Multi-objective Optimization Using Self-adaptive Differential Evolution Algorithm

V. L. Huang, S. Z. Zhao, R. Mallipeddi and P. N. Suganthan

Abstract — **In this paper, we propose a Multiobjective Self-adaptive Differential Evolution algorithm with objective-wise learning strategies (OW-MOSaDE) to solve numerical optimization problems with multiple conflicting objectives. The proposed approach learns suitable crossover parameter values and mutation strategies for each objective separately in a multi-objective optimization problem. The performance of the proposed OW-MOSaDE algorithm is evaluated on a suit of 13 benchmark problems provided for the CEC2009 MOEA Special Session and Competition (http://www3.ntu.edu.sg/home/epnsugan/) on Performance Assessment of Constrained / Bound Constrained Multi-Objective Optimization Algorithms.**

I. Introduction

MANY real world problems can be formulated as optimization problems with multiple objectives. Since the first attempt to solve multi-objective optimization problems by using evolutionary algorithms [12], multi-objective evolutionary algorithms (MOEAs) have been much researched and are widely used to solve numerous applications [13][14] in recent years. MOEAs benefit from the evolutionary algorithm's ability to generate a set of solutions concurrently in a single run thereby yielding several trade-off solutions.

Differential Evolution (DE) [8] is one of the most commonly used EAs. It is a simple and powerful population-based stochastic direct search method for solving numerical optimization problems in continuous search space. In DE, one certain trial vector generation strategy is required to be pre-specified with its parameters being tuned via a time-consuming trial and error scheme. Recently, we have developed a Self-adaptive Differential Evolution (SaDE) algorithm, in which both trial vector generation strategies and their associated parameters can be automatically adapted according to their previous experience of generating superior or inferior offspring than parent. Accordingly, as the search proceeds, different strategies with their associated parameters can be learned and applied to efficiently evolve the population at different stages. The SaDE algorithm has demonstrated promising performance when solving different types of optimization problems [1][2][3][5].

In this work, we improve the MOSaDE algorithm [5] with objective-wise learning strategies (called as OW-MOSaDE) to solve problems with multiple conflicting objectives and evaluate the performance on the 13 test problems [15]. The original MOSaDE learns just one set of parameters for all the objectives. In MOPs, different objective functions may possess different properties. Hence, it can be beneficial to learn one set of parameters for each objective as proposed in the OW-MOSaDE.

V. L. Huang, S. Zhao, R. Mallipeddi and P. N. Suganthan are with School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Ave., 639798 Singapore (emails: huangling@pmail.ntu.edu.sg, epnsugan@ntu.edu.sg)

II. Multi-objective differential evolution

The Multi-objective Optimization Problem (MOP) can be defined as:

Minimize/Maximize

$$\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x}))$$

Subject to $$\mathbf{G}(\mathbf{x}) = (g_1(\mathbf{x}), g_2(\mathbf{x}), \dots, g_j(\mathbf{x})) \geq 0$$

where $$\mathbf{x} = (x_1, x_2, \dots x_n)$$

and $\mathbf{x}$ is the decision vector, $\mathbf{F(x)}$ is the objective vector, and the constraints $g(\mathbf{x}) \geq 0$ determine the feasible region.

There already exists some work to adapt the control parameters of DE when solving the multi-objective opimization problems (MOPs). Abbass et al. introduced a Pareto-frontier Differential Evolution algorithm (PDE) to solve MOPs by incorporating Pareto dominance[3]. Later the first author self-adapted the crossover rate of PDE, by encoding the crossover rate into each individual and simultaneously evolving with other parameters. The scaling factor F was generated for each variable from a Gaussian distribution $N(0, 1)$ [7]. Zaharie proposed a parameter adaptation for DE (ADE) based on the concept of controlling the population diversity [10]. Following the same ideas, Zaharie and Petcu designed an adaptive Pareto DE algorithm for solving MOPs and analyzed its parallel implementation [11]. Xue et al. used a fuzzy logic controller (FLC) to dynamically adjust the parameters of multi-objective differential evolution [9].

III. Objective-wise Learning in Multi-objective SaDE

The earlier Multi-objective SaDE algorithm (MOSaDE) [5] is an extension of our recently developed SaDE [3] to optimize problems with multiple objectives. Similar to SaDE, the MOSaDE algorithm automatically adapts the trial vector generation strategies and their associated parameters according to their previous experience of generating superior

or inferior offspring. When extending the single-objective SaDE algorithm to the multi-objective domain, the criterion used to determine if the offspring is superior or inferior to its parent must be changed. However, in the original MOSaDE algorithm, trial vector A is better than target vector B, if (1) individual A dominates B, or (2) individual A and individual B are non-dominated with each other, but A is less crowded than individual B. Therefore, in case that the trial vector is better than the target vector according to this criterion, the associated parameter and strategy are recorded as done in SaDE. In our proposed objective-wise MOSaDE (OW-MOSaDE), we have *m+1* number of vectors, each corresponding to one objective, to store the CR parameter value and strategy probability values that improve the particular objective, the other one vector is the non-dominance-related criteria used in the original MOSaDE [5]. For example, in a 3 objective problem, if a particular set of parameter and strategy values can improve objectives 1 and 2 then the vectors corresponding to objectives 1 and 2 are updated by the new parameter and strategy values. The two strategies incorporated into our proposed OW-MOSaDE algorithm are:

DE/rand/1/bin:

$$u_{i,j} = \begin{cases} x_{i,j} + F \cdot \left(x_{r_1,j} - x_{r_2,j} \right) & \text{if } rand[0,1) < CR \text{ or } j = j_{rand} \\ x_{i,j} & \text{otherwise} \end{cases}$$

DE/rand/2/bin:

$$u_{i,j} = \begin{cases} x_{i,j} + F \cdot \left(x_{r_3,j} - x_{r_4,j} \right) + F \cdot \left(x_{r_5,j} - x_{r_6,j} \right) & \text{if } rand[0,1) < CR \text{ or } j = j_{rand} \\ x_{i,j} & \text{otherwise} \end{cases}$$

The indices r_1, r_2, r_3, r_4, r_5, r_6 are mutually exclusive integers randomly generated within the range [1, *NP*], these indices are randomly generated once for each mutant vector. These mutation strategies have been commonly used when solving MOPs [4][6][7][10][11].

The main steps of OW-MOSaDE are described below:

Step 1. Randomly initialize a population of *NP* individuals. Initialize *n_obj* number of vectors each corresponding to one objective. Each vector contains a strategy probability (p_k, $k=1,\ldots,K$, K is the no. of available strategies), the median value of CR(CRm_k) for each strategy, learning period ($LP=50$) .

Step 2. Evaluate the individuals in the population, and fill the external archive with these individuals.

Step 3. For each objective, we have a set of parameters as p_k and CR. In the initialization stage, we randomly select one of the *m+1* sets of parameters to be used to generate the trial vector population. After selecting a set of parameter values associated with the chosen objective, we perform the following optimization loop:

(1) Assign trial vector generation strategy and parameters to each target vector X_i

(a) Use stochastic universal sampling to select one strategy k for each target vector X_i

(b) Assign control parameters F and CR

F: Generate normally distributed F values with linearly reducing mean value from 1.0 to 0.05 and standard deviation 0.1.

CR: Generate the CR values under normal distribution of mean CRm_k and standard deviation 0.1.

After all the sets of parameters assignment,

(2) Generate a new population where each trial vector $\mathbf{U}_i^k$ is generated according to the associated trial vector generation strategy k and parameter F and CR.

(3) Selection:

FOR *i=1:NP*

(a) Evaluate the trial vector $\mathbf{U}_i^k$, and compare with the target vector $\mathbf{X}_{n(i)}$ nearest to $\mathbf{U}_i^k$ in the solution space.

IF $\mathbf{X}_{n(i)}$ dominates $\mathbf{U}_i^k$, discard $\mathbf{U}_i^k$.

ELSE

IF $\mathbf{U}_i^k$ dominates $\mathbf{X}_{n(i)}$, replace $\mathbf{X}_{n(i)}$ with $\mathbf{U}_i^k$;

IF non-dominated with each other, randomly choose one to be the new target vector;

$\mathbf{U}_i^k$ will enter the external archive if (i) $\mathbf{U}_i^k$ dominates some individual(s) of the archive (the dominated individuals in the archive are deleted); or (ii) $\mathbf{U}_i^k$ is non-dominated with archived individuals

END IF

(b) We compare $\mathbf{U}_i^k$ to $\mathbf{X}_{n(i)}$ in each objective to update the *m+1* sets of parameters if offspring is better than parent. In case that $\mathbf{U}_i^k$ is better than $\mathbf{X}_{n(i)}$ in a particle objective, we record the associated parameter CR and flag strategy k corresponding to the objective as successful objective-wise vector. Otherwise, flag the vector as failed one. Calculate strategy probability p_k which is the percentage of the success rate of trial vectors generated by each strategy during the learning period for each objective separately. Updating of *CR:* After the first *LP* generation, calculate CRm_k according to the recorded CR values.

(c) When the external archive exceeds the maximum specified size, we select the less crowded individuals based on harmonic average distance [4] to maintain the archive size.

END FOR

Step 4. While one of the termination conditions is not satisfied, go to Step 3.

IV. Experimental Results

We evaluate the performance of the proposed objective-wise MOSaDE algorithm (OW-MOSaDE) and the original MOSaDE [5] on a set of 13 benchmark functions [15], which include 7 two-objective test functions, 3 three-objective test functions and 3 five-objective test functions. The experiments are designed according to [15] and the experimental results are presented in the following as required by [15]. We use the same four learning strategies

introduced in section III in both algorithms.

PC Configuration:

Windows XP Professional. Intel Pentium® 4 CPU 3.00 GHZ 2 GB of memory.Language: MATLAB 7.1

Parameters Setting:

The population size is set at 50. The maximum external archive size is set at 100, 150 and 800 for 2, 3 and 5 objectives respectively, as specified in [15]. The maximum FES is set as 3e+5.

Results Achieved:

For all the test functions, we recorded the approximate set every generation in each run. According to these approximate sets, we calculated the performance metrics IGD [15]. The smallest, the largest, the mean and the standard deviation of the IGD values obtained for each test instance of the 30 runs are presented in Table I for OW-MOSaDE and the mean of the IGD values obtained by original MOSaDE is also shown in the table. Smaller IGD values indicate better results. For the OW-MOSaDE, the plot of the final approximation set with the smallest IGD value in the objective space for some test instances with 2 and 3 objectives are shown in Figs. 1-10.

From IGD values shown in Table I, among all the 13 test problems, OW-MOSaDE performs better than the original MOSaDE on all the 13 benchmark test functions. It demonstrates that OW-MOSaDE improves upon the search ability of the original MOSaDE by using the new objective-wise learning strategy and non-domination-wise learning strategy instead of only the non-domination-wise learning strategy when solving these multi-objective test problems.

V. Conclusion

In this paper, we enhanced the multi-objective self-adaptive differential evolution algorithm with objective-wise learning of parameters and mutation strategies. We assign an individual set of parameters the most appropriate to for optimizing each objective. The performance of our approach was evaluated on the test benchmark functions from CEC2009 special session on Performance Assessment of Constrained / Bound Constrained Multi-Objective Optimization Algorithms.

ACKNOWLEDGEMENT

Authors acknowledge the financial support offered by the A*Star (Agency for Science, Technology and Research) under the grant # 052 101 0020 to conduct this research.

References

[1] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *IEEE Congress on Evolutionary Computation (CEC 2005)* Edinburgh, Scotland, IEEE Press, Sept. 2005, pp. 1785-1791.

[2] V. L. Huang, A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for constrained real-parameter optimization", in *2006 IEEE Congress on Evolutionary Computation (CEC'06)*, Vancouver, BC, Canada, July 2006.

[3] A. K. Qin, V. L. Huang and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimisation," *IEEE Trans on Evolutionary Computation,* DOI: 10.1109/TEVC.2008.927706, 2009.

[4] V. L. Huang, P. N. Suganthan, A. K. Qin and S. Baskar, "Multiobjective differential evolution with external archive and harmonic distance-based diversity measure", Technical Report, Nanyang Technological University, 2005.

[5] V. L. Huang, A, K. Qin, P. N. Suganthan and M. F. Tasgetiren, "Multi-objective Optimization based on Self-adaptive Differential Evolution Algorithm", *P. IEEE-CEC*-07, Sept. 2007, Singapore.

[6] H. A. Abbass, R. Sarker and C. Newton. "PDE: A Pareto-frontier differential evolution approach for multiobjective optimization problems," in *Proc. of IEEE Congress on Evolutionary Computation*, pp. 971-978. 2001.

[7] H. A. Abbass, "The self-adaptive Pareto differential evolution algorithm," in *Proc. of Congress on Evolutionary Computation* Vol.1, IEEE Press, 831-836. 2002.

[8] R. Storn and K. V. Price, "Differential evolution-A simple and efficient heuristic for global optimization over continuous Spaces," *Journal of Global Optimization*, vol.11, pp.341-359. 1997.

[9] F. Xue, A. C. Sanderson, P. P. Bonissone and R. J. Graves, "Fuzzy logic controlled multi-objective differential evolution," *The 14th IEEE International Conference on Fuzzy Systems*, 2005:720-725.

[10] D. Zaharie, "Control of population diversity and adaptation in differential evolution algorithms," in *R. Matousek, P. Osmera (eds.), Proc. of Mendel 2003, 9th International Conference on Soft Computing*, Brno, Czech Republic, June 2003, pp. 41-46.

[11] D. Zaharie and D. Petcu "Adaptive pareto differential evolution and its parallelization," in *Proc. of 5th International Conference on Parallel Processing and Applied Mathematics*, Czestochowa, Poland, Sept. 2003, pp. 261-268.

[12] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," In J. J. Grefenstette (Ed.), *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Pittsburgh, PA, pp. 93–100,1985.

[13] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Chischester, U.K.: Wiley, 2001.

[14] C. A. C. Coello, D. A. V. Veldhuizen and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York, March 2002.

[15] Qingfu Zhang, Aimin Zhou, S. Z. Zhao, P. N. Suganthan, Wudong Liu and Santosh Tiwari, "Multiobjective Optimization Test Instances for the CEC2009 Special Session and Competition," Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report, University of Essex, Colchester, UK and Nanyang Technological University, Singapore, 2008.

Table I: The IGD value of the 30 final approximation sets obtained for each test problem

Fun	OW-MOSaDE				Original-MOSaDE
	Smallest (IGD)	Largest (IGD)	Mean (IGD)	Std (IGD)	Mean (IGD)
1. UF1	0.0103	0.0152	0.0122	0.0012	0.0983
2 UF2	0.0056	0.0135	0.0081	0.0023	0.0607
3. UF3	0.0727	0.1512	0.1030	0.0190	0.3248
4. UF4	0.0483	0.0562	0.0513	0.0019	0.0977
5. UF5	0.3991	0.4766	0.4303	0.0174	0.6963
6. UF6	0.0926	0.2271	0.1918	0.0290	0.3640
7. UF7	0.0208	0.1576	0.0585	0.0291	0.1916
8. UF8	0.0800	0.1199	0.0945	0.0119	0.4019
9. UF9	0.0574	0.1621	0.0983	0.0244	0.3984
10. UF10	0.5777	0.9402	0.7430	0.0885	2.9313
11. UF11	0.3360	0.4724	0.3951	0.0384	0.5450
12. UF12	593.3990	812.4774	734.5680	65.0659	783.2456
13. UF13	2.8298	4.5500	3.2573	0.3521	3.6881

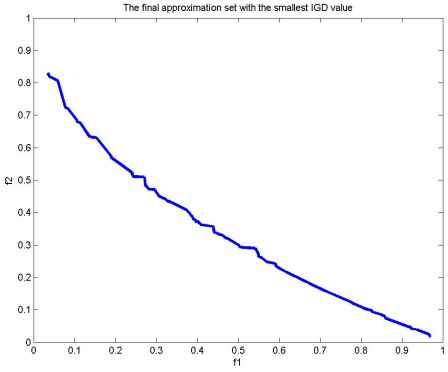

FIGURE 1. The plot of the final approximation set with the smallest IGD value in the objective space for each test instances UF1.

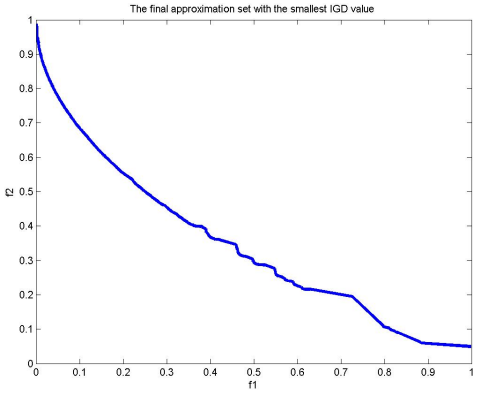

FIGURE 2. The plot of the final approximation set with the smallest IGD value in the objective space for each test instances UF2.

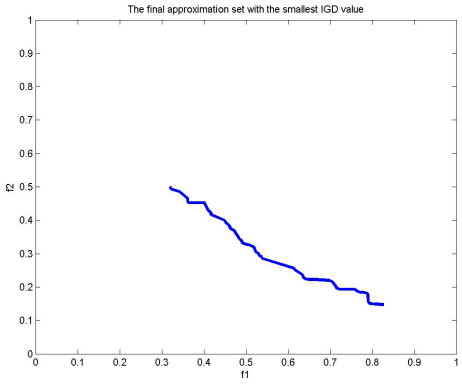

FIGURE 3. The plot of the final approximation set with the smallest IGD value in the objective space for each test instances UF3.

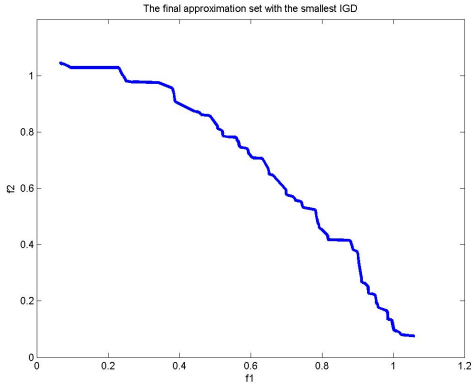

FIGURE 4. The plot of the final approximation set with the smallest IGD value in the objective space for each test instances UF4.

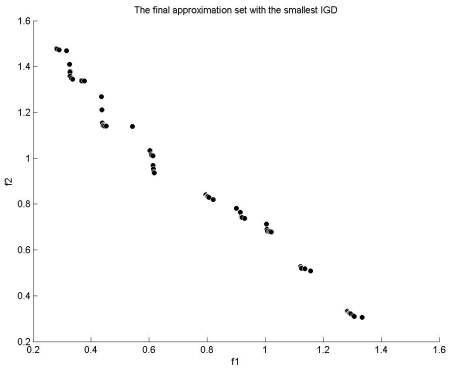

FIGURE 5. The plot of the final approximation set with the smallest IGD value in the objective space for each test instances UF5.

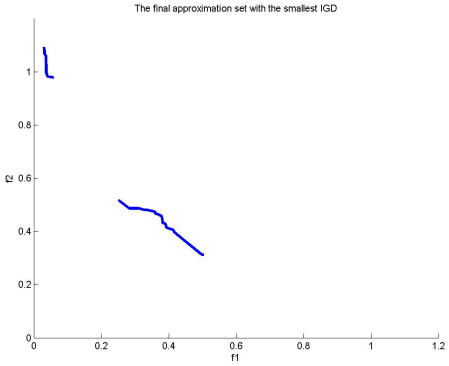

FIGURE 6. The plot of the final approximation set with the smallest IGD value in the objective space for each test instances UF6.

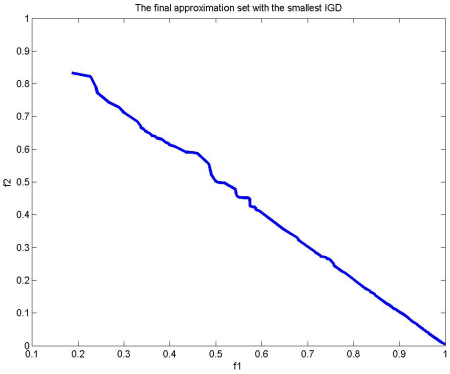

FIGURE 7. The plot of the final approximation set with the smallest IGD value in the objective space for each test instances UF7.

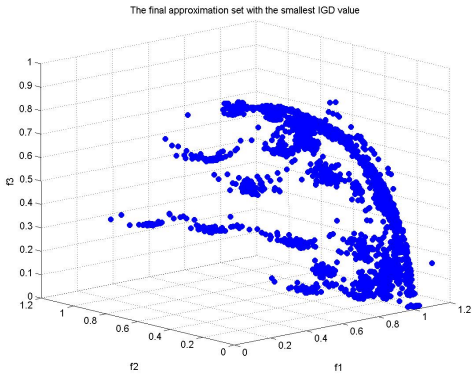

FIGURE 8. The plot of the final approximation set with the smallest IGD value in the objective space for each test instances UF8.

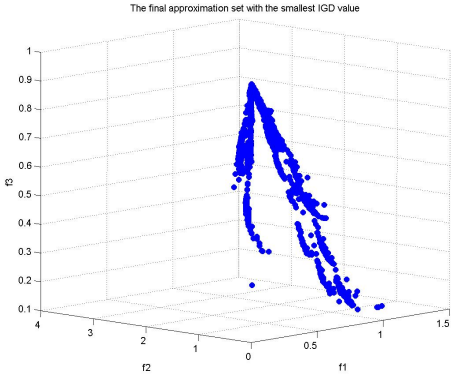

FIGURE 9. The plot of the final approximation set with the smallest IGD value in the objective space for each test instances UF9.

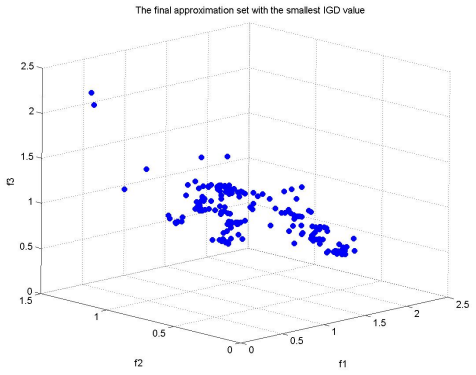

FIGURE 10. The plot of the final approximation set with the smallest IGD value in the objective space for each test instances UF10.