{-3} \pm 5.65 \times 10^{-9}}$ $\dagger$	$4.432 \times 10 \pm 6.57 \times 10^{-1}$ $\dagger$	$2.194 \times 10^{-3} \pm 5.87 \times 10^{-9}$ $\dagger$	$1.581 \times 10^{-2} \pm 3.93 \times 10^{-3}$	$8.446 \times 10^{-2} \pm 3.04 \times 10^{-2}$ $\dagger$
DTLZ6	$5.596 \times 10^{-3} \pm 9.35 \times 10^{-8}$	$8.356 \times 10^{-3} \pm 4.43 \times 10^{-7}$ $\dagger$	$5.887 \times 10^{-3} \pm 4.79 \times 10^{-8}$ $\dagger$	$8.817 \times 10^{-3} \pm 1.28 \times 10^{-6}$ $\dagger$	$5.767 \times 10^{-3} \pm 5.98 \times 10^{-8}$ $\dagger$	$1.019 \times 10^{-2} \pm 2.26 \times 10^{-6}$ $\dagger$	$\mathbf{5.208 \times 10^{-3} \pm 4.25 \times 10^{-8}}$ $\dagger$
DTLZ7	$\mathbf{5.105 \times 10^{-3} \pm 2.05 \times 10^{-8}}$	$8.745 \times 10^{-3} \pm 7.24 \times 10^{-7}$ $\dagger$	$2.097 \times 10^{-2} \pm 6.83 \times 10^{-3}$ $\dagger$	$8.962 \times 10^{-3} \pm 1.33 \times 10^{-6}$ $\dagger$	$6.744 \times 10^{-3} \pm 6.79 \times 10^{-8}$ $\dagger$	$9.539 \times 10^{-3} \pm 1.39 \times 10^{-6}$ $\dagger$	$3.609 \times 10^{-2} \pm 1.32 \times 10^{-2}$ $\dagger$
ZDT1	$\mathbf{4.528 \times 10^{-3} \pm 3.32 \times 10^{-8}}$	$7.800 \times 10^{-3} \pm 1.10 \times 10^{-6}$ $\dagger$	$4.660 \times 10^{-3} \pm 3.30 \times 10^{-8}$ $\dagger$	$7.732 \times 10^{-3} \pm 1.08 \times 10^{-6}$ $\dagger$	$4.593 \times 10^{-3} \pm 2.87 \times 10^{-8}$	$8.253 \times 10^{-3} \pm 1.52 \times 10^{-6}$ $\dagger$	$5.014 \times 10^{-3} \pm 1.16 \times 10^{-7}$
ZDT2	$4.748 \times 10^{-3} \pm 5.31 \times 10^{-8}$	$7.711 \times 10^{-3} \pm 4.20 \times 10^{-7}$ $\dagger$	$5.386 \times 10^{-3} \pm 6.72 \times 10^{-8}$ $\dagger$	$7.955 \times 10^{-3} \pm 7.53 \times 10^{-7}$ $\dagger$	$\mathbf{4.668 \times 10^{-3} \pm 3.52 \times 10^{-8}}$	$9.236 \times 10^{-3} \pm 6.66 \times 10^{-6}$ $\dagger$	$5.077 \times 10^{-3} \pm 1.26 \times 10^{-7}$
W-D-L	-	11-2-2	11-1-3	15-0-0	12-3-0	11-4-0	10-1-4

Table 8. The mean ± standard deviation of the IGD values across the 30 runs over the testing set. For each testing problem, the best performance is indicated in bold (note for IGD, the smaller, the better). † indicates that the performance of the algorithm is significantly different from the performance of NSGA-II/PAP (according to a Wilcoxon's bilateral rank sum test with $p = 0.05$). The significance test of NSGA-II/PAP against the compared algorithm is summarized in the win-draw-loss (W-D-L) counts.

| Problem	NSGA-II/PAP	MOEA/D (Ngen)	MOPSO (Ngen)	MOEA/D (Nsize)	MOPSO (Nsize)
UF1	$3.690 \times 10^{-2} \pm 2.88 \times 10^{-4}$	$3.789 \times 10^{-2} \pm 2.31 \times 10^{-4}$ †	$3.584 \times 10^{-2} \pm 2.16 \times 10^{-4}$ †	$\mathbf{2.620 \times 10^{-2} \pm 1.63 \times 10^{-4}}$ †	$3.087 \times 10^{-2} \pm 2.47 \times 10^{-4}$ †
UF2	$2.180 \times 10^{-2} \pm 8.39 \times 10^{-6}$	$2.754 \times 10^{-2} \pm 3.01 \times 10^{-4}$ †	$2.790 \times 10^{-2} \pm 1.92 \times 10^{-6}$ †	$2.185 \times 10^{-2} \pm 2.01 \times 10^{-4}$	$\mathbf{2.088 \times 10^{-2} \pm 6.01 \times 10^{-7}}$ †
UF5	$\mathbf{2.677 \times 10^{-1} \pm 2.68 \times 10^{-3}}$	$4.470 \times 10^{-1} \pm 2.36 \times 10^{-2}$ †	$5.405 \times 10^{-1} \pm 3.96 \times 10^{-2}$ †	$2.821 \times 10^{-1} \pm 4.72 \times 10^{-3}$ †	$6.174 \times 10^{-1} \pm 7.04 \times 10^{-2}$ †
UF8	$\mathbf{1.142 \times 10^{-1} \pm 2.93 \times 10^{-4}}$	$\mathbf{1.142 \times 10^{-1} \pm 7.01 \times 10^{-4}}$	$1.149 \times 10^{-1} \pm 9.64 \times 10^{-4}$ †	$1.154 \times 10^{-1} \pm 3.17 \times 10^{-4}$ †	$1.147 \times 10^{-1} \pm 3.63 \times 10^{-4}$ †
UF9	$\mathbf{6.188 \times 10^{-2} \pm 3.61 \times 10^{-5}}$	$7.184 \times 10^{-2} \pm 3.67 \times 10^{-3}$ †	$2.070 \times 10^{-1} \pm 9.66 \times 10^{-3}$ †	$6.776 \times 10^{-2} \pm 1.08 \times 10^{-4}$ †	$8.757 \times 10^{-2} \pm 3.96 \times 10^{-3}$ †
WFG2	$9.415 \times 10^{-2} \pm 5.97 \times 10^{-6}$	$1.281 \times 10^{-1} \pm 3.42 \times 10^{-6}$ †	$9.076 \times 10^{-2} \pm 2.55 \times 10^{-6}$ †	$8.935 \times 10^{-2} \pm 4.83 \times 10^{-7}$ †	$\mathbf{3.913 \times 10^{-2} \pm 1.61 \times 10^{-7}}$ †
WFG3	$\mathbf{8.358 \times 10^{-2} \pm 4.31 \times 10^{-5}}$	$1.283 \times 10^{-1} \pm 6.10 \times 10^{-5}$ †	$1.544 \times 10^{-1} \pm 6.11 \times 10^{-5}$ †	$\mathbf{8.353 \times 10^{-2} \pm 1.28 \times 10^{-5}}$	$8.589 \times 10^{-2} \pm 4.04 \times 10^{-6}$ †
WFG4	$1.157 \times 10^{-1} \pm 5.75 \times 10^{-6}$	$1.235 \times 10^{-1} \pm 3.99 \times 10^{-6}$ †	$2.104 \times 10^{-1} \pm 5.92 \times 10^{-5}$ †	$\mathbf{9.481 \times 10^{-2} \pm 3.81 \times 10^{-6}}$ †	$1.533 \times 10^{-1} \pm 7.97 \times 10^{-7}$ †
WFG7	$\mathbf{1.147 \times 10^{0} \pm 6.21 \times 10^{-6}}$	$1.278 \times 10^{0} \pm 2.43 \times 10^{-5}$ †	$1.290 \times 10^{0} \pm 8.17 \times 10^{-5}$ †	$1.267 \times 10^{0} \pm 8.03 \times 10^{-6}$ †	$1.264 \times 10^{0} \pm 9.01 \times 10^{-5}$ †
WFG8	$2.216 \times 10^{0} \pm 5.05 \times 10^{-5}$	$2.175 \times 10^{0} \pm 2.79 \times 10^{-6}$ †	$2.912 \times 10^{0} \pm 4.51 \times 10^{-5}$ †	$\mathbf{2.173 \times 10^{0} \pm 9.51 \times 10^{-7}}$ †	$2.889 \times 10^{0} \pm 1.60 \times 10^{-5}$ †
DTLZ1	$1.083 \times 10^{-2} \pm 6.97 \times 10^{-5}$	$3.176 \times 10^{-2} \pm 7.46 \times 10^{-8}$ †	$2.728 \times 10^{1} \pm 2.44 \times 10^{2}$ †	$\mathbf{5.318 \times 10^{-3} \pm 2.49 \times 10^{-10}}$ †	$3.789 \times 10^{1} \pm 2.85 \times 10^{2}$ †
DTLZ6	$\mathbf{5.596 \times 10^{-3} \pm 9.35 \times 10^{-8}}$	$7.114 \times 10^{-3} \pm 3.14 \times 10^{-7}$ †	$6.585 \times 10^{-3} \pm 1.76 \times 10^{-7}$ †	$6.174 \times 10^{-3} \pm 1.62 \times 10^{-9}$ †	$\mathbf{5.681 \times 10^{-3} \pm 1.52 \times 10^{-8}}$
DTLZ7	$\mathbf{5.105 \times 10^{-3} \pm 2.05 \times 10^{-8}}$	$1.062 \times 10^{-2} \pm 2.15 \times 10^{-6}$ †	$7.098 \times 10^{-3} \pm 1.92 \times 10^{-7}$ †	$5.581 \times 10^{-3} \pm 8.04 \times 10^{-9}$ †	$5.640 \times 10^{-3} \pm 1.39 \times 10^{-8}$ †
ZDT1	$\mathbf{4.528 \times 10^{-3} \pm 3.32 \times 10^{-8}}$	$6.981 \times 10^{-3} \pm 4.99 \times 10^{-7}$ †	$6.856 \times 10^{-3} \pm 1.61 \times 10^{-7}$ †	$4.655 \times 10^{-3} \pm 4.97 \times 10^{-9}$ †	$5.371 \times 10^{-3} \pm 2.10 \times 10^{-9}$ †
ZDT2	$\mathbf{4.748 \times 10^{-3} \pm 5.31 \times 10^{-8}}$	$6.953 \times 10^{-3} \pm 2.93 \times 10^{-7}$ †	$5.922 \times 10^{-3} \pm 7.65 \times 10^{-8}$ †	$5.121 \times 10^{-3} \pm 2.46 \times 10^{-9}$ †	$4.398 \times 10^{-3} \pm 3.50 \times 10^{-9}$ †
W-D-L	-	13-1-1	13-0-2	8-2-5	11-1-3
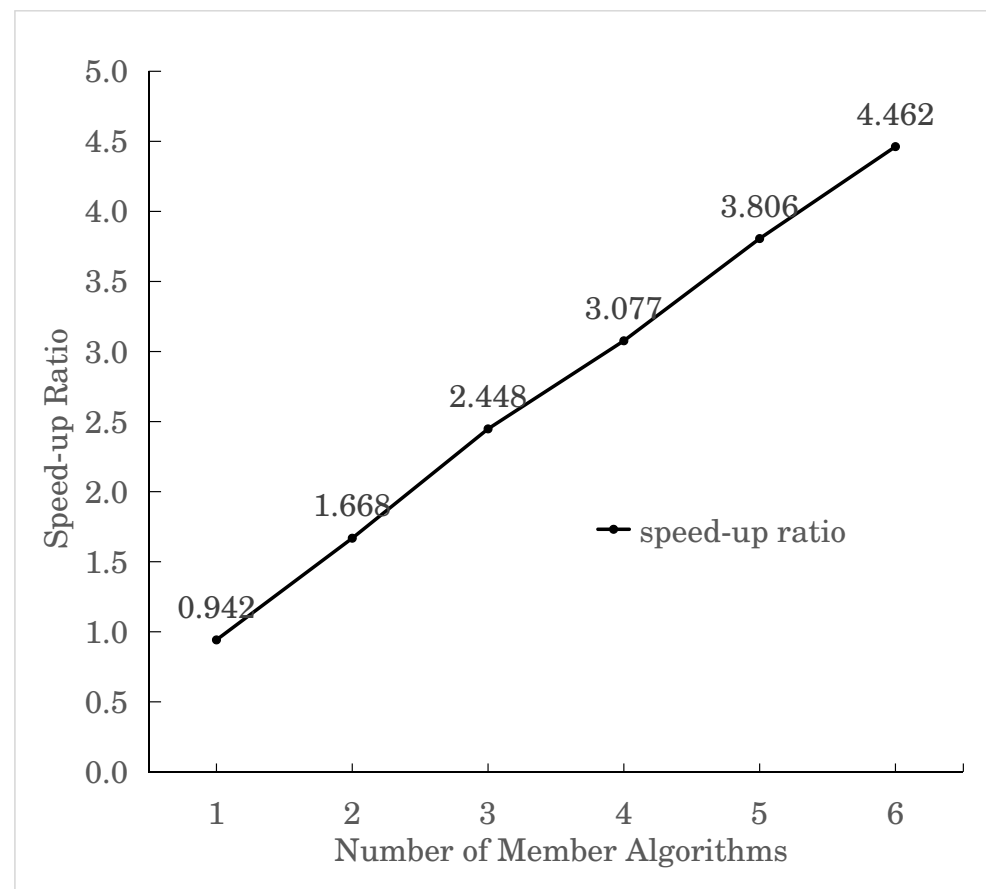
Figure 2. The speed-up ratios of running NSGA-II/PAP in parallel compared to running in serial, with different numbers of member algorithms.

5.5. Performances of Member Algorithms

Table 9 reports the independent performance of each member algorithm of NSGA-II/PAP, and the performance of NSGA-II/PAP, on the training problems in terms of the metric IHVR (see Section 4.2). The details of these six member algorithms can be found in Table 3. As before, on each problem, the best performance is indicated in bold; moreover, the best performance achieved among the member algorithms is indicated by an underline. One can observe that the performance of different member algorithms vary a lot. For example, member algorithm 1 performed poorly on UF10 but achieved an excellent performance on DTLZ3, while member algorithm 3 performed exactly the opposite. Actually, noticing that every member algorithm column in Table 9 has at least one cell underlined, this means each member algorithm achieved the best performance among all the member algorithms on at least one problem. Leveraging such performance complementarity eventually leads to a powerful PAP, i.e., NSGA-II/PAP, which performed better than any single member algorithm on all the problems.

Table 9. The mean of the IHVR values achieved by the member algorithms of NSGA-II/PAP, the variant of NSGA-II/PAP without the Restructure procedure, and NSGA-II/PAP, across 30 independent runs over the training set. For each problem, the best performance is indicated in bold (note that, for IHVR, the larger, the better), and the best performance achieved among the member algorithms is indicated by underline.

Problem	Member Algorithm 1	Member Algorithm 2	Member Algorithm 3	Member Algorithm 4	Member Algorithm 5	Member Algorithm 6	No Restructure	NSGA-II/PAP
UF3	0.575877	0.384182	0.514065	0.427070	0.545803	<u>0.577853</u>	0.597666	**0.630382**
UF4	0.981580	<u>0.986358</u>	0.984794	0.982849	0.981665	0.981615	**0.986358**	**0.986358**
UF6	0.544830	0.378446	0.409731	<u>0.603205</u>	0.472694	0.504911	0.648320	**0.791040**
UF7	0.805299	0.821830	0.789846	<u>0.884180</u>	0.768611	0.791250	0.890959	**0.931463**
UF10	0.124504	0.442486	<u>0.544413</u>	0.171657	0.240853	0.113756	**0.621673**	**0.621673**
WFG1	0.549970	0.116983	0.417594	0.136134	<u>0.732619</u>	0.500422	0.748021	**0.757083**
WFG5	0.845592	0.837363	0.838631	0.841177	<u>0.850955</u>	0.843478	0.850955	**0.852089**
WFG6	0.827467	0.869466	0.821787	<u>0.890711</u>	0.826054	0.813707	**0.893355**	**0.893355**
WFG9	0.704421	0.582577	0.684037	0.662125	<u>0.707536</u>	0.705735	0.707590	**0.707642**
DTLZ2	0.993334	<u>0.994437</u>	0.993827	0.994103	0.993781	0.993154	**0.994510**	**0.994510**
DTLZ3	<u>0.933205</u>	0.000739	0.010216	0.001359	0.006734	0.785523	0.933205	**0.944154**
DTLZ4	0.993408	<u>0.994484</u>	0.973001	0.994136	0.993670	0.993355	**0.994495**	**0.994495**
DTLZ5	0.993334	<u>0.994437</u>	0.993826	0.994103	0.993781	0.993153	**0.994510**	**0.994510**
ZDT3	0.993333	0.994189	0.888358	<u>0.996239</u>	0.975032	0.993530	**0.996299**	**0.996299**
ZDT4	0.961404	0.115557	0.273397	0.062209	0.702779	<u>0.965560</u>	0.971306	**0.972525**
ZDT5	**<u>1.000000</u>**	**<u>1.000000</u>**	**<u>1.000000</u>**	**<u>1.000000</u>**	**<u>1.000000</u>**	**<u>1.000000</u>**	**1.000000**	**1.000000**
ZDT6	0.989010	0.990608	0.312566	<u>0.992934</u>	0.971480	0.990346	**0.993043**	**0.993043**

5.6. Effectiveness of the Restructure Procedure

To investigate the effect of the Restructure procedure, we removed it from NSGA-II/PAP and tested the resulting PAP on the training set. Note that this PAP is a conventional PAP as defined in Equation (11), where the best solution set found by the member algorithms would be returned as its output. The "No Restructure" column in Table 9 presents the results. It can be observed that, due to the integration of the Restructure procedure, NSGA-II/PAP achieved a performance improvement on eight out of seventeen problems, clearly verifying the effectiveness of the procedure. Figure 3 further demonstrates that, in the construction process, as more member algorithms are included in the PAP, the Restructure procedure could bring a larger performance improvement.
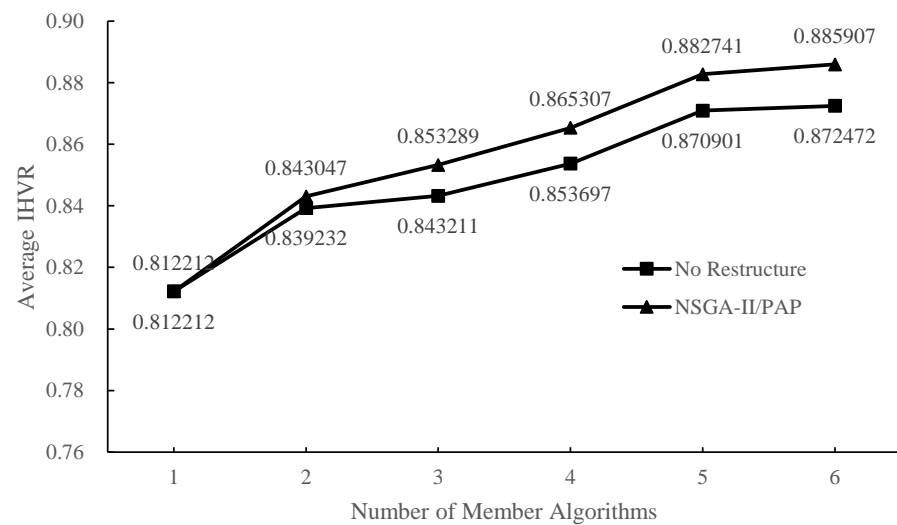
Figure 3. The average IHVR achieved by NSGA-II/PAP on the training set, with or without the Restructure procedure, as more member algorithms are added into the PAP during the construction process.

6. Conclusions

This work extended the realm of the automatic construction of PAPs to multi-objective optimization. Specifically, we proposed a variant of PAP, namely MOEAs/PAP, which involves a Restructure procedure to better determine the output solution set of the PAP. Then, we presented an automatic construction approach for MOEAs/PAP, which uses a novel metric to evaluate the performance of MOEAs/PAPs across multiple MOPs. Based on a training set of MOPs and an algorithm configuration space defined by several variants of NSGA-II, our approach could build MOEAs/PAPs that outperformed existing single-operator based-MOEAs and the state-of-the-art multi-operator-based MOEAs designed by human experts.

The promising results presented in this work have indicated the huge potential of the automatic construction of PAPs in the area of multi-objective optimization. Further directions for investigations are listed below.

- The algorithm configuration space used in this work is still defined based on the general algorithm framework of NSGA-II. In the literature, there have been some studies on developing highly parameterized MOEA frameworks [45,46]. It is valuable to apply our construction approach to these MOEA frameworks, hopefully leading to even better MOEAs/PAPs.
- When constructing MOEAs/PAPs, it is important to maintain the diversity among the member algorithms. Hence, the population diversity preservation schemes, such as negatively correlated search [47], can be introduced into the construction approach to promote cooperation between different member algorithms.
- In real-world applications, one may be unable to collect sufficient MOPs as training problems. How to automatically build powerful PAPs in these scenarios is also worth studying.
- The effectiveness of MOEAs/PAP has been primarily demonstrated through experimental evidence, but with an absence of theoretical analysis. A more thorough investigation of its exceptional performance is crucial for advancing our understanding, which, in turn, can lead to enhancements in its design and the development of a more comprehensive automatic construction algorithm.

References

1. Yang, P.; Zhang, L.; Liu, H.; Li, G. Reducing idleness in financial cloud services via multi-objective evolutionary reinforcement learning based load balancer. *arXiv* **2023**, arXiv:2305.03463.
2. Liu, S.; Tang, K.; Yao, X. Memetic search for vehicle routing with simultaneous pickup-delivery and time windows. *Swarm Evol. Comput.* **2021**, *66*, 100927. [CrossRef]
3. Liu, S.; Lu, N.; Hong, W.; Qian, C.; Tang, K. Effective and imperceptible adversarial textual attack via multi-objectivization. *arXiv* **2021**, arXiv:2111.01528.
4. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
5. Zhang, Q.; Li, H. MOEA/D: A Multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [CrossRef]
6. Zitzler, E.; Künzli, S. Indicator-based selection in multiobjective search. In Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, PPSN'2004, Birmingham, UK, 18–22 September 2004; pp. 832–842.
7. Zhou, A.; Qu, B.Y.; Li, H.; Zhao, S.Z.; Suganthan, P.N.; Zhang, Q. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm Evol. Comput.* **2011**, *1*, 32–49. [CrossRef]
8. Zitzler, E.; Laumanns, M.; Thiele, L. SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK Rep.* **2001**, *103*.
9. Coello, C.C.; Lechuga, M.S. MOPSO: A proposal for multiple objective particle swarm optimization. In Proceedings of the 2002 Congress on Evolutionary Computation, CEC'2002, Honolulu, HI, USA, 12–17 May 2002; pp. 1051–1056.
10. Knowles, J.; Corne, D. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In Proceedings of the 1999 Congress on Evolutionary Computation, CEC'99, Washington, DC, USA, 6–9 July 1999; pp. 98–105.
11. Wang, C.; Xu, R.; Qiu, J.; Zhang, X. AdaBoost-inspired multi-operator ensemble strategy for multi-objective evolutionary algorithms. *Neurocomputing* **2020**, *384*, 243–255. [CrossRef]
12. Gao, X.; Liu, T.; Tan, L.; Song, S. Multioperator search strategy for evolutionary multiobjective optimization. *Swarm Evol. Comput.* **2022**, *71*, 101073. [CrossRef]
13. Elsayed, S.; Sarker, R.; Coello, C.A.C. Fuzzy rule-based design of evolutionary algorithm for optimization. *IEEE Trans. Cybern.* **2017**, *49*, 301–314. [CrossRef]
14. Sun, J.; Liu, X.; Bäck, T.; Xu, Z. Learning adaptive differential evolution algorithm from optimization experiences by policy gradient. *IEEE Trans. Evol. Comput.* **2021**, *25*, 666–680. [CrossRef]
15. Wang, W.; Yang, S.; Lin, Q.; Zhang, Q.; Wong, K.; Coello, C.A.C.; Chen, J. An effective ensemble framework for multiobjective optimization. *IEEE Trans. Evol. Comput.* **2019**, *23*, 645–659. [CrossRef]
16. Coello, C.A.C.; Lamont, G.B.; Van Veldhuizen, D.A. *Evolutionary Algorithms for Solving Multi-Objective Problems*; Springer: Berlin/Heidelberg, Germany, 2007.
17. Goh, C.K.; Tan, K.C.; Liu, D.S.; Chiam, S.C. A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design. *Eur. J. Oper. Res.* **2010**, *202*, 42–54. [CrossRef]
18. Li, H.; Zhang, Q. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Trans. Evol. Comput.* **2008**, *13*, 284–302. [CrossRef]
19. Das, S.; Suganthan, P.N. Differential evolution: A Survey of the state-of-the-art. *IEEE Trans. Evol. Comput.* **2011**, *15*, 4–31. [CrossRef]
20. Peng, F.; Tang, K.; Chen, G.; Yao, X. Population-based algorithm portfolios for numerical optimization. *IEEE Trans. Evol. Comput.* **2010**, *14*, 782–800. [CrossRef]
21. Tang, K.; Peng, F.; Chen, G.; Yao, X. Population-based algorithm portfolios with automated constituent algorithms selection. *Inf. Sci.* **2014**, *279*, 94–104. [CrossRef]
22. Asanovic, K.; Bodik, R.; Demmel, J.; Keaveny, T.; Keutzer, K.; Kubiatowicz, J.; Morgan, N.; Patterson, D.; Sen, K.; Wawrzynek, J.; et al. A view of the parallel computing landscape. *Commun. ACM* **2009**, *52*, 56–67. [CrossRef]

23. Gebser, M.; Kaufmann, B.; Neumann, A.; Schaub, T. *clasp*: A conflict-driven answer set solver. In Proceedings of the 9th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR'2007, Tempe, AZ, USA, 15–17 May 2007; pp. 260–265.

24. Ralphs, T.K.; Shinano, Y.; Berthold, T.; Koch, T. Parallel solvers for mixed integer linear optimization. In *Handbook of Parallel Constraint Reasoning*; Hamadi, Y., Sais, L., Eds.; Springer: Berlin/Heidelberg, Germany, 2018; pp. 283–336.

25. Liu, S.; Tang, K.; Yao, X. Automatic construction of parallel portfolios via explicit instance grouping. In Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI'2019, Honolulu, HI, USA, 27 January–1 February 2019; pp. 1560–1567.

26. Tang, K.; Liu, S.; Yang, P.; Yao, X. Few-shots parallel algorithm portfolio construction via co-evolution. *IEEE Trans. Evol. Comput.* **2021**, *25*, 595–607. [CrossRef]

27. Liu, S.; Tang, K.; Yao, X. Generative adversarial construction of parallel portfolios. *IEEE Trans. Cybern.* **2022**, *52*, 784–795. [CrossRef]

28. Hamadi, Y.; Wintersteiger, C.M. Seven Challenges in Parallel SAT Solving. *AI Mag.* **2013**, *34*, 99–106. [CrossRef]

29. Liu, S.; Tang, K.; Lei, Y.; Yao, X. On performance estimation in automatic algorithm configuration. In Proceedings of the 34th AAAI Conference on Artificial Intelligence, AAAI'2020, New York, NY, USA, 7–12 February 2020; pp. 2384–2391.

30. Liu, S.; Zhang, Y.; Tang, K.; Yao, X. How good is neural combinatorial optimization? A systematic evaluation on the traveling salesman problem. *IEEE Comput. Intell. Mag.* **2023**, *18*, 14–28. [CrossRef]

31. Liu, S.; Yang, P.; Tang, K. Approximately optimal construction of parallel algorithm portfolios by evolutionary intelligence. *Sci. Sin. Technol.* **2023**, *53*, 280–290. [CrossRef]

32. Zitzler, E.; Deb, K.; Thiele, L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.* **2000**, *8*, 173–195. [CrossRef]

33. Deb, K.; Thiele, L.; Laumanns, M.; Zitzler, E. Scalable multi-objective optimization test problems. In Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02, Honolulu, HI, USA, 12–17 May 2002; pp. 825–830.

34. Huband, S.; Hingston, P.; Barone, L.; While, L. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans. Evol. Comput.* **2006**, *10*, 477–506. [CrossRef]

35. Zhang, Q.; Zhou, A.; Zhao, S.; Suganthan, P.N.; Liu, W.; Tiwari, S. *Multiobjective Optimization test Instances for the CEC 2009 Special Session and Competition*; Technical Report CES-487; University of Essex: Colchester, UK; Nanyang Technological University: Nanjing, China; Clemson University: Clemson, SC, USA, 2008; Volume 264, pp. 1–30.

36. Bosman, P.A.; Thierens, D. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Trans. Evol. Comput.* **2003**, *7*, 174–188. [CrossRef]

37. Zitzler, E.; Thiele, L. Multiobjective optimization using evolutionary algorithms — A comparative case study. In Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, PPSN'1998, Amsterdam, The Netherlands, 27–30 September 1998; pp. 292–301.

38. Emmerich, M.; Beume, N.; Naujoks, B. An EMO algorithm using the hypervolume measure as selection criterion. In Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimizatio, EMO'2005, Guanajuato, Mexico, 9–11 March 2005; pp. 62–76.

39. Rajagopalan, R.; Mohan, C.K.; Mehrotra, K.G.; Varshney, P.K. Emoca: An evolutionary multi-objective crowding algorithm. *J. Intell. Syst.* **2008**, *17*, 107–124. [CrossRef]

40. Mezura-Montes, E.; Reyes-Sierra, M.; Coello, C.A.C. Multi-objective optimization using differential evolution: A survey of the state-of-the-art. In *Advances in Differential Evolution*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 173–196.

41. Deb, K. An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl. Mech. Eng.* **2000**, *186*, 311–338. [CrossRef]

42. Freund, Y.; Schapire, R.E. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* **1997**, *55*, 119–139. [CrossRef]

43. Liu, S.; Peng, F.; Tang, K. Reliable robustness evaluation via automatically constructed attack ensembles. In Proceedings of the 35th AAAI Conference on Artificial Intelligence, AAAI'2023, Washington, DC, USA, 7–14 February 2023; pp. 8852–8860.

44. Lindauer, M.; Eggensperger, K.; Feurer, M.; Biedenkapp, A.; Deng, D.; Benjamins, C.; Ruhkopf, T.; Sass, R.; Hutter, F. SMAC3: A versatile bayesian optimization package for hyperparameter optimization. *J. Mach. Learn. Res.* **2022**, *23*, 2475–2483.

45. Bezerra, L.C.T.; López-Ibáñez, M.; Stützle, T. Automatic component-wise design of multiobjective evolutionary algorithms. *IEEE Trans. Evol. Comput.* **2016**, *20*, 403–417. [CrossRef]

46. Bezerra, L.C.T.; López-Ibáñez, M.; Stützle, T. Automatically designing state-of-the-art multi- and many-objective evolutionary algorithms. *Evol. Comput.* **2020**, *28*, 195–226. [CrossRef] [PubMed]

47. Tang, K.; Yang, P.; Yao, X. Negatively correlated search. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 542–550. [CrossRef]