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

An MOEA/D with Multiple Differential Evolution Mutation Operators

Yang Li
Department of Computer Science and Technology
East China Normal University
Shanghai, China, 200241
Email: yangli@student.ecnu.edu.cn

Aimin Zhou
Department of Computer Science and Technology
East China Normal University
Shanghai, China, 200241
Email: amzhou@cs.ecnu.edu.cn

Guixu Zhang
Department of Computer Science and Technology
East China Normal University
Shanghai, China, 200241
Email: gxzhang@cs.ecnu.edu.cn

***Abstract*—In evolutionary algorithms, the reproduction operators play an important role. It is arguable that different operators may be suitable for different kinds of problems. Therefore, it is natural to combine multiple operators to achieve better performance. To demonstrate this idea, in this paper, we propose an MOEA/D with multiple differential evolution mutation operators called MOEA/D-MO. MOEA/D aims to decompose a *multiobjective optimization problem (MOP)* into a number of *single objective optimization problems (SOPs)* and optimize those SOPs simultaneously. In MOEA/D-MO, we combine multiple operators to do reproduction. Three mutation strategies with randomly selected parameters from a parameter pool are used to generate new trial solutions. The proposed algorithm is applied to a set of test instances with different complexities and characteristics. Experimental results show that the proposed combining method is promising.**

I. Introduction

In the past decades, *multiobjective optimization problems (MOPs)* have attracted much attention. In this paper, we consider continuous MOPs, without loss of generality, which can be formulated as follows:

$$\begin{array}{ll} \min & F(x) = (f_1(x), f_2(x), ..., f_m(x)) \\ \text{s.t.} & x \in \prod_{i=1}^{n} [a_i, b_i] \end{array} \quad (1)$$

where $x = (x_i, ..., x_n)^T \in R^n$ is a decision variable vector, $\prod_{i=1}^{n} [a_i, b_i] \in R^n$ is the feasible region of the search space, $f_i : R^n \to R, i = 1, ..., m$, is a continuous mapping, and $F(x)$ is an objective vector.

In an MOP, the objectives usually conflict with each other. As a result, there does not exist one optimal solution that can optimize all the objectives in (1) at the same time. Let $u, v \in R^n$ be two vectors, u is said to dominate v if $u_i \leq v_i$ for all $i = 1, ..., n$, and $u \neq v$. A vector $x^* \in \prod_{i=1}^{n} [a_i, b_i]$ is called *Pareto optimal* if there doesn't exist an $x \in \prod_{i=1}^{n} [a_i, b_i]$ such that $F(x)$ dominates $F(x^*)$. All the Pareto optimal vectors are called *Pareto set (PS)* and their objective vectors, $PF = \{F(x) \in R^m | x \in PS\}$, are called the *Pareto front*.

A number of multiobjective evolutionary algorithms (MOEAs) have been proposed to deal with MOPs [1], [2], [3], [4], since an MOEA can obtain the approximation in a single run. Among them, the multiobjective evolutionary algorithm based on decomposition, called MOEA/D, has shown promising results [5]. The basic idea of MOEA/D is to decompose an MOP into a set of single objective problems (SOPs), then the SOPs are optimized simultaneously to get a solution set of the original MOP. Neighborhood is another key concept in MOEA/D. Each SOP has a neighborhood which contains the most similar SOPs. The assumption is that the SOPs in the same neighborhood may have similar fitness landscapes and their solutions might be close to each other. The reproduction and update procedures are done within neighborhoods. Since MOEA/D was proposed in 2007, a lot of variants which mainly modify the reproduction operator have been presented. The original version of MOEA/D uses SBX (simulated binary crossover) as the reproduction operator, then it is replaced with a differential evolution (DE) mutation operator in MOEA/D-DE [6] which performs well on MOPs with complicated PS shapes. Zhou et al. [7] proposed a probability model based reproduction operator within the MOEA/D framework. A new version of MOEA/D with uniform design is proposed in [8].

Many reproduction operators are proposed for solving SOPs or MOPs, but they are usually suitable for a certain set of problems. And theory and experiments have demonstrated that it is impossible to design a single algorithm which could always have the best performance on a diverse set of optimization problems [9]. The reason might be that operators have different search abilities on different regions in the search space, at different stages of the search, or on different optimization problems. Some researchers try to make full use of one operator, in which its parameters can be changed during the search, to exploit its search ability. In [10], for example, a parameter control mechanism which uses the solution diversity information is proposed to improve NSGA-II. Others turn to find ways of using a set of operators which can perform together, complement each other and augment their search capabilities [11]. Vrugt te al. [12] proposed a method to solve MOPs and it runs multiple optimization algorithms simultaneously and adaptively select the favor individual algorithms to generate offsprings for each individual according to their reproductive success rates. In [13], each search operator has a subpopulation and the subpopulation size varies adaptively according to its reproductive success rate.

In this paper, we proposes a method to use multiple DE mutation operators within the MOEA/D framework to tackle

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

continuous MOPs. Firstly three DE mutation strategies are selected for MOPs and then the parameter for each mutation strategy is tuned. Finally the three DE operators with randomly selected parameters from the parameter pool are used to produce new trial solutions.

The rest of the paper is organized as follows. In Section II, we present the proposed algorithm with details. Experimental results are reported in Section III, and the details how the parameter is tuned are described. Finally, Section IV concludes the paper.

II. MOEA/D WITH MULTIPLE DE MUTATION OPERATORS

In this section, the general framework of MOEA/D is firstly introduced. Then the method of using multiple DE mutation operators to produce new trial solutions is described.

A. MOEA/D Framework

MOEA/D decomposes an MOP into a set of SOPs (or subproblems) and the optimal solution of each subproblem will hopefully be a Pareto optimal solution of the MOP. In principle, any decomposition technique can be used. In this paper, the Tchebycheff technique [14] is adopted. A subproblem can be defined as:

$$\begin{aligned} \min \quad & g(x|\lambda, z^*) = \max_{1\le i\le m} \{\lambda_i |f_i(x) - z_i^*|\} \\ \text{s.t.} \quad & x \in \Omega \end{aligned} \tag{2}$$

where $\lambda = (\lambda_1, \cdots, \lambda_m)$ is a weight vector, i.e., $\lambda_i \geq 0$ for all $i = 1, \cdots, m$ and $\sum_{i=1}^{m} \lambda_i = 1$. $z^* = (z_1^* \cdots, z_m^*)$ is a reference point, i.e., $z_i^* = min\{f_i(x)|x \in \Omega\}$ for each $i = 1, \cdots, m$. In MOEA/D, N weight vectors $\lambda^1, \cdots, \lambda^N$ are used to define N subproblems. Subproblem i is with weight vector λ^i, and its objective function is denoted as $g(x|\lambda^i, z^*)$. For simplicity, we use $g^i(x)$ to denote $g(x|\lambda^i, z^*)$ hereafter.

MOEA/D attempts to deal with these N subproblems simultaneously. Observing that $g^i(x)$ is continuous of λ, the optimal solutions of neighboring subproblems whose weight vectors are close should be close in the decision space. Neighborhood concept is then employed to improve search capability. In MOEA/D, T closest weight vectors in $\{\lambda^1, \cdots, \lambda^N\}$ to weight vector λ^i constitute the neighborhood of λ^i. Therefore, the neighborhood of subproblem i consists of all the subproblems whose weight vectors are in the neighborhood of λ^i. A subproblem is optimized mainly within its neighboring subproblems, such as selecting parent solutions and updating parent solutions.

In MOEA/D, the following elements of the ith ($i = 1, \ldots, N$) subproblem will be maintained:

- the objective function $g^i(x)$ in (2),
- the current solution x^i and the objective vector of x^i, i.e., $F^i = F(x^i)$, and
- the index set of its neighboring subproblem B^i.

Thus, we could use a tuple of (x^i, F^i, B^i, g^i) to denote the i^{th} subproblem. MOEA/D also needs to maintain:

- a reference point $z^* = (z_1^*, \ldots, z_m^*)^T$.

Algorithm 1: Main Framework of MOEA/D

```
1  Initialize a set of subproblems (x^i, F^i, B^i, g^i),
   i = 1, ··· , N;
2  Initialize the reference point z* as z*_j = min_{i=1,···,N} f_j(x^i),
   j = 1, ··· , m;
3  while not terminate do
4      foreach i ∈ perm({1, ..., N}) do
5          if rand() < p_n then
6              π = perm(B^i);
7          else
8              π = perm({1, ..., N});
9          end
10         (y1, ..., yK) = OP(π);
11         foreach y ∈ {y1, ..., yK} do
12             for j = 1 : m do
13                 if f_j(y) < z*_j then
14                     Set z*_j = f_j(y);
15                 end
16             end
17             Set counter c = 0;
18             foreach j ∈ π do
19                 if g^j(y) < g^j(x^j) and c < C then
20                     Replace x^j by y;
21                     Set c = c + 1;
22                 end
23             end
24         end
25     end
26 end
```

The main framework of MOEA/D is presented in Algorithm 1. We would like to make the following comments on the MOEA/D framework.

- N is the number of subproblems.
- T is the neighborhood size.
- p_n is an algorithm parameter to balance the exploitation and exploration of the search.
- C is the maximal number of parent solutions allowed to be replaced for each child solution.
- $perm(\cdot)$ randomly permutes the input values, and $rand()$ generates a random real number in $[0, 1]$.
- *Line 1:* The initial solutions of the subproblems are uniformly randomly sampled from the feasible search space. The weight vectors of the subproblems are uniformly distributed. The detail on generating these weight vectors is referred to [6].

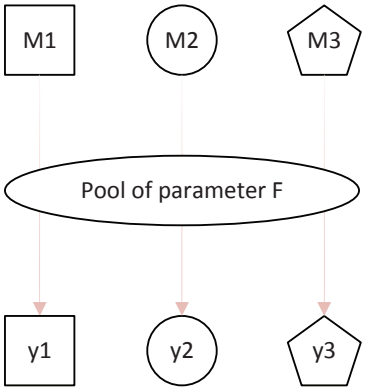


Fig. 1. Illustration of combining multiple DE strategies to generate solutions: each strategy (M1, M2 or M3) randomly selects an F value from the pool to generate a new solution.

- *Line 3:* A maximal number of generations is used as the termination criteria.
- *Line 4:* In each generation, a subproblem is randomly selected to avoid the influence of always selecting subproblems in the same order.
- *Lines 5-9:* The parent set π is either the neighborhood or the whole population which is controlled by p_n. If the neighborhood is used, the algorithm will do exploitation; otherwise, it will do exploration.
- *Lines 17-23:* The parent solutions are updated by child solutions in a random order to avoid the influence of always updating in the same order. To prevent premature convergence, at most C solutions can be updated. The update is based on the subproblem objective values.

$OP(\pi)$ in *Line 10* of Algorithm 1 is the reproduction procedure and new solutions are generated based on the parent solutions from the neighborhood π. One or multiple operators can be used to generate K new solutions at one time. Usually, K is set to 1, but in our algorithm it is 3.

B. Combining Multiple DE Mutation Operators

Differential evolution (DE) proposed by Storn and Price [15] is a promising evolutionary algorithm and many variants have been developed. But the performance of DE is sensitive to its parameters and the best settings for the parameters can be different for different optimization problems [16]. Based on the above analysis, Qin et al. [17] proposed a self-adaptive differential evolution algorithm (SaDE) for SOPs which chooses differential evolution strategies with probability proportional to their previous success rates. Mallipeddi et al. [16] presented an ensemble of generation strategies and control parameters of DE (ESPDE). ESPDE aims to find good combinations of strategies and control parameters. Another algorithm proposed for SOPs called composite differential evolution (CoDE) uses a random method to combine several trial vector generation strategies with several control parameter settings, and it generates more than one candidate solution for each individual at one generation [18].

However, it's also important to study the performance of combining multiple operators for solving MOPs and there is few work has been done. Motivated by CoDE, in this paper, we propose a method to use multiple DE operators within the MOEA/D framework to solve continuous MOPs. Firstly three frequently used mutation strategies, named M1, M2 and M3, are selected for MOPs. Then the parameter F for the mutation strategies is tuned, and we get a parameter pool for F. Finally the three mutation strategies with randomly selected parameters from the parameter pool are used to produce new trial solutions. Fig. 1 illustrates how to combine DE operators.

DE mutation strategies need to be invariant of any orthogonal coordinate rotation to deal with complicated PSs of MOPs [6]. So crossover operator may be not suitable for MOPs. In this paper, parameter CR for all the three mutation strategies is set to 1.0. For similarity, the three mutation strategies are shown as follows without crossover.

$$\begin{cases} M1 : \bar{y} = x^i + F \cdot (x^{r1} - x^{r2}) \\ M2 : \bar{y} = x^i + rand \cdot (x^{r1} - x^{r2}) + F \cdot (x^{r3} - x^{r4}) \\ M3 : \bar{y} = x^{r1} + rand \cdot (x^{r2} - x^{r3}) + F \cdot (x^{r4} - x^{r5}) \end{cases} \tag{3}$$

where x^i is the current solution of the i^{th} subproblem and $r1$, $r2$, $r3$, $r4$, $r5$ are indexes randomly selected from π. Strategy M1 is the most commonly used strategy in the literature. In strategies M2 and M3, two difference vectors are added to a base vector which is the current vector in M2 and a random one in M3. This might lead to a better perturbation and more different trial solution.

If any element of $\bar{y}$ is out of boundary of the feasible search space, its value is reset to a randomly selected value beyond the boundary. The polynomial mutation is then applied to $\bar{y}$ to generate $y = (y_1, \ldots, y_n)$ as follows:

$$y_k = \begin{cases} \bar{y}_k + \sigma_k \times (b_k - a_k) & \text{if } rand < p_m, \\ \bar{y} & \text{otherwise} \end{cases} \tag{4}$$

with

$$\sigma_k = \begin{cases} (2 \times rand)^{\frac{1}{\eta+1}} - 1 & \text{if } rand < 0.5, \\ 1 - (2 - 2 \times rand)^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases}$$

where $rand$ is a uniform random number in $[0, 1]$. The distribution index η and the mutation rate p_m are two control parameters. a_k and b_k are the lower and upper bounds of the k^{th} decision variable, respectively.

We select two values, 0.5 and 0.7, for F as the parameter pool. The details how we select them are shown in Section III-D. With three mutation strategies and two F values for each strategy, we finally get six mutation operators.

In contrast to other algorithms that use multiple operators, at each generation, all the three mutation strategies with randomly selected F value from the pool are used to produce new trial solutions. Three new trial solutions are generated for each subproblem at one time. Since the new trial solutions may be suitable for different subproblems, everyone is evaluated and used to update the neighborhood.

III. Experimental Results

A. Experimental Settings

We call the proposed MOEA/D with multiple DE mutation operators MOEA/D-MO. In this section we apply MOEA/D-MO to nine test instances (named F1-F9) with complicated PS shapes which are introduced in [6] and ten unconstraint test instances (named UF1-UF10) introduced in [19]. We compare MOEA/D-MO with MOEA/D-DE [6] which uses a single DE mutation operator to generate new trial solutions. MOEA/D-DE is proved to be efficient for solving MOPs. The parameter settings are as follows.

- The number of decision variables are $n = 30$ for all the test instances. Both algorithms stop after 150000 function evaluations (FES). The statistical results are based on 50 independent runs. The population size, i.e., the number of subproblems, is 300 for bi-objective problems and 595 for tri-objective problems. Both the two algorithms are implemented in Matlab and run in a same desktop computer.
- In MOEA/D-MO and MOEA/D-DE, the other parameters for MOEA/D framework are: $T = 20$, $p_n = 0.9$, $C = 2$. $\eta = 20$ and $p_m = 1/n$ in the polynomial mutation operator. $CR = 1.0$ and $F = 0.5$ in the DE operator of MOEA/D-DE. All these parameters are the same as in [6].

B. Performance Metrics

We use the inverted generational distance (IGD) [20] and hypervolume metric [21] to assess the performance of the algorithms in our experimental studies.

1) Inverted generational distance metric: Let P^* be a set of uniformly distributed points in the objective space along the PF and P be an approximation to the PF. The IGD from P^* to P is defined as

$$IGD(P^*, P) = \frac{\sum_{v \in P^*} d(v, P)}{P^*}$$

where $d(v, P)$ is the minimum Euclidean distance between v and any point in P and $|P^*|$ is the cardinality of P^*. If P^* is large enough to represent the PF very well, $IGD(P^*, P)$ measures both the diversity and convergence of P. To have a low value of $IGD(P^*, P)$, P must be very close to the PF and cannot miss any part of the whole PF.

2) Hypervalume indicator: As we have known the true PF, the hypervolume metric can be defined as $I_H^-(P, z^*) = I_H(P^*, z^*) - I_H(P, z^*)$. $I_H^-(P, z^*)$ can measure both the diversity and convergence of a set. To have a small value of $I_H^-(P, z^*)$, the approximation set P must be as close to the true PF and diverse as possible.

In our experiments, 500 evenly distributed points in PF are generated as the P^* for bi-objective problems and 990 points for tri-objective problems. z is set $(10, 10)^T$ for 2-objective instances and $(10, 10, 10)^T$ for 3-objective instances.

C. Comparison Results and Analysis

We compare MOEA/D-MO and MOEA/D-DE on F1-F9 and UF1-UF10. The statistical results of the IGD and hypervolume metrics are based on the final PF approximations obtained by the two algorithms over 50 runs. Tables I and II show the statistics based on the two metrics on F1-F9 respectively. The average IGD values versus FES for these two algorithms on F1-F9 are plotted in Fig. 2. And Fig. 3 plots all the 50 final approximations and populations obtained by MOEA/D-MO on F1-F9. Tables III and IV show the statistics based on the two metrics on UF1-UF10 respectively.

Table I shows that MOEA/D-MO performs better than MOEA/D-DE on five test instances, worse than MOEA/D-DE on three test instances and has similar performance with MOEA/D-DE on one test instance. From table II, we can see that MOEA/D-MO wins on five of the nine test instances. Overall MOEA/D-MO has better performance than MOEA/D-DE on F1-F9.

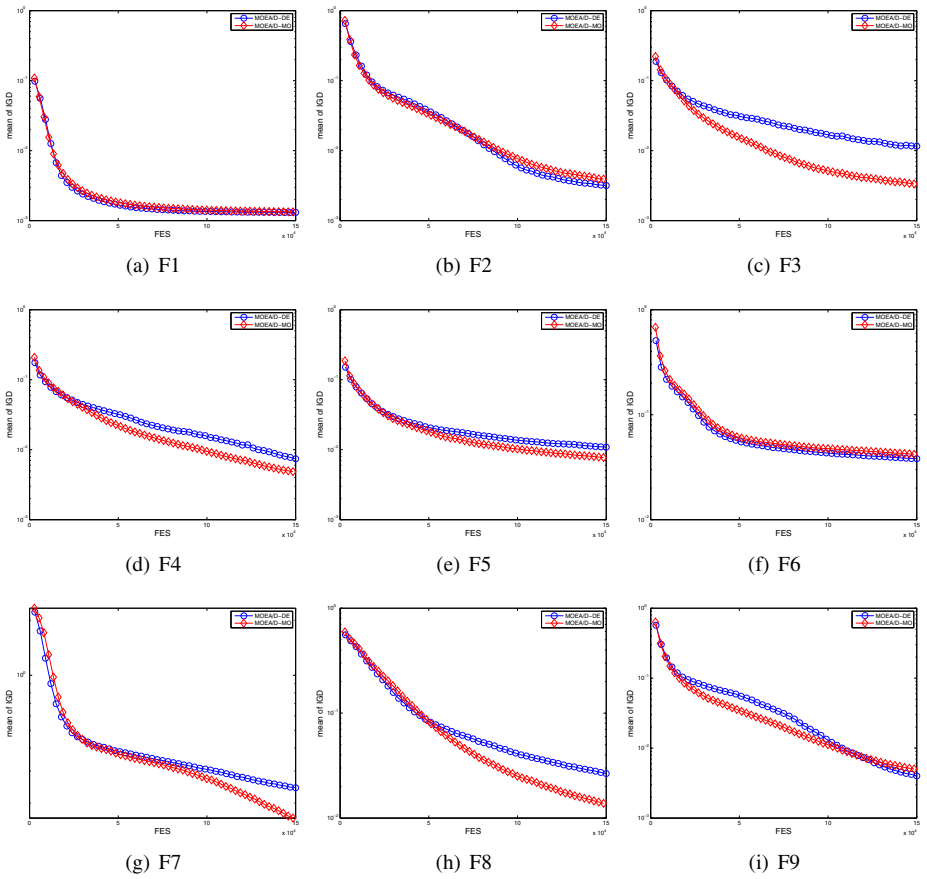
Fig. 2 shows the run time performance. It is clear that convergence speeds of the two algorithms are more or less similar on F1, F2, F6 and F9, but MOEA/D-MO beats MOEA/D-DE on F3-F5, F7 and F8.

In Fig. 3, all 50 final approximations and populations obtained by MOEA/D-MO on F1-F9 are shown. We can see that F1-F5 are 2-objective instances and they have the same convex PF shapes while their PS shapes are various nonlinear curves. F2-F5 have much more complex PS shapes than F1 does. From tables I and II, we can find that MOEA/D-MO is good at solving complex problems since it outperforms MOEA/D-DE on F3-F5 considering both IGD and I_H. Similar conclusion can be made on F7 and F8 which have many local Pareto solutions. So MOEA/D-MO is also good at solving problems with local optimal solutions. Therefore we can conclude that a single operator method works well on some simple problems while multiple operators method work well on complex problems with some difficulties. The reason might be that each operator in the multiple operators method contributes to different conditions during the search.

We also compare the two algorithms on ten unconstrained test instances (UF1-UF10). From Table III, we can see that MOEA/D-MO performs better than MOEA/D-DE on six test instances, worse than MOEA/D-DE on three test instances and has similar performance with MOEA/D-DE on one test instances. Table IV indicates that MOEA/D-MO wins on eight of the ten test instances. Overall, our proposed MOEA/D-MO is better than the compared MOEA/D-DE on UF1-UF10.

D. Parameter Pool of F for Three DE Strategies

To tune parameter F to get a parameter pool, we set $F = 0.1, 0.3, 0.5, 0.7, 0.9$ with a step of 0.2 for each of the three mutation strategies (M1, M2 and M3) on nine test instances with distinct complexity. Fig. 4 shows the results in the form of bar plot. There is tendency that operators with the middle F values can have better performance on most of the instances, although in some problems one operator may not be sensitive to the parameter F, to be specific, operators M2 and M3 on



(a) F1 (b) F2 (c) F3

(d) F4 (e) F5 (f) F6

(g) F7 (h) F8 (i) F9

Fig. 2. The mean IGD values versus numbers of fitness evaluations obtained by the two algorithms over 50 runs

TABLE I
IGD-METRIC VALUES OF THE APPROXIMATIONS FOUND BY MOEA/D-DE AND MOEA/D-MO ON F1-F9 OVER 50 RUNS

instance	MOEA/D-DE		MOEA/D-MO	
	mean	std.	mean	std.
F1	**0.0013**	0.0000	**0.0013**	0.0000
F2	**0.0032**	0.0005	0.0039	0.0006
F3	0.0116	0.0198	**0.0033**	0.0009
F4	0.0074	0.0087	**0.0048**	0.0014
F5	0.0109	0.0031	**0.0077**	0.0013
F6	**0.0381**	0.0065	0.0418	0.0073
F7	0.2426	0.0868	**0.1620**	0.1014
F8	0.0265	0.0271	**0.0137**	0.0131
F9	**0.0040**	0.0013	0.0049	0.0012

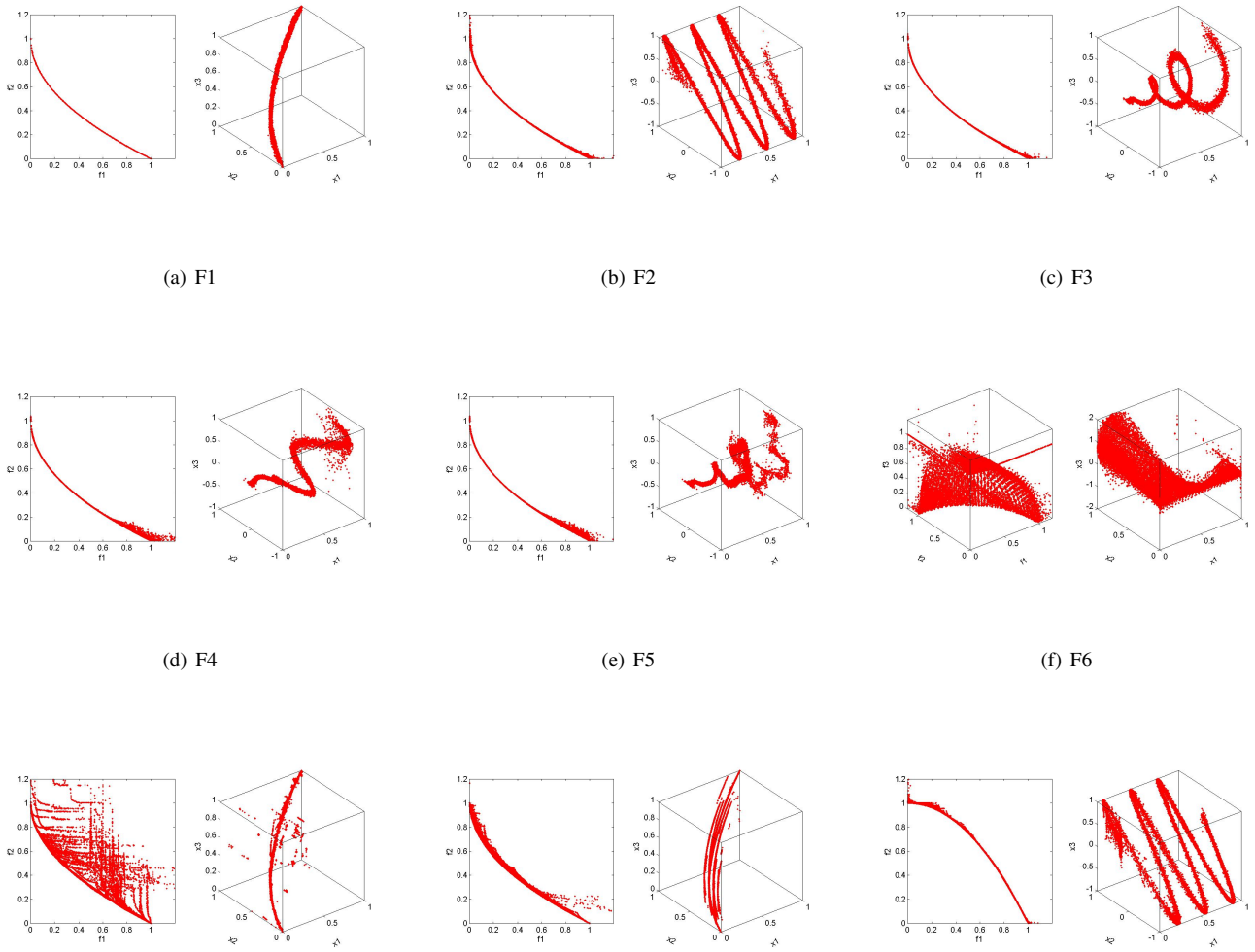
TABLE II
HYPERVOLUME-METRIC VALUES OF THE APPROXIMATIONS FOUND BY MOEA/D-DE AND MOEA/D-MO ON F1-F9 OVER 50 RUNS

instance	MOEA/D-DE		MOEA/D-MO	
	mean	std.	mean	std.
F1	**0.0010**	0.0000	0.0011	0.0000
F2	**0.1313**	0.0668	0.1669	0.0856
F3	0.3093	0.6613	**0.1102**	0.0547
F4	0.5011	0.7547	**0.1047**	0.0696
F5	0.2992	0.2375	**0.2753**	0.1683
F6	**0.5036**	0.1039	0.5075	0.1051
F7	5.1954	2.7963	**2.3148**	2.6628
F8	0.7023	0.8817	**0.1672**	0.4762
F9	**0.0839**	0.0474	0.1073	0.0635

instances F2-F5. We can observe that 0.5 and 0.7 work well on most instances, while the other values perform badly with some mutation strategies on certain instances. So $F = 0.5, 0.7$ are selected for all the mutation strategies in our proposed algorithm. And we can find that these six combinations have differences on almost all the instances, especially on F1, F6, F7 and F8. We expect that they can complement each other and improve the search capability. Here we want to mention that the strategies may not have to share the same parameter pool, i.e., each strategy can have its own parameter pool and pool size. In this paper, for similarity, we use the same parameter pool.

E. Contributions of Each DE Operator

We compare MOEA/D-MO with six algorithms, each of which uses one DE operator to produce offsprings. The results on F1-F9 over 50 runs are shown in Table V where 'M1-0.5'



(a) F1 (b) F2 (c) F3

(d) F4 (e) F5 (f) F6

(g) F7 (h) F8 (i) F9

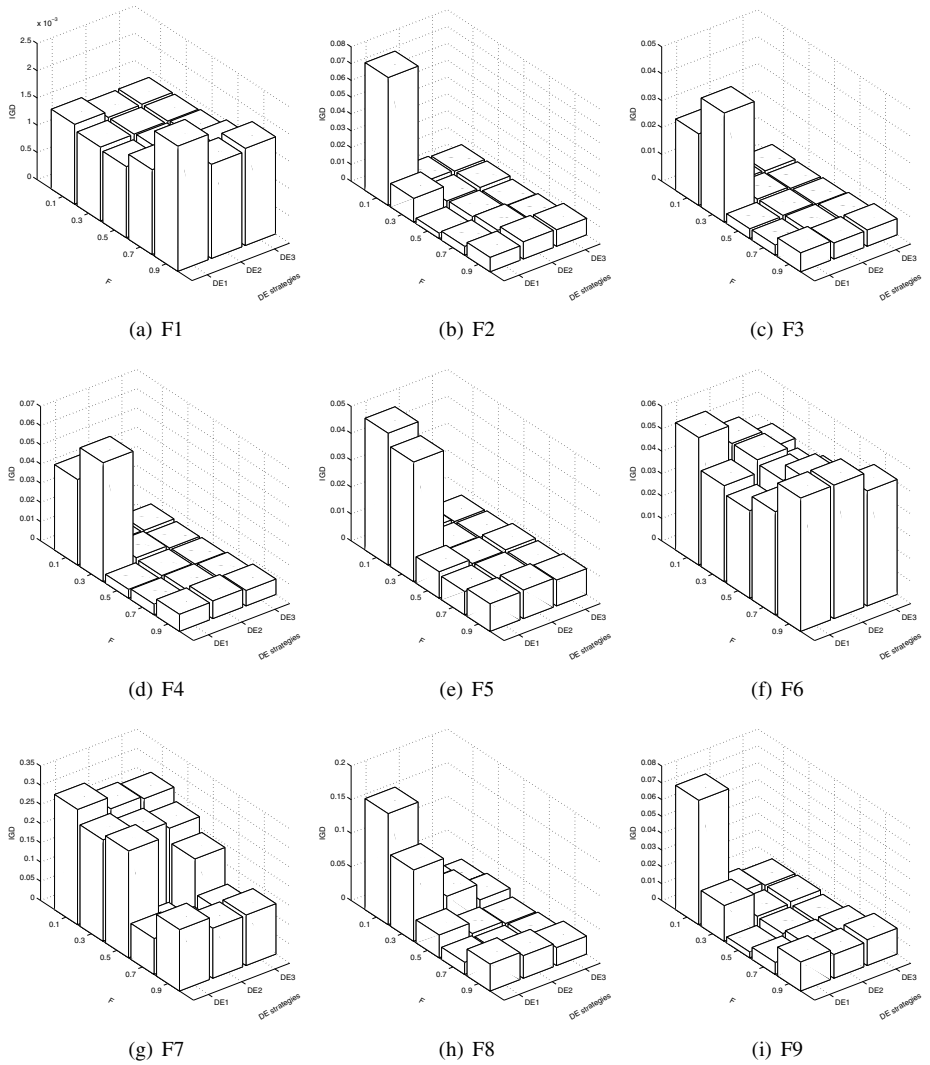
Fig. 3. All the final PF and PS approximations obtained by MOEA/D-MO on F1-F9 over 50 runs

TABLE III
IGD-METRIC VALUES OF THE APPROXIMATIONS FOUND BY MOEA/D-DE AND MOEA/D-MO ON UF1-UF10 OVER 50 RUNS

instance	MOEA/D-DE		MOEA/D-MO	
	mean	std.	mean	std.
UF1	**0.0020**	0.0002	0.0024	0.0002
UF2	0.0094	0.0118	**0.0052**	0.0006
UF3	0.0110	0.0127	**0.0049**	0.0048
UF4	0.0602	0.0051	**0.0533**	0.0032
UF5	0.2761	0.0527	**0.2636**	0.0512
UF6	0.1898	0.1664	**0.1392**	0.1657
UF7	**0.0028**	0.0013	**0.0028**	0.0003
UF8	**0.0755**	0.0125	0.0764	0.0114
UF9	0.1086	0.0572	**0.0723**	0.0413
UF10	**0.5301**	0.0727	0.5760	0.0865

TABLE IV
HYPERVOLUME-METRIC VALUES OF THE APPROXIMATIONS FOUND BY MOEA/D-DE AND MOEA/D-MO ON UF1-UF10 OVER 50 RUNS

instance	MOEA/D-DE		MOEA/D-MO	
	mean	std.	mean	std.
UF1	**0.0972**	0.0428	0.1561	0.0753
UF2	0.3671	0.5056	**0.1854**	0.1536
UF3	0.9791	1.0418	**0.3279**	0.6699
UF4	1.0181	0.1848	**0.8803**	0.1203
UF5	5.8243	1.8983	**4.1520**	1.3710
UF6	2.6626	2.3506	**2.3319**	2.5473
UF7	0.1764	0.2923	**0.1279**	0.0793
UF8	**0.9932**	0.2149	1.0357	0.2066
UF9	3.6362	1.6181	**2.9451**	1.2509
UF10	111.4497	24.8251	**106.6439**	18.5378



(a) F1 (b) F2 (c) F3

(d) F4 (e) F5 (f) F6

(g) F7 (h) F8 (i) F9

Fig. 4. Bar plot of three DE mutation strategies on five F values (0.1, 0.3, 0.5, 0.7, 0.9) over 10 runs

means the algorithm uses strategy M1 with F=0.5, and so on. Table V also presents the ranks of the seven algorithms on each instance and the overall ranks of the algorithms according to their average rank values on all the instances. We can see from Table V that each DE operator has distinct performance on different instances and overall our algorithm performs the best, ranking top 3 on most of the instances.

IV. Conclusion

This paper proposed a method of combining multiple DE mutation operators to improve the reproduction procedure in the MOEA/D framework. The main idea behind the combining method is to make use of different operators with distinct advantages. In the proposed reproduction procedure, three DE mutation strategies are selected, and the parameter F for each strategy is tuned to obtain a parameter pool. At each generation, for each subproblem of MOEA/D, all the three mutation strategies with randomly selected parameter are used to generate new trial solutions. After the new trial solutions are repaired and applied polynomial mutation, they are evaluated and used to update the neighborhood of the subproblem. The proposed MOEA/D-MO was compared with MOEA/D-DE. For a fair comparison, the maximal number of fitness evaluations is used as the stopping criteria. Further more, the details how the parameter pool is generated were given and we compared MOEA/D-MO with each single operators.

Experimental results indicate that MOEA/D-MO outperforms MOEA/D-DE on most of the test instances with different complexities and characteristics. MOEA/D-MO shows promising results for solving instances with complicated PSs. Moreover, it works well on instances which have local Pereto solutions. We can conclude that different operators are suitable for different kinds of MOPs and the proposed combining method can make multiple operators perform together and improve the search capability.

The work in this paper is preliminary and much work can be considered in the future: (1) designing efficient operators, besides DE mutation strategies, for MOPs, (2) introducing

TABLE V
IGD-METRIC VALUES OF THE APPROXIMATIONS FOUND BY SEVEN ALGORITHMS ON F1-F9 OVER 50 RUNS AND THE RANKS

instance	M1-0.5	M1-0.7	M2-0.5	M2-0.7	M3-0.5	M3-0.7	MOEA/D-MO
	mean(rank)	mean(rank)	mean(rank)	mean(rank)	mean(rank)	mean(rank)	mean(rank)
F1	**0.0013**(1)	0.0016(7)	0.0014(3)	0.0015(5)	0.0014(4)	0.0015(6)	0.0013(2)
F2	**0.0032**(1)	0.0051(5)	0.0050(4)	0.0070(7)	0.0046(3)	0.0060(6)	0.0039(2)
F3	0.0116(7)	0.0039(3)	0.0040(5)	0.0045(6)	**0.0033**(1)	0.0040(4)	0.0033(2)
F4	0.0074(7)	0.0055(5)	0.0050(3)	0.0065(6)	**0.0042**(1)	0.0053(4)	0.0048(2)
F5	0.0109(7)	0.0084(3)	0.0084(2)	0.0088(6)	0.0084(4)	0.0086(5)	**0.0077**(1)
F6	**0.0381**(1)	0.0464(6)	0.0462(5)	0.0506(7)	0.0391(2)	0.0438(4)	0.0418(3)
F7	0.2426(7)	**0.0654**(1)	0.0907(3)	0.0699(2)	0.1682(6)	0.1103(4)	0.1620(5)
F8	0.0265(7)	0.0157(6)	0.0142(5)	0.0139(4)	**0.0116**(1)	0.0129(2)	0.0137(3)
F9	**0.0040**(1)	0.0070(4)	0.0069(3)	0.0095(7)	0.0071(5)	0.0095(6)	0.0049(2)
rank statistics	4.3333(4)	4.4444(5)	3.6667(3)	5.5556(7)	3.0000(2)	4.5556(6)	**2.4444**(1)

stable and effective mechanism to balance the operators where better operators can contribute more to the search, not with equal contributions.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (No.61273313).

REFERENCES

[1] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the pareto archived evolution strategy," *Evolutionary computation*, vol. 8, no. 2, pp. 149–172, 2000.

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

[3] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design, Optimisation, and Control*, 2002, pp. 95–100.

[4] A. Zhou, B.-Y. Qu, H. Li, S.-Z. 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.

[5] Z. Qingfu and L. Hui, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[6] L. Hui and Z. Qingfu, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[7] A. Zhou, Q. Zhang, and G. Zhang, "A multiobjective evolutionary algorithm based on decomposition and probability model," in *2012 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2012, pp. 1–8.

[8] Y. Tan and Y. Jiao, "MOEA/D with uniform design for solving multiobjective knapsack problems," *Journal of Computers*, vol. 8, no. 2, pp. 302–307, 2013.

[9] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[10] A. G. Carvalho and A. F. Araujo, "Improving nsga-ii with an adaptive mutation operator," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. ACM, 2009, pp. 2697–2700.

[11] A. Kafafy, A. Bounekkar, and S. Bonnevay, "A hybrid evolutionary metaheuristics (hemh) applied on 0/1 multiobjective knapsack problems," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 2011, pp. 497–504.

[12] 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.

[13] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Multi-operator based evolutionary algorithms for solving constrained optimization problems," *Computers & Operations Research*, vol. 38, no. 12, pp. 1877–1896, 2011.

[14] K. Miettinen, *Nonlinear multiobjective optimization*. Springer, 1999, vol. 12.

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

[16] 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.

[17] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *2005 IEEE Congress on Evolutionary Computation (CEC)*, vol. 2. IEEE, 2005, pp. 1785–1791.

[18] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.

[19] 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," *University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report*, 2008.

[20] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[21] C. M. Fonseca, J. D. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," in *Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, vol. 216, 2005, p. 240.