2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Multiobjective Evolutionary Algorithm Portfolio: Choosing Suitable Algorithm for Multiobjective Optimization Problem

Shiu Yin Yuen and Xin Zhang

Abstract— **The concept of algorithm portfolio has a long history. Recently this concept draws increasing attention from researchers, though most of the researches have concentrated on single objective optimization problems. This paper is intended to solve multiobjective optimization problems by proposing a multiple evolutionary algorithm portfolio. Differing from previous approaches, each component algorithm in our portfolio method has an independent population and the component algorithms do not communicate in any way with each other. Another difference is that our algorithm introduces no control parameters. This parameter-less characteristic is desirable as each additional parameter requires independent parameter tuning or control. A novel score calculation method, based on predicted performance, is used to assess the contributions of component algorithms during the optimization process. Such information is used by an algorithm selector which decides, for each generation, which algorithm to use. Experimental results show that our portfolio method outperforms individual algorithms in the portfolio. Moreover, it outperforms the AMALGAM method.**

I. Introduction

THE CONCEPT of algorithm portfolio has a long history. Gomes and Selman define that a portfolio of algorithms is "*a collection of different algorithms and/or different copies of the same algorithm running on different processors*"[1]. They also point out that algorithm portfolio can also be run on one single processor. Their target is to solve hard combinatorial search problems. Later, researchers try algorithm portfolios to solve traveling salesperson problem (TSP) [2], supply chain optimization problem [3], classification problem [4], single objective real parameter optimization problem [5] [6], and multiobjective optimization problem (MOP) [7].

In this paper, we intend to solve MOP by constructing an algorithm portfolio. MOP widely exists in the real world (e.g., scheduling problem, data mining, chaotic system and so on). Usually, it has at least two objectives, and the objectives are often in conflict with each other. MOP can be formulated as [8] [9]:

$$\begin{aligned} min \quad & F(\mathbf{x}) = \left(f_1(\mathbf{x}), \dots, f_m(\mathbf{x})\right)^T \\ s.t. \quad & \mathbf{x} \in \Omega \end{aligned} \tag{1}$$

Shiu Yin Yuen and Xin Zhang are with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, China; e-mail:, kelviny.ee@cityu.edu.hk, xinzhang9-c@my.cityu.edu.hk.

The work described in this paper was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 125313].

where Ω is the decision space and $\mathbf{x}$ is a decision vector. $F(\mathbf{x})$ consists of m objective functions $f_i: \Omega \rightarrow R$, $1 \leq i \leq m$, and the objective space is R^m.

In MOP, the definitions of Pareto front and Pareto set are closely related. Without loss of generality, consider the optimization problem as a minimization. A vector $\mathbf{u} = u1,\dots,umT$ is said to weakly dominate another vector $\mathbf{v} = (v_1, \dots, v_m)^T$, denoted as $\mathbf{u} \preccurlyeq \mathbf{v}$, if and only if $\forall\, i \in \{1,2, \dots, m\}$, $u_i \leq v_i$. $\mathbf{u}$ is said to dominate $\mathbf{v}$, denoted as $\mathbf{u} \prec \mathbf{v}$, if and only if $\forall\, i \in \{1,2, \dots, m\}$, $u_i \leq v_i$ and $\mathbf{u} \neq \mathbf{v}$. A solution $\mathbf{x}^*$ of problem (1) is called a *Pareto optimal solution*, if and only if $\nexists \mathbf{x} \in \Omega$ such that $F(\mathbf{x}) \prec F(\mathbf{x}^*)$. The set of all Pareto optimal solutions is called *Pareto set* (PS) denoted as:

$$\text{PS} = \{\mathbf{x}^* \in \Omega | \nexists \mathbf{x} \in \Omega, F(\mathbf{x}) \prec F(\mathbf{x}^*)\} .$$

The mapping of PS in the objective vector space is called the *Pareto front* (PF):

$$\text{PF} = \{F(\mathbf{x}^*) | \mathbf{x}^* \in \text{PS}\} .$$

Based on the definition of domination, two different solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ have three possible relations:

1) $\boldsymbol{x}_1$ dominates $\boldsymbol{x}_2$;
2) $\boldsymbol{x}_1$ is dominated by $\boldsymbol{x}_2$;
3) $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are non-dominated with each other.

Since evolutionary algorithms (EAs) are usually designed with a population of solutions, they can find a set of Pareto optimal solutions in a single run. EAs handling MOP are called multiobjective evolutionary algorithms (MOEAs). These algorithms have two targets: 1) find a set of non-dominated solutions reaching PF as close as possible, and 2) maintain diversity of the solutions found such that they are uniformly spread out along the PS. Fulfilling the two targets, MOEAs could suggest a set of solutions to a practitioner such that he can choose the proper solution according to his subjective preference.

It is recognized that MOEAs are often highly tuned for particular problem domains and does not work well outside the designated domains [1]. Algorithm portfolio approach attempts to combine the strength of multiple MOEAs such that the resulting portfolio approach can deal with most, if not all of the MO problems, thus increasing the robustness of the MOEA.

This paper proposes a novel Multiple Evolutionary Algorithm for MO problems (MultiEA-MO). Our portfolio approach differs from others in the sense that each MOEA is

independent of others. Each MOEA has its own population which does not communicate with others (i.e., no migration or information sharing of solutions among populations). In this way, the communication problems about when to communicate and how to implement the communication are avoided. Furthermore, with this independence, each MOEA can keep its own evolutionary characteristic without being influenced by others. Another difference is that our algorithm introduces no control parameters. This parameter-less characteristic is desirable as each additional parameter requires independent parameter tuning or control, which is itself a challenging problem to solve. So it would do well if one can avoid them altogether and still obtains a good performance. As to the selection of which MOEA to run, a novel predictor method is used. We propose a scheme to calculate the score of each MOEA for estimating the future performance of MOEA. Then our predictor method could work and decide which MOEA is to be used for the current generation. After each generation, the predictor and the algorithm selector is invoked again to choose either the same or a different algorithm. Thus our MOEA can perform online algorithm switching as a function of the computational budget. This is desirable as the best EA to run should be a function of the budget available.

This paper is organized as follows. Section II gives a review about previous related works. Section III presents the proposed method. Section IV reports experimental results. The paper is concluded in Section V.

II. Related Works

There are several related research works. For convenience, these works are listed in the following categories:

1) Hyper-heuristic approaches. "*The definition of hyper-heuristics has been recently extended to refer to a search method or learning mechanism for selecting or generating heuristics to solve computational search problems*" [10]. These approaches can be classified into: heuristic selection and heuristic generation. The former means combining already existing heuristics in a higher-level search scheme, while the latter means creating new heuristics based on basic components (e.g., recombination operators and mutation operators) of existing heuristics. Cowling *et al.* propose an indirect genetic algorithm for optimizing a personnel scheduling problem [11]. Grobler *et al.* investigate the use of a set of evolutionary algorithms under a hyper-heuristic framework through different selection methods [12]. McClymont *et al.* employ a finite Markov chain model to adaptively select heuristics for solving MOP [13]. A complete and up to date survey of hyper-heuristic approaches can be found in [10].
2) Multiple algorithm approaches. Fukunaga builds a genetic algorithm portfolio for solving TSP [2]. Vrugt *et al.* propose an AMALGAM approach for MOP by combining four MOEAs [7]. Later they extend their approach to single objective problem [5]. Peng *et al.* propose a population-based algorithm portfolio [6]. This approach runs each component algorithm with a part of the predefined time budget and then encourages interaction amongst component algorithms with a migration scheme. Yuen *et al.* [14] propose an algorithm portfolio which for each generation, uses a predictor to estimate the performance of each component algorithm at a common future and then choose the best predicted performance algorithm for running the next generation. Burke *et al.* classify multi-population based approaches (e.g., multiple algorithm portfolio approach) as hyper-heuristics [10]. Here we would like to present them in different categories as they have different historical origins and motivations.

Besides the above, we would like to mention ensemble methods. Unlike constructing algorithm portfolio, ensemble methods are comprised of multiple operators (i.e., crossover operators and mutation operators) [15].

III. MultiEA-MO

This section presents the proposed MultiEA-MO method as well as the pseudo code of the method.

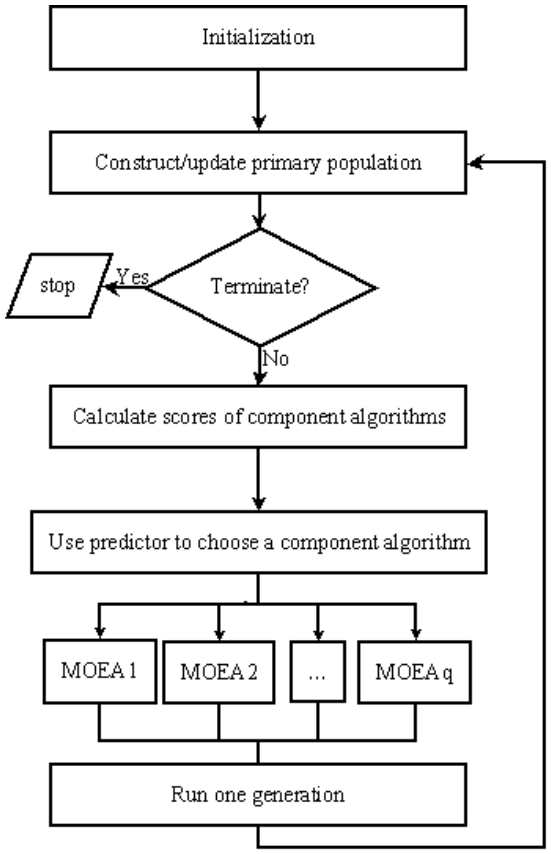

Fig. 1. Flow chart of MultiEA-MO method.

Given MOEAs A_i, $i = 1, \ldots, q$, which we have selected to compose our portfolio, each A_i is called a *component algorithm*. It has its own independent population (e.g., nondominated sorting genetic algorithm II (NSGA-II) [16]) or its own archive (e.g., adaptive multiobjective simulated annealing (AMOSA) [17]), depending on the strategy used by the original authors, though they have similar functionality (i.e., storing a certain number of non-dominated solutions). Our method also keeps a population of *all* non-dominated

solutions, which is called *primary population*.

Fig. 1 gives the flow chart of the proposed method. Algorithm 1-4 shows the pseudo code of the proposed portfolio method.

Algorithm 1: Pseudo code of MultiEA-MO method.

Input	Problem $f(\cdot)$, Search space Ω, A_i, $i = 1, \dots, q$.
	Set $score_i = 0$, $i = 1, \dots, q$. Set generation counter $g = 0$.
Step1	Randomly initialize component algorithms.
	Construct primary population (see Algorithm 2).
Step2	For $i = 1$ to q
	Run algorithm A_i until $n_i > 0$, such that A_i has contributed to build primary population.
Step3	Compute score of each component algorithm (see Algorithm 3).
	$g := g+1$.
Step4	While termination criteria are not fulfilled
Step5	Choose an algorithm A_i (see Algorithm 4).
Step6	Algorithm A_i evolves one generation.
Step7	Update primary population (see Algorithm 2).
	Update $score_i$, $i = 1, \dots, q$ (see Algorithm 3).
	$g := g+1$.
Output	Primary population

Algorithm 2: Pseudo code of constructing primary population.

Step1	Merge the population of each component algorithm into primary population.
Step2	Use non-dominated sorting technique to choose all non-dominated solutions to construct a primary population.
	Denote n_i as the number of solutions of algorithm A_i going into the primary population.

Algorithm 3: Pseudo code of score calculation of each component algorithm.

Step1	Sort n_i ($1 \le i \le q$) in ascending order and denote the sorted index as r_i.
Step2	$score_i += r_i$

Algorithm 4: pseudo code of algorithm selection.

Step1	For $j = 1$ to q
	Construct sub-curves for A_j.
	For each sub-curve l
	Least square line fit to get line parameters (a, b).
	Predict score at the smallest common future point $ps(g, l)$.
	Use all predicted scores to construct a probability distribution.
	Sample the distribution to get ps_j.
Step2	Choose the algorithm $i = \arg\max_j ps_j$.

The primary population is constructed by merging the population of each component algorithm, and use non-dominated sorting technique to choose all non-dominated solutions (Algorithm 2). This makes sure that the Pareto set is represented faithfully and fully. This has an advantage over techniques which stores only a fixed number of solutions (e.g. [16]) or a fixed sized archive (e.g., [17]) since no non-dominated solution is discarded. The disadvantage is that it involves more computation and memory.

Initially, our method needs to warm up by running each component algorithm such that we can obtain an initial impression about the performance of each component algorithm. This is realized by running each component algorithm until it contributes at least one non-dominated solution in the primary population (steps 1 and 2 of Algorithm 1).

The performance of an algorithm is assessed by computing a score. This is done by sorting the current number of non-dominated solutions of each algorithm in ascending order, which ranks the algorithms – the higher is the rank the better. The score is then updated by adding the rank (Algorithm 3).

For each algorithm, a curve of score vs the total number of evaluations (by all algorithms) is plotted and the method in [14] is used to predict its performance. The predicted performance of component algorithms at a common future point is then compared (step 1 of Algorithm 4). Then the algorithm with the best predicted performance is chosen (step 2 of Algorithm 4) to run one generation and generate new solutions (step 6 of Algorithm 1). The new solutions are then used to update the primary population and the current scores of all the component algorithms are updated (step 7 of Algorithm 1). This procedure is repeated for each new generation until the termination criteria are satisfied.

Note that our portfolio algorithm is parameter-less; it introduces no control parameters.

IV. Experimental Results

This section presents the experiments conducted on the thirteen unconstrained test functions from the special session on performance assessment of multiobjective optimization algorithms in Congress on Evolutionary Computation 2009 (CEC 2009) [18]. In this test suite, $F_1 - F_7$ are bi-objective functions, F_8, F_9, and F_{10} are tri-objective functions, and F_{11}, F_{12}, and F_{13} are five-objective functions. These functions are designed with complicated Pareto optimal front shapes.

A. Experimental Settings

To study the effectiveness of MultiEA-MO method, we choose five MOEAs, namely AMOSA [17], generalized differential evolution 3 (GDE3) [19], multiobjective differential evolution algorithm (MODEA) [20], MOEA based on decomposition (MOEA/D) [21], and NSGA-II [16]. These algorithms are chosen because:

1) They are representatives of a diverse set of MOEAs. AMOSA is a variant of multiobjective simulated annealing; GDE3 and MODEA are variants of multiobjective differential evolution (DE); MOEA/D decomposes a multiobjective problem into a number of single objective subproblems; NSGA-II is one of the most famous MOEAs.
2) They excel on different problems. This is observed from our empirical experience and also can be seen from Table A1 in Appendix. All five algorithms except GDE3 has rank 1 (i.e., the best performance) on some functions.
3) They have good performance as reported in associated papers [16] [17] [19]-[21] and the authors have released the source code.

The termination condition used in our experiment is: the algorithm terminates after 25000 function evaluations (FEs) for bi-objective problems (i.e., $F_1 - F_7$), while it terminates after 50000 FEs for tri- and five-objective problems (i.e., $F_8 - F_{13}$).

The parameter setting of each component algorithm is set to be the same as reported in the associated papers [16] [17] [19]-[21].

The inverted generational distance (IGD) performance

measure of CEC 2009 test set is used in this experiment. The final population or archive with size N_p = 100 found by an algorithm is used to calculate the IGD value, according to [18]. For our method, we use non-dominated sorting and crowding distance technique to select the N_p solutions. Each test function is independently run 30 times to obtain an average performance, according to [18]. The statistics of IGD values are listed in Table A1 of Appendix. Mann-Whitney U test (U test) is taken to statistically analyze the results of algorithms. It is a non-parametric statistical hypothesis test and a significant level 0.05 is used.

B. Comparing MultiEA-MO with Component Algorithms

B.1 Ranking Comparisons

The mean and standard deviation (std) IGD values of MultiEA-MO and each component algorithm are reported in Table A1 of Appendix. The rank table showing the rank of IGD values of all algorithms is given below in Table I. In this table, bolded values correspond to the best result amongst all algorithms.

Table I. Rank table of tested algorithms. We use the following notations for presenting the result: A1=AMOSA, A2=GDE3, A3=MODEA, A4=MOEA/D, A5=NSGA-II, and A6=MultiEA-MO.

	A1	A2	A3	A4	A5	A6
f_1	5	4	3	6	**1**	2
f_2	5	4	3	6	2	**1**
f_3	4	6	5	3	**1**	2
f_4	5	6	**1**	4	3	2
f_5	4	5	3	6	2	**1**
f_6	5	4	3	6	2	**1**
f_7	5	4	**1**	6	3	2
f_8	6	4	5	2	**1**	3
f_9	3	5	4	6	2	**1**
f_{10}	6	4	2	**1**	5	3
f_{11}	2	5	6	3	4	**1**
f_{12}	2	6	5	3	4	**1**
f_{13}	6	5	2	3	4	**1**
sum	58	62	43	55	34	**21**
std	1.391	0.832	1.601	1.833	1.325	**0.768**

It can be seen from Table I that the proposed MultiEA-MO method attains rank 1 in seven test functions, while NSGA-II attains rank 1 in three test functions, MODEA has rank 1 in two cases, MOEA/D has rank 1 in one case, AMOSA and GDE3 do not have rank 1 cases. The second last row of Table I sums up the ranks of each algorithm on all functions. The last row of this table shows the std value of the ranks of each algorithm. Clearly, MultiEA-MO obtains the lowest summed rank and std value compared with the other five algorithms. Note that the std value of GDE3 is lower than AMOSA, MODEA, MOEA/D, and NSGA-II, but it is the worst algorithm in terms of summed ranks on all functions. This phenomenon indicates that std can measure robustness of algorithms but it cannot assess the effectiveness of algorithms.

B.2 Pairwise Comparisons

For a statistical pairwise comparison of algorithms, Table II shows the *p*-value computed by U test between MultiEA-MO and each of the other algorithms. In this table, cells with dark gray background designates that MultiEA-MO significantly outperforms another algorithm with 95% confidence level, while cells with light gray background means that MultiEA-MO is significantly outperformed by another algorithm with 95% confidence level. The last row of Table II shows the counts in "+,=,-" format, where "+" means test functions in which MultiEA-MO attains significantly superior performance over another algorithm; "=" indicates test functions in which there is no significant difference between MultiEA-MO and another algorithm; "-" indicates functions in which MultiEA-MO displays significantly inferior performance compared with another algorithm (95% confidence level is used).

Table II. *p*-values computed by U test comparing MultiEA-MO with each of the five algorithms (AMOSA, GDE3, MODEA, MOEA/D, and NSGA-II).

	AMOSA	GDE3	MODEA	MOEA/D	NSGA-II
f_1	**7.221E-06**	**4.083E-05**	**2.510E-02**	**3.018E-11**	**4.856E-03**
f_2	**3.018E-11**	**6.687E-11**	**4.075E-11**	**3.018E-11**	**7.655E-05**
f_3	**1.698E-08**	**3.018E-11**	**3.016E-11**	**1.861E-06**	**3.352E-08**
f_4	**3.018E-11**	**3.016E-11**	1.334E-01	**3.014E-11**	1.493E-01
f_5	**3.020E-11**	**7.736E-06**	**3.958E-08**	**3.018E-11**	**6.524E-07**
f_6	**3.690E-11**	5.012E-02	4.035E-01	**6.066E-11**	6.843E-01
f_7	**3.352E-08**	5.793E-01	**8.563E-04**	**1.558E-08**	2.519E-01
f_8	**9.829E-08**	3.711E-01	1.858E-01	6.309E-01	**3.005E-04**
f_9	**2.433E-05**	**1.070E-09**	**5.091E-06**	**3.020E-11**	**7.199E-05**
f_{10}	**1.028E-06**	6.520E-01	2.519E-01	2.398E-01	1.413E-01
f_{11}	**5.494E-11**	**3.020E-11**	**3.020E-11**	**3.020E-11**	**3.020E-11**
f_{12}	**2.959E-05**	**3.020E-11**	**3.020E-11**	**3.020E-11**	**3.020E-11**
f_{13}	**3.020E-11**	**3.020E-11**	**3.018E-11**	**3.020E-11**	**3.018E-11**
	13+,0=,0-	9+,4=,0-	9+,4=,0-	11+,2=,0-	6+,3=,4-

Observed from Table II, MultiEA-MO is significantly superior to AMOSA in 13 out of 13 test functions. It is significantly superior to GDE3, MODEA, and MOEA/D in 9, 9, and 11 out of 13 test functions, while MultiEA-MO is not significantly outperformed by AMOSA, GDE3, MODEA, and MOEA/D in any test case. MultiEA-MO is significantly superior to, equal to, and inferior to NSGA-II in 6, 3, and 4 out of 13 test cases, respectively. Thus our method overall attains slightly better performance than NSGA-II on the chosen test functions, while it is significantly superior to AMOSA, GDE3, MODEA, and MOEA/D.

Through the above comparison, we can conclude that our method is able to combine the strength of component algorithms and achieves a good and robust performance. Furthermore, a positive *synergy effect* is observed from Table I that our method has seven rank 1 test cases (i.e., $f_2, f_5, f_6, f_9, f_{11} - f_{13}$). The synergy effect means that the use of multiple component algorithms produces better result (ranking in this case) than using a component algorithm alone. This is so especially for tri- and five-objective problems.

C. Comparing MultiEA-MO with AMALGAM

To demonstrate the usefulness of the proposed method, it is compared with AMALGAM, which is one of the successful multiple method combinations. In this experiment, AMALGAM is tested using the default parameter setting in [7]. The IGD values for AMALGAM are reported in Appendix.

Table III. p-values computed by U test comparing MultiEA-MO with AMALGAM.

	AMALGAM
f_1	**4.218E-03**
f_2	**1.438E-10**
f_3	**1.994E-08**
f_4	6.842E-01
f_5	**2.992E-11**
f_6	**3.367E-05**
f_7	**3.554E-04**
f_8	**6.013E-07**
f_9	**3.159E-10**
f_{10}	**1.244E-05**
f_{11}	**2.992E-11**
f_{12}	**2.992E-11**
f_{13}	**3.018E-11**
	10+,1=,2-

The p-values computed by U test are shown in Table III. The pattern of Table III is similar to that of Table II. In Table III, MultiEA-MO is significantly superior to, equal to and inferior to AMALGAM in 10, 1, and 2 test functions. Specifically, MultiEA-MO vs. AMALGAM on bi-objective functions $F_1 - F_7$ is 4+, 1=, 2-; the proposed method slightly outperforms AMALGAM on these seven functions. MultiEA-MO vs. AMALGAM on tri- and five-objective functions $F_8 - F_{13}$ is 6+, 0=, 0-; the proposed method performs significantly better than AMALGAM on these six functions. Thus MultiEA-MO performs much better than AMALGAM on the harder problems with more objectives.

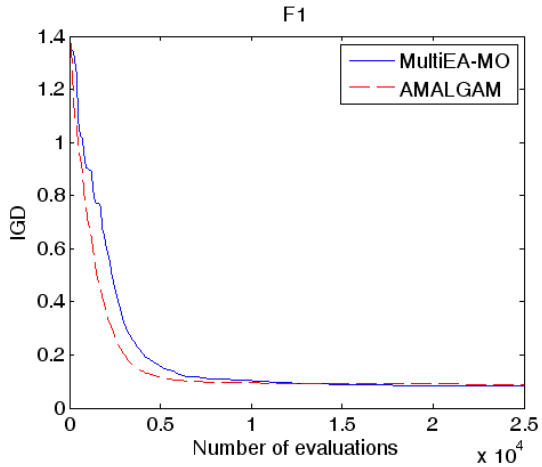

Fig. 2. Diagram of IGD versus number of function evaluations: F_1.

Fig. 2 shows the curve of IGD value versus number of FEs of F_1 for MultiEA-MO and AMALGAM. Since this is a synthetic function and its Pareto optimal front is known, we can calculate IGD values at any time of the evolutionary process of an algorithm. Seen from Fig. 2, from the start to about the 13000th FEs, AMALGAM (dotted line) attains better performance than MultiEA-MO (solid line); while after about the 13000th FEs MultiEA-MO attains smaller IGD values than AMALGAM. As claimed in [7], AMALGAM converges faster which is also verified in Fig. 2, however, our method converges slower but better in later evolutionary stages. This is reasonable since AMALGAM utilizes a common population for component algorithms, when the common population prematurely converges, AMALGAM cannot escape from local optima; on the other hand, our portfolio approach utilizes independent population for component algorithms, the probability of convergence of primary population is approximately equal to the product of the probability of convergence of component algorithms. In the case that one component algorithm converges to local optima, the other component algorithms can continue to search for better solutions. Therefore, our approach converges slower but better than AMALGAM.

V. Conclusions

This paper proposes a multiple evolutionary algorithm portfolio for multiobjective optimization problem (MultiEA-MO). Note that the idea of algorithm portfolio is not novel at all [22]. Many successful multiple method or multiple operator combination techniques have been reported recently [3]-[7]. However, comparing with these approaches, our algorithm has the following characteristics:

1) Each component algorithm is run independently. This avoids the difficult problem of designing effective communication schemes between component algorithms that work well in general. Moreover, it preserves the search philosophy and characteristics of each component algorithm.
2) The predicted performance at a common future point is used to select algorithms. Thus algorithms can be selected as a function of the computational budget available.
3) Our approach is parameter-less; it does not introduce any additional control parameter and has the significant advantage of avoiding the difficult parameter tuning and control problem [23].

Our method is instantiated by using five multiobjective evolutionary algorithms (MOEAs), namely AMOSA, GDE3, MODEA, MOEA/D, and NSGA-II. All unconstrained functions from CEC 2009 multiobjective optimization problem competition are taken as benchmarks. Experimental results show that our method can achieve better and more robust performance than the five individual component algorithms both when put together in a ranking contest or when compared pairwise. Furthermore, the effectiveness of our algorithm is demonstrated by comparing with AMALGAM. We also study the convergence behavior in one case (F_1). Although in this case, AMALGAM converges faster than our method in the early stages of evolutionary process, our method can achieve better performance than AMALGAM in later evolutionary stages. This behavior can be explained by the use of independent population, which reduces the overall probability of convergence of the portfolio.

APPENDIX

Table A1. Mean and standard deviation (std) of IGD values obtained by each algorithm tested on unconstrained functions of CEC 2009.

		AMOSA	GDE3	MODEA	MOEA/D
f_1	mean	0.1032	0.0979	0.0939	0.2465
	std	0.0151	0.0091	0.0203	0.1028
f_2	mean	0.0583	0.0451	0.0450	0.0920
	std	0.0106	0.0047	0.0039	0.0531
f_3	mean	0.3184	0.3795	0.3754	0.3173
	std	0.0329	0.0105	0.0117	0.0408
f_4	mean	0.0845	0.0902	0.0521	0.0776
	std	0.0047	0.0028	0.0033	0.0034
f_5	mean	0.3757	0.4408	0.3739	0.5470
	std	0.0943	0.2780	0.1649	0.0834
f_6	mean	0.4293	0.2982	0.2762	0.5890
	std	0.0998	0.1133	0.1322	0.1201
f_7	mean	0.2848	0.1582	0.0524	0.4695
	std	0.1095	0.1381	0.0182	0.1853
f_8	mean	0.3767	0.2498	0.2590	0.2040
	std	0.0817	0.0219	0.0290	0.0445
f_9	mean	0.1965	0.2214	0.2061	0.2504
	std	0.0641	0.0345	0.0627	0.0337
f_{10}	mean	0.5796	0.4665	0.4084	0.3869
	std	0.1105	0.2510	0.1744	0.1728
f_{11}	mean	0.5558	1.5048	1.7042	1.0757
	std	0.0857	0.1521	0.2124	0.2595
f_{12}	mean	514.03	2087.25	1690.86	1370.14
	std	178.46	208.02	123.34	270.03
f_{13}	mean	4.6229	3.4838	1.9241	2.5354
	std	0.2587	0.2594	0.0258	0.4076
		NSGA-II	MultiEA-MO	AMALGAM	
f_1	mean	0.0701	0.0832	0.0941	
	std	0.0175	0.0152	0.0082	
f_2	mean	0.0380	0.0317	0.0432	
	std	0.0063	0.0047	0.0040	
f_3	mean	0.2218	0.2618	0.3238	
	std	0.0188	0.0226	0.0405	
f_4	mean	0.0539	0.0536	0.0543	
	std	0.0020	0.0038	0.0045	
f_5	mean	0.2622	0.2049	0.3516	
	std	0.0772	0.0089	0.1236	
f_6	mean	0.2524	0.2329	0.1889	
	std	0.1105	0.0490	0.2102	
f_7	mean	0.1279	0.1067	0.0501	
	std	0.1346	0.0641	0.0117	
f_8	mean	0.1584	0.2183	0.3049	
	std	0.0438	0.0715	0.0239	
f_9	mean	0.1960	0.1243	0.4302	
	std	0.0582	0.0509	0.1235	
f_{10}	mean	0.4845	0.4180	0.5549	
	std	0.1393	0.1078	0.1074	
f_{11}	mean	1.1878	0.3729	2.9390	
	std	0.1887	0.0192	0.3058	
f_{12}	mean	1679.72	345.1056	662.28	
	std	263.91	60.0262	29.34	
f_{13}	mean	3.2994	1.8136	1.8722	
	std	0.1970	0.0044	0.0182	

REFERENCES

[1] C. P. Gomes and B. Selman, "Algorithm portfolio design: theory vs. practice," in *Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97)*, Morgan Kaufman, Publishers, 1997, pp. 190-197.

[2] A. S. Fukunaga, "Genetic algorithm portfolios," in *Proceedings of IEEE Congress on Evolutionary Computation*, La Jolla, CA, 2000, pp. 1304-1311.

[3] S. R. Yadav, R. R. M. R. Muddada, M. K. Tiwari, and R. Shankar, "An algorithm portfolio based solution methodology to solve supply chain optimization problem," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8407-8420, 2009.

[4] S. Srikamdee, S. Rimcharoen, and K. Chinnasarn, "NeuroEAs-Based algorithm portfolios for classification problems," in *The fourth International Conference on Knowledge and Smart Technology (KST)*, Chonburi, Thailand, 2012, pp. 62-68.

[5] J. A. Vrugt, B. A. Robinson, and J. M. Hyman, "Self-adaptive multimethod search for global optimization in real-parameter spaces," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 243-259, 2009.

[6] F. Peng, K. Tang, G. Chen, and X. Yao, "Population-based algorithm portfolios for numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, pp. 782-800, 2010.

[7] J. A. Vrugt and B. A. Robinson, "Improved evolutionary optimization from genetically adaptive multimethod search," *Proceedings of the National Academy of Sciences*, vol. 104, no. 3, pp. 708-711, 2007.

[8] A. Zhou, B. Qu, H. Li, S. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: a survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

[9] S. W. Leung, X. Zhang, and S. Y. Yuen, "Multiobjective differential evolution algorithm with opposition-based parameter control," in *Proceedings of IEEE Congress on Evolutionary Computation*, Jun 2012, pp. 2106-2113.

[10] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu, "Hyper-heuristics: a survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695-1724, 2013.

[11] P. Cowling, G. Kendall, and L. Han, "An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem," in *Proceedings of IEEE Congress on Evolutionary Computation*, Hawaii, USA, 2002, pp. 1185-1190.

[12] J. Grobler, A.P. Engelbrecht, G. Kendall, and V.S.S. Yadavalli, "Investigating the impact of alternative evolutionary selection strategies on multi-method global optimization," in *Proceedings of IEEE Congress on Evolutionary Computation*, New Orleans, LA, 2011, pp. 2337-2344.

[13] K. McClymont and E. C. Keedwell, "Markov chain hyper-heuristic (MCHH): an online selective hyper-heuristic for multiobjective continuous problems," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO'11)*, 2011, pp. 2003-2010.

[14] S. Y. Yuen, C. K. Chow, and X. Zhang, "Which algorithm should I choose at any point of the search: an evolutionary portfolio approach," in *Proceedings of the 14th International Conference on Genetic and Evolutionary Computation Conference (GECCO'13)*, Jul 2013, pp. 567-574.

[15] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, no. 2, pp. 1679-1696, 2011.

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

[17] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, "A simulated annealing-based multiobjective optimization algorithm: AMOSA," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 3, pp. 269-283, 2008.

[18] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari, "Multiobjective optimization test instances for the cec 2009 special session and competition," The School of Computer Science and Electronic Engineering, University of Essex, Tech. Rep. CES-487, Apr. 2008.

[19] S. Kukkonen and J. Lampinen, "Generalized differential evolution for constrained multi-objective optimization," in *Multi-objective Optimization in Computational Intelligence: Theory and Practice*, L. T. Bui and S. Alam, Eds. Hershey, New York: Information Science Reference, 2008, ch. 3, pp. 43–75.

[20] M. Ali, P. Siarry, and M. Pant, "An efficient differential evolution based algorithm for solving multi-objective optimization problems," *European Journal of Operational Research*, vol. 217, no. 2, pp. 404–416, 2012.

[21] Q. Zhang and H. Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.
[22] J. A. Vrugt, B. A. Robinson, and J. M. Hyman, "Comment on paper "multi-strategy ensemble evolutionary algorithm for dynamic multi-objective optimization" by Wang and Li," *Memetic Computing*, vol. 2, no. 1, pp. 161-162, 2010.
[23] F. G. Lobo, C. F. Lima, and Z. Michalewicz (eds.), *Parameter Setting in Evolutionary Algorithms*, Springer, Berlin, Germany, 2007.