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

A Novel Differential Evolution (DE) Algorithm for Multi-objective Optimization

Xin Qiu
NUS Graduate School for
Integrative Sciences and Engineering
National University of Singapore
119077, Singapore
Email: qiuxin@nus.edu.sg

Jianxin Xu
Department of Electrical
and Computer Engineering
National University of Singapore
119077, Singapore
Email: elexujx@nus.edu.sg

Kay Chen Tan
Department of Electrical
and Computer Engineering
National University of Singapore
119077, Singapore
Email: eletankc@nus.edu.sg

***Abstract*—Convergence speed and parametric sensitivity are two issues that tend to be neglected when extending Differential Evolution (DE) for multi-objective optimization. To fill in this gap, we propose a multi-objective DE variant with an extraordinary mutation strategy and unfixed parameters. Wise tradeoff between convergence and diversity is achieved via the novel cross-generation mutation operators. In addition, a dynamic mechanism enables the parameters to evolve continuously during the optimization process. Empirical results show that the proposed algorithm is powerful in handling multi-objective problems.**

I. Introduction

Differential Evolution (DE), proposed by Storn and Price [1], is arguably one of the most powerful evolutionary algorithms for single objective numerical optimization. DE inherently has many advantages over most Evolutionary Algorithms (EA) [2], and it generally produces good performance in different types of optimization problems. Due to this success, its use has been extended into multi-objective optimization (MO) in recent years.

When a DE operator is employed for multi-objective problems, the original crossover and mutation strategy of DE need to be altered as the solution set consists of more than one solution. Unlike single-objective optimization, the target of multi-objective optimization problems is to find a set of evenly distributed solutions in Pareto Front. Thus, two core problems that have been studied by the researchers who have extended DE into MO are diversity maintenance and survival selection criterion. Based on these two design aspects, some non-Pareto-based approaches and Pareto-based approaches have been proposed.

Chang et al. [3] constitute the first systematical attempt to extend DE in MO. In their paper, DE/rand/1/bin is utilized with a Pareto optimal set, which is an external archive to store the non-dominated solutions during the search process. Fitness sharing is also incorporated into their approach in order to maintain the diversity of the whole population. Babu and Jehan [4] proposed the Differential Evolution for Multi-Objective Optimization approach. One objective function is incorporated as an additional constraint in their algorithm, and an aggregation function is utilized. Li and Zhang [5] figured out an MO Differential Evolution based on decomposition for continuous MO problems with variable linkages. Kukkonen and Lampinen [6] [7] modified basic DE algorithm for MO problems, and named the framework Generalized Differential Evolution (GDE). The survival selection between the parent and the offspring is based on the Pareto Dominance and the constraints of the problems are also handled with Pareto Dominance in the constraint space. In their later work [8] [9], two new versions of GDE were proposed to achieve a better diversity of the final population and to overcome the slow converging speed. Other MO-DEs where Pareto Dominance works as the criterion for survival selection include [10] [11] [12] [13]. Another branch of the Pareto-based algorithms encompass approaches that introduce a Pareto ranking procedure into DE. Representative methods include Pareto-Based Differential Evolution (PBDE) algorithm [14], Nondominated Sorting Differential Evolution (NSDE) [15] and Differential Evolution for Multi-Objective Optimization (DEMO) [16]. In these algorithms, a $(\mu + \lambda)$-selection is implemented after a set of trial vectors have been generated from the current population. One of the more recent works that gave an outstanding performance should be the MOEA/D-DE and NSGA-II-DE proposed in [17]. In that paper, a neighborhood selection mechanism is utilized to help enhance the diversity maintenance ability of MOEA/D-DE, which outperforms NSGA-II-DE in most benchmark problems with complicated Pareto sets.

Based on the above review, most existing works on MO-DE focus on designing a diversity maintenance mechanism and / or a survival selection procedure. However, there are still several other problems remaining while extending DE into multi-objective optimization: 1. The performance of DE is sensitive to parametric setting, which would be a thorny issue when multiple problems need to be handled simultaneously. 2. While much effort has been paid to maintain the diversity of the population, a satisfactory converging speed cannot be guaranteed. 3. During applications, different multi-objective optimization frameworks may be needed depending on the nature of the problem. However, most existing MO-DEs are inconvenient to implement on different MO frameworks because they already define the whole evolutionary process instead of only providing a way to reproduce offspring.

To address the above issues, this paper presents a new

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

multi-objective Differential Evolution algorithm with a novel mutation strategy, namely Cross-Generation mutation and a dynamic parametric tuning mechanism (CGDE). The proposed DE variant utilizes the information between different generations to help predict the correct searching direction for each solution so that not only the converging speed could be increased but also the diversity of the whole population is enhanced. Meanwhile, the dynamic parameters of the proposed algorithm would be tuned automatically during the optimization process similar to an evolving variable. Moreover, CGDE is portable, and can be easily implemented onto different MO frameworks. The experimental results demonstrate that CGDE is powerful in handling Multi-objective Optimization Problems (MOPs).

The remainder of this paper is organized as follows. Section II gives a brief introduction about the structure of basic DE. Section III provides the detailed algorithmic description of the proposed approach. Section IV studies the performance of the algorithm on benchmark problems with comparison to several state-of-the-art MOEAs. Finally, Section V concludes this paper and highlights some potential future research directions.

II. Classical DE Algorithm

A. Initialization

The first step of DE is the initialization of the population of NP and D over the search space, where NP denotes the population size and D denotes the variable dimensions. Now we symbolize each individual by $X_{i,g} = [x^1_{i,g}, x^2_{i,g}, \ldots, x^D_{i,g}]$, where $i = 1, 2, \ldots, NP, g = 0, 1, \ldots, G_{max}$ and G_{max} denotes the maximum number of generations. Also, let us define the lower search bound as $X_{min} = [x^1_{min}, x^2_{min}, \ldots, x^D_{min}]$ and the upper search bound as $X_{max} = [x^1_{max}, x^2_{max}, \ldots, x^D_{max}]$. Then, the initial value of the ith individual is generated as below:

$$x^j_{i,0} = x^j_{min} + rand(0,1) \cdot (x^j_{max} - x^j_{min}), j = 1, 2, 3, \ldots, D \tag{1}$$

B. Mutation

In this step, each individual will generate a new individual, called the mutant vector $v_{i,g}$. The most frequently used mutation strategies are listed below:

1) "DE/rand/1"

$$v_{i,g} = x_{r_1,g} + F \cdot (x_{r_2,g} - x_{r_3,g}) \tag{2}$$

2) "DE/best/1"

$$v_{i,g} = x_{best,g} + F \cdot (x_{r_1,g} - x_{r_2,g}) \tag{3}$$

3) "DE/current-to-best/1"

$$v_{i,g} = x_{i,g} + F \cdot (x_{best,g} - x_{i,g}) + F \cdot (x_{r_1,g} - x_{r_2,g}) \tag{4}$$

where $x_{best,g}$ denotes the individual with the best fitness value in current generation. The indices r_1, r_2, r_3 are randomly selected integers from $[1, 2, 3, \ldots, NP]$ that are distinct from i and mutually different. $F \in [0, 1]$ is a real parameter, called the scaling factor.

C. Crossover

After mutation, crossover operation is employed to generate the trial vectors. During the crossover, the mutant vectors are recombined with the original members of the current population, called the target vectors, to form the trial vectors. Two basic crossover schemes of DE are the exponential recombination and the binomial recombination. Binomial recombination is mostly used in DE literature [18].

Binomial recombination is performed on each variable and it could be outlined as below:

$$u^j_{i,g} = \begin{cases} v^j_{i,g} \ if \ rand_{i,j}[0,1] \leq Cr \ or \ j = j_{rand} \\ x^j_{i,g} \ otherwise \end{cases} \tag{5}$$

where $j_{rand} \in [1, 2, 3, \ldots, D]$ is a randomly selected index to ensure that trial vector could get at least one component from the mutant vector. Cr is called the crossover probability.

D. Selection

Selection is the last step to generate the population of next generation. The process of selection is to determine whether the target vector or the trial vector survives to the next generation according to their fitness value. The selection operation in DE is described below:

$$X_{i,g+1} = \begin{cases} U_{i,g} \ if \ f(U_{i,g}) \ \leq f(X_{i,g}) \\ X_{i,g} \ if \ f(X_{i,g}) < f(U_{i,g}) \end{cases} \tag{6}$$

where $f(X)$ is the fitness function to be minimized.

III. Proposed Algorithm

A. Cross-Generation Mutation

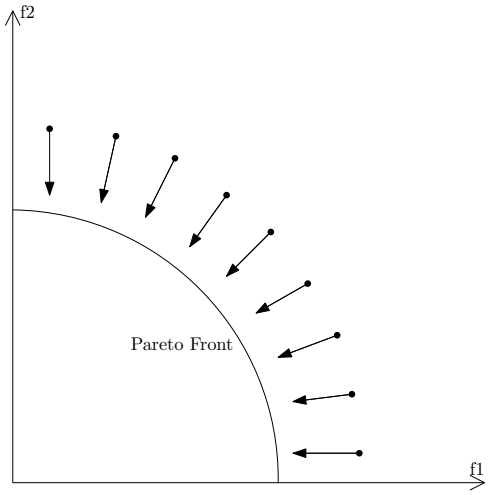

Figure 1. This figure shows the different converging directions of each solution in objective space. The circle line draws the Pareto Optimal Front of the 2-objective minimization problem. Each dot represents a solution and the arrows denotes their convergence directions.

1) Converging direction and searching direction: Here, convergence direction is defined as the direction that approaches the Pareto Optimal Front given a solution in the objective space whereas searching direction is referred to the

moving direction of an individual in the decision space. It is intuitive that the best searching direction of an individual in decision space should be the mapping of its corresponding convergence direction in objective space. Thus, it would be beneficial if the correct converging direction could be obtained during the optimization process. Although the exact converging direction is unavailable, it is possible to estimate the convergence direction using the information across generations. Considering the nature of evolutionary algorithms, the solutions are getting better and better while the number of evolving generations increase. The changing of solutions may provide some hints for estimating the converging directions. As can be seen in Fig. 1, the converging direction varies for the solutions from different regions in objective space. To obtain a reasonable result, estimation of converging direction should be derived within different regions separately. Defining neighborhood for each individual is an intuitive way to extract the information for certain limited area in objective space. In the following subsections, a simple yet novel way to define the neighborhood will be discussed.

2) Neighborhood based on subrank: In multi-objective optimization, one common way to define the neighborhood of a solution in objective space is to measure the Euclidean distance between two solutions based on the objective values. One shortcoming of this method is that for some problems, the domain of each objective is different, which would make it biased if the distance is calculated only based on the objective values. Under these circumstances, normalization is a necessary pre-processing step. Nevertheless, it cannot be guaranteed that the true domains of each problem are known, especially in some real life applications. Normalization becomes difficult without knowledge of true domains. To circumvent this problem, a new term is defined: subrank, which is similar to the ranking in [19]. Unlike the rank in non-dominated sorting [20], subrank is a vector comprising the ranks of an individual in each objective. For instance, in a 2-objective problem, if the subrank of one individual is $[1, 1]$, it means this individual has the best fitness value for both objectives. In the proposed algorithm, subrank would replace fitness value to measure the distance between two individuals in objective space. In this way, normalization could be skipped and the bias caused by various domains would be eliminated. Similarly, the Euclidean distance in terms of subrank is calculated between every two individuals. Subsequently, the neighborhood of each individual will be decided based on the calculated distance and the predefined neighborhood size N. The N nearest solutions are marked as the neighborhood of this individual.

3) Neighborhood-based Cross-Generation Mutation: Conventional DE mutation operators only makes use of the information within the current generation to generate the mutant vector. However, based on the above discussions, the information across generations may reveal the trend of how solutions moved in the searching space, and in turn help to guide search directions. In order to estimate the converging directions for different regions in objective space and utilize the corresponding searching directions to guide the evolution process, a novel neighborhood-based cross-generation mutation strategy is proposed. The mutant vector is generated based on the differentiation of two individuals from different generations.

In the proposed strategy, one individual will generate one mutant vector, and the individual is called the parent of this mutant vector. One interesting feature of the proposed mutation strategy is that the parent itself is not involved in the mutation process. Instead, the mutant vector is generated from the two neighborhood pools of the parent. One neighborhood pool is formed by N individuals selected from the population of current generation, another consists of N individuals picked from the population of previous generation. More specifically, given a parent, first the distance between this parent and all the other individuals of current generation would be calculated based on subrank, then the N nearest individuals are marked to become the neighborhood of this parent in current generation. Similarly, for the same parent, the distance from it to all the individuals of previous generation based on subrank is computed, and the N nearest individuals are recorded as the parent's neighborhood of previous generation. With these two neighborhood pools, the new mutation operation can be conducted following the formula below:

$$v_{i,g} = x_{rn_1,g} + F \cdot (x_{rn_1,g} - x_{rn_2,g-1}) \tag{7}$$

where $v_{i,g}$ denotes the new generated individual, called the mutant vector, and i is the index of parent, g is the number of current generation. $x_{rn_1,g}$ is an individual randomly selected from the parent's neighborhood in current generation, where index rn_1 is a randomly selected integer from $[1, 2, 3, \ldots, N]$ and N is the pre-defined neighborhood size. $x_{rn_2,g-1}$ is an individual randomly selected from the parent's neighborhood in previous generation, where index rn_2 is a randomly selected integer from $[1, 2, 3, \ldots, N]$.

The underlying rationale of the designed strategy is that the movement of the solutions during evolution process may assist to guide the subsequent search direction. Following the proposed formula, $x_{rn_1,g}$ stays in the current generation while $x_{rn_2,g-1}$ comes from the previous generation, so taking into account that both are selected from the neighborhood pools of the same parent, the difference between them may reveal the moving trend of the solutions near the parent in objective space. Since the essence of evolutionary algorithm is to evolve solutions, the better solutions would have a higher chance to survive to next generation and the poor individuals will be discarded. Thus, the moving directions of the solutions may be close to the true converging directions. Adding the differentiation of $x_{rn_1,g}$ and $x_{rn_2,g-1}$ as a perturbation to $x_{rn_1,g}$ is like to make the current solution keep exploring following the searching direction corresponding to the estimated converging direction.

4) Population-based variant: One common issue with neighborhood-based mutation strategies is that the searching step may become excessively small because of the gradually converged population. Since the individuals involved in mu-

tation are selected from neighborhood of the same parent, the differentiation of them should not be too large, which will make the new generated individual stay near the parent. If the population come to converge, the explorative ability of the algorithm would be impaired. To make a reasonable tradeoff between exploitative ability and explorative ability of the search, another variant of the cross-generation mutation based on population would be utilized simultaneously. Below shows the details of the operation:

$$v_{i,g} = x_{i,g} + F \cdot (x_{rp_1,g} - x_{rp_2,g-1}) \tag{8}$$

where $v_{i,g}$ denotes the new generated mutant vector, and i is the index of parent, g is the number of current generation. $x_{i,g}$ is the parent itself. $x_{rp_1,g}$ is an individual randomly selected from the whole population in current generation, where index rp_1 is a randomly selected integer from $[1, 2, 3, \ldots, NP]$ and NP is the population size. $x_{rp_2,g-1}$ is an individual randomly selected from the whole population of the previous generation, where index rp_2 is a randomly selected integer from $[1, 2, 3, \ldots, NP]$.

Unlike the neighborhood-based variant, the differentiation part in population-based variant is formed with two individuals randomly selected from the whole population, which is similar to the traditional DE mutation strategy. However, because the two individuals come from two different generations, the scale of their differentiation may vary larger than the original DE mutation strategy. In this way, the explorative ability could be enhanced in some sense. Another special modification of the population-based variant is that the differentiation would be added to the parent directly. By this means, the searching will continue centering on the parent instead of performing a purely stochastic exploration. This could lead to a more efficient optimization process.

In the current version of CGDE, the above two variants are employed in a half-half manner, which means they have the same probability (50%) to be utilized during each mutation operation. Simulation results demonstrate that the performance of the algorithm becomes better than the versions with only one of them in most testing problems. A final note about the new mutation strategy is that each individual in the current generation would be a parent of a mutant vector, and after the above mutation operation, each mutant vector still needs to go through a binomial recombination (crossover operation) with the parent to generate the final offspring.

5) Enhancing both the diversity and convergence: While designing a multi-objective optimization algorithm, it is difficult to hasten the convergence speed without loss of diversity as if they were conflicting to each other. Nonetheless, the underlying mechanism enables the proposed approach to enhance convergence along with diversity. In the neighborhood-based variant, the convergence is sped up by guiding the searching direction with information across generations. Meanwhile, due to the neighborhood-based selection mode, the final offspring will stay around the parent in objective space so that the diversity of the population could be consistently maintained. Analogously, the parent-centric search in population-based variant helps preserve the diversity, and the more explorative mutation operator contributes to a higher converging speed to true Pareto Front.

B. Dynamic Parameters

1) Parametric Sensitivity in multi-objective optimization: Sensitivity to the parametric setting is a critical issue during the extension of DE into multi-objective optimization. The performance of DE may vary tremendously with different settings of parameters, especially with the scaling factor F. This problem becomes more challenging in multi-objective optimization. Each objective may have different requirements for the parametric setup, and even for a certain objective, the optimal setup of parameters may vary during different searching stages, e.g., a large searching step is needed at the start of exploration whereas a relatively smaller searching scope is preferred near the end of the search. Hence, a dynamic parameter tuning mechanism is proposed to help the algorithm adapt its parameters automatically. Two targets are going to be achieved with the new tuning method: first is that there should be a selection pressure to reserve more proper parameters and discard poor parameters; second is that as the current fitting parameters may not conform to the requirement of the next search stage, the value of parameters should not be fixed. In other words, convergence of parameters need to be avoided. In the following subsections, the details of the proposed dynamic mechanism for tuning F (Scaling Factor) and Cr (Crossover Rate) will be discussed.

2) Self-Tuning F (Scaling Factor): As shown in the mutation formula of DE, Scaling Factor F has a significant effect on the searching step of the algorithm, therefore the performance of DE depends heavily on the selection of F. In multi-objective optimization, each individual is responsible for searching different regions in objective space. Concerning that different objective might have various favored parametric settings, the selection of F should not be identical for every individual. To realize it, each individual will be independently assigned a value randomly generated from a given interval during initialization, and this value would be inherited by the children of each individual. After that, in order to avert convergence of F, a perturbation will be added to the current F value of each survived solution after every generation. The scale of the perturbation is randomly generated from another pre-defined interval, and it is conducted separately for each individual. To summarize, now F is similar to an additional variable for each solution, and it would be inherited by the offspring. The survival mechanism in multi-objective optimization framework will gradually evolve the value of F to make it more suitable for current problem.

3) Stochastic Cr (Crossover Rate): Crossover rate decides how different the final offspring is from the parent, so the choice of Cr mainly affects the convergence speed of the optimization without too much influence on the searching ability of the algorithm. A simple stochastic selection of Cr value is applied here. The value of Cr is randomly generated

Table I
MEAN AND STANDARD DEVIATION OF THE IGD VALUES (30 RUNS)

Problems	CGDE-NSGA-II	NSGA-II-DE	NSGA-II(SBX)	MOEA/D-DE	MOEA/D(SBX)	CCPSO	MOEGS	SPEA2
	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
UF1	0.0536	0.0603	0.1230	**0.0475**	0.1568	0.0483	0.2207	0.1341
	(0.0156)	(0.0162)	(0.0318)	**(0.0372)**	(0.0652)	(0.0108)	(0.1025)	(0.0407)
UF2	**0.0233**	0.0429	0.0481	0.0426	0.0640	0.0481	0.0484	0.0626
	(0.0027)	(0.0047)	(0.0125)	(0.0316)	(0.0310)	(0.0079)	(0.0091)	(0.0071)
UF3	**0.0970**	0.1515	0.2179	0.1513	0.3064	0.3024	0.1318	0.3025
	(0.0168)	(0.0271)	(0.0666)	(0.0688)	(0.0300)	(0.0390)	(0.0370)	(0.0410)
UF4	**0.0409**	0.0723	0.0533	0.0866	0.0560	0.0639	0.1277	0.0682
	(**0.0003**)	(0.0078)	(0.0018)	(0.0104)	(0.0034)	(0.0068)	(0.0105)	(0.0031)
UF5	**0.2839**	0.8494	0.3257	0.7643	0.4318	0.4535	0.8727	0.4741
	(0.0785)	(0.1698)	(0.0943)	(0.1307)	(0.0812)	(0.0734)	(0.5201)	(0.0877)
UF6	**0.1700**	0.4181	0.2302	0.4386	0.4374	0.4701	0.6323	0.4609
	(0.1034)	(0.0819)	(0.0680)	(0.2206)	(0.1509)	(0.0356)	(0.4519)	(0.1292)
UF7	**0.0243**	0.0389	0.2359	0.1018	0.3536	0.0961	0.0833	0.1693
	(**0.0042**)	(0.0422)	(0.1447)	(0.1648)	(0.1552)	(0.0381)	(0.1027)	(0.1285)
WFG1	**0.8784**	1.2181	1.0790	1.1634	1.0483	0.9758	1.1818	1.0859
	(0.0166)	(0.0052)	(0.0813)	(0.0138)	(0.0458)	(0.0359)	(0.0183)	(0.0185)
WFG2	**0.0135**	0.0461	0.1604	0.1666	0.1871	0.5447	0.5816	0.2381
	(0.0005)	(0.0253)	(0.0277)	(0.0880)	(0.0643)	(0.1463)	(0.1444)	(0.0281)
WFG3	**0.0201**	0.0344	0.0211	0.0204	0.0203	0.1826	0.2399	0.1036
	(0.0010)	(0.0017)	(0.0016)	(0.0018)	(0.0059)	(0.0521)	(0.0558)	(0.0305)
WFG4	0.0215	0.0927	0.0189	0.0811	**0.0167**	0.0614	0.1842	0.0363
	(0.0022)	(0.0037)	(0.0011)	(0.0081)	**(0.0015)**	(0.0328)	(0.0342)	(0.0052)
WFG5	**0.0682**	0.0754	0.0705	0.0692	0.0691	0.0899	0.1698	0.0734
	(0.0009)	(0.0017)	(0.0005)	(0.0003)	(0.0006)	(0.0134)	(0.1090)	(0.0012)
WFG6	0.1141	0.1079	**0.0640**	0.1072	0.0820	0.1228	0.1335	0.0867
	(0.0276)	(0.0349)	**(0.0068)**	(0.0319)	(0.0237)	(0.0489)	(0.0373)	(0.0162)
WFG7	**0.0168**	0.0306	0.0170	0.0190	0.0205	0.2122	0.2250	0.0507
	(0.0011)	(0.0019)	(0.0010)	(0.0011)	(0.0111)	(0.0730)	(0.1229)	(0.0184)
WFG8	**0.1089**	0.1413	0.1371	0.1271	0.1270	0.2203	0.2502	0.1700
	(0.0036)	(0.0101)	(0.0065)	(0.0128)	(0.0097)	(0.0489)	(0.0337)	(0.0107)
WFG9	0.1260	0.1106	0.0844	**0.0597**	0.0606	0.1519	0.2660	0.1070
	(0.0003)	(0.0380)	(0.0529)	**(0.0287)**	(0.0381)	(0.0534)	(0.1514)	(0.0386)

from a pre-defined interval for each crossover operation. From the experimental results, a stochastic Cr could make the algorithm perform more consistently in face of different types of problems.

IV. EXPERIMENTAL RESULTS

A. Implementation

The proposed CGDE is implemented into nondominated sorting genetic algorithm II (NSGA-II) [20], which is a well-known powerful multi-objective optimization algorithm. The offspring in NSGA-II would be generated via CGDE, and other parts of the original framework is reserved including fast-non-dominated-sort and density estimation. The population size NP is set to 100 for 2-objective problems. The neighborhood size is selected as 5. The initialization interval for F is $[0.2, 0.9]$, and the lower bound of F during evolution is 0.2 while the upper bound is 0.9. The interval for perturbation added to F is $[-0.2, +0.2]$.

B. Benchmark Problems

In total, 16 frequently-used two-objective benchmark problems were utilized to evaluate the performance of the algorithm, among which UF1 to UF7 are unconstrained problems from CEC 2009 Special Session and Competition [21] and WFG1 to WFG9 are from WFG test suites [22]. Numerous types of problems are covered in terms of separability, modality, bias and shape of Pareto Optimal Front, and all of them are minimization problems.

C. Comparison with State-of-the-art MOEAs

The performance of CGDE-NSGA-II is compared with seven state-of-the-art Multi-objective Optimization Evolutionary Algorithms (MOEAs), namely, NSGA-II(SBX) [20], NSGA-II-DE [17], MOEA/D(SBX) [23], MOEA/D-DE [17], CCPSO [24], MOEGS [25], SPEA2 [26]. The simulation results of the above MOEAs have referred to [27]. The population size is fixed as 100 for all the testing algorithms, and the maximum number of function evaluations is set to 5×10^4. Setup of other parameters are identical as suggested in the original studies. Inverted Generational Distance (IGD) [28] is employed as the indicator to quantitatively evaluate the performance of each algorithm at the end of each run. All of the simulations were done on an Intel(R) Core(TM) i7 machine with 16-GB RAM and 3.40-GHz speed. Table I shows the mean and standard deviation of the IGD values for

30 independent runs of each algorithms on each benchmark. The best entries are marked in boldface.

From the comparative results, it clearly elucidates that CGDE-NSGA-II is powerful in tackling multi-objective problems when compared to its competitors: it achieves the best results in 12 out of 16 benchmark problems. To examine further, CGDE-NSGA-II outperforms original NSGA-II-DE in 14 out of 16 problems, and the latter cannot give the best performance in any of the benchmarks. The only disparity between them lies in the DE operator, from which we can draw the conclusion that the success of CGDE-NSGA-II should benefit from the newly developed cross-generation mutation operators as well as the dynamic parameters.

V. Conclusion

This paper has presented a new DE variant for multi-objective optimization, which employs information across generations to help guide the searching directions. Two variants of cross-generation mutation operators have been proposed to enhance both the convergence and diversity in the evolution. Moreover, the dynamic parameters would evolve automatically according to different problems and searching stages. Experimental results demonstrate that the proposed algorithm is able to outperform seven state-of-the-art MOEAs in most benchmark problems.

In future work, we would like to further study the optimal settings for neighborhood size and perturbation interval. In addition, to better evaluate the performance of CGDE, problems with more than two objectives are expected to be examined and implementation into other multi-objective frameworks will be considered.

Acknowledgment

This work was supported by the Singapore Ministry of Education Academic Research Fund Tier 1.

References

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

[2] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 1, pp. 4 –31, feb. 2011.

[3] C. S. Chang, D. Y. Xu, and H. B. Quek, "Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mass transit system," *IEEE Proceeding on Electric Power Applications*, vol. 146(5), pp. 577–583, 1999.

[4] B. V. Babu and M. M. L. Jehan, "Differential evolution for multi-objective optimization," *Proceedings of the 2003 Congress on Evolutionary Computation (CEC2003)*, vol. 4, pp. 2696–2703, 2003.

[5] H. Li and Q. Zhang, "A multiobjective differential evolution based on decomposition for multiobjective optimization with variable linkages," *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference*, vol. 4193, pp. 583–592, 2006.

[6] J. Lampinen, "DE's selection rule for multiobjective optimization," *Technical report, Lappeenranta University of Technology, Department of Information Technology*, 2001.

[7] S. Kukkonen and J. Lampinen, "Solving multiobjective optimization problems using an artificial immune system," *IASTED International Conference on Artificial Intelligence and Applications (AIA 2004)*, pp. 96–102, 2004.

[8] ——, "An extension of generalized differential evolution for multi-objective optimization with constraints," *Parallel Problem Solving from Nature - PPSN VIII*, vol. 3242, pp. 752–761, 2004.

[9] ——, "GDE3: The third evolution step of generalized differential evolution," *IEEE Congress on Evolutionary Computation (CEC'2005)*, vol. 1, pp. 443–450, 2005.

[10] H. A. Abbass, R. Sarker, and C. Newton, "PDE:a pareto-frontier differential evolution approach for multi-objective optimization problems," *Proceedings of the Congress on Evolutionary Computation 2001*, vol. 2, pp. 971–978, 2001.

[11] H. A. Abbass, "The self-adaptive pareto differential evolution algorithm," *Congress on Evolutionary Computation 2002*, vol. 1, pp. 831–836, 2002.

[12] ——, "A memetic pareto evolutionary approach to artificial neural networks," *The Australian Joint Conference on Artificial Intelligence*, vol. 2256, pp. 1–12, 2001.

[13] L. V. Santana-Quintero and C. A. C. Coello, "An algorithm based on differential evolution for multi-objective problems," *International Journal of Computational Intelligence Research*, vol. 1(2), pp. 151–169, 2005.

[14] N. K. Madavan, "Multiobjective optimization using apareto differential evolution approach," *Congress on Evolutionary Computation (CEC'2002)*, vol. 2, pp. 1145–1150, 2002.

[15] A. W. Iorio and X. Li, "Solving rotated multi-objective optimization problems using differential evolution." *Advances in Artificial Intelligence, Proceedings*, vol. 3339, pp. 861–872, 2004.

[16] T. Robic and B. Filipic, "DEMO: Differential evolution for multiobjective optimization," *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO*, vol. 3410, pp. 520–533, 2005.

[17] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 2, pp. 284–302, 2009.

[18] S. Das, A. Abraham, U. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 3, pp. 526 –553, june 2009.

[19] S. Kukkonen and J. Lampinen, "Ranking-dominance and many-objective optimization," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, Sept 2007, pp. 3983–3990.

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

[21] 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," *Technical Report*, 2008.

[22] L. Bradstreet, L. Barone, L. While, S. Huband, and P. Hingston, "Use of the WFG toolkit and PISA for comparison of MOEAs," in *Computational Intelligence in Multicriteria Decision Making, IEEE Symposium on*, 2007, pp. 382–389.

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

[24] C. Goh, K. Tan, D. Liu, and S. Chiam, "A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design," *European Journal of Operational Research*, vol. 202, no. 1, pp. 42–54, 2010.

[25] C. Goh, Y. Ong, K. Tan, and E. Teoh, "An investigation on evolutionary gradient search for multi-objective optimization," in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, 2008, pp. 3741–3746.

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

[27] S. B. Gee, X. Qiu, and K. C. Tan, "A novel diversity maintenance scheme for evolutionary multi-objective optimization," *Intelligent Data Engineering and Automated Learning IDEAL 2013, Lecture Notes in Computer Science*, vol. 8206, pp. 270–277, 2013.

[28] C. A. C. Coello and N. C. Cortes, "Solving multiobjective optimization problems using an artificial immune system," *Genetic Programming and Evolvable Machines*, vol. 6, no. 2, pp. 163–190, 2005.