Adaptive Cross-Generation Differential Evolution Operators for Multiobjective Optimization

Xin Qiu, Jian-Xin Xu, *Fellow, IEEE*, Kay Chen Tan, *Fellow, IEEE*, and Hussein A. Abbass, *Senior Member, IEEE*

***Abstract*—Convergence performance and parametric sensitivity are two issues that tend to be neglected when extending differential evolution (DE) to multiobjective optimization (MO). To fill this research gap, we develop two novel mutation operators and a new parameter adaptation mechanism. A multiobjective DE variant is obtained through integration of the proposed strategies. The main innovation of this paper is the simultaneous use of individuals across generations from an objective-based perspective. Good convergence–diversity tradeoff and satisfactory exploration–exploitation balance are achieved via the hybrid cross-generation mutation operation. Furthermore, the cross-generation adaptation mechanism enables the individuals to self-adapt their associated parameters not only optimization-stage-wise but also objective-space-wise. Empirical results indicate the statistical superiority of the proposed algorithm over several state-of-the-art evolutionary algorithms in handling MO problems.**

***Index Terms*—Differential evolution (DE), multiobjective optimization (MO), mutation operation, parameter adaptation.**

I. Introduction

DIFFERENTIAL evolution (DE), proposed by Storn and Price [1], is arguably one of the most efficient evolutionary algorithms (EAs) for numerical optimization. Due to its inherent superiority over many EAs [2], DE generally produces good performance in a wide variety of single-objective optimization problems [3]–[10]. This success triggered the popularity of extending DE to multiobjective optimization (MO) in recent years (see supplementary material for a detailed literature survey).

Manuscript received September 30, 2014; revised January 14, 2015; accepted April 9, 2015. Date of publication June 29, 2015; date of current version March 29, 2016. This work was supported by the Singapore Ministry of Education Academic Research Fund Tier 1 under Project R-263-000-A12-112.

X. Qiu is with the NUS Graduate School for Integrative Sciences and Engineering, National University of Singapore, Singapore 119077 (e-mail: qiuxin@nus.edu.sg).

J.-X. Xu and K. C. Tan are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119077 (e-mail: elexujx@nus.edu.sg; eletankc@nus.edu.sg).

H. A. Abbass is with the School of Engineering and Information Technology, University of New South Wales, Australian Defence Force Academy, Canberra, ACT 2600, Australia (e-mail: h.abbass@adfa.edu.au).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors. This material complements the study presented in the paper within the page restrictions of this journal. The material is 2460 KB in size.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2015.2433672

Intrinsically, the main task of any evolutionary MO (EMO) algorithm is to achieve best tradeoffs among multiple objectives. In practice, EMO algorithms operate by evolving a set of solutions to approximate the Pareto front (PF), which is defined by the ideally optimal tradeoffs [11]–[13]. How to distribute these solutions along the PF more evenly and more closely then becomes the core problem during the design of EMO algorithms [14]–[16]. When extending DE to MO, researchers focus more on maintaining diversity and developing survival selection criterions [17]. Although impressive progress has been reported, there are still several issues that tend to be neglected. First, the performance of DE is sensitive to the setup of its control parameters [2], [18]–[21]. It emerges as a thorny issue when multiple objectives are optimized simultaneously [22]–[24]. Second, while much effort has been paid to maintain the diversity of the population, a satisfactory convergence speed cannot be guaranteed [25]. Third, most existing MO–DEs are inconvenient to implement among different MO frameworks since they have already fixed the whole evolutionary process [26]–[30] or are tightly combined with one category of MO frameworks (e.g., Pareto dominance- or decomposition-based) [31]–[37]. During applications, varied types of MO frameworks are needed depending on the nature of the problem [22], [38]–[40]. It is desirable to design MO–DE in a more flexible way.

To address the above issues, this paper proposes a novel multiobjective DE algorithm utilizing information across generations. First, two new mutation strategies are developed to enhance both convergence speed and diversity maintenance. One is the neighborhood-based cross-generation (NCG) mutation, which tries to estimate the promising searching directions by analyzing the differences between consecutive generations. Another is the population-based cross-generation (PCG) mutation, in which populations from two generations are involved to facilitate a more explorative search. The two new mutation operators are employed in a hybrid manner in order to strike a delicate balance between exploitation and exploration. Subsequently, a new scheme called cross-generation adaptation (CGA) mechanism is introduced to tune the parameters automatically. The proposed CGA mechanism self-adapts the parameters not only in a dynamic way but also from the perspective of objective space. In each generation, the optimal parameters for a limited region in objective space are gauged via integrating the parameter values of consecutive generations. Based on the aforementioned algorithmic components, we develop an adaptive cross-generation DE (ACGDE)

algorithm for MO. There is no dedicated survival selection criterion in ACGDE, thereby leading to sufficient flexibility in implementation. To implement ACGDE into an MO framework, the main step is to replace the original reproduction operator (e.g., mutation and crossover operators) with ACGDE. Regeneration of more promising offspring is the principal role of ACGDE. To the best of the authors' knowledge, there is no previous study that focuses on examining the usability of information across generations. The experimental results validate the effectiveness of each algorithmic component. The implementation into two well-known MO frameworks {i.e., nondominated sorting genetic algorithm II (NSGA-II) [41] and multiobjective evolutionary algorithm based on decomposition (MOEA/D) [42]} demonstrates that ACGDE is very powerful and robust in handling MO problems (MOPs).

This paper extends our previous work presented in [43]. The remainder of this paper is organized as follows. Section II introduces the basic DE algorithmic structure and summarizes the existing research niches in related works. Section III describes the algorithmic details of the proposed approaches and discusses about their underlying rationales. Section IV studies the effectiveness of the proposed mechanisms through empirical tests. Performance of the new algorithm is evaluated by comparison with several state-of-the-art multiobjective EAs (MOEAs) on a wide variety of benchmark problems. Finally, Section V concludes this paper.

II. Background

A. Classical DE Algorithm

A classical DE algorithm [1] employs four main steps, namely initialization, mutation, crossover, and selection, during the optimization process. To save space, the algorithmic details of the four steps are outlined in order of their occurrence in the supplementary material. It is recommended to refer to this material for clearly understanding the definitions of all the symbols that may appear in the later sections.

B. Research Niches in Related Work

A thorough literature review on three-related research topics is conducted in the supplementary material (interested readers may go through for more details). Based on this survey, the current research niches for each topic are summarized below.

1) Differential Evolution Algorithms for Multiobjective Optimization: Being a powerful heuristic for numerical optimization, DE has been extended into MO by many researchers. Based on our literature review (see the supplementary material), most existing works on MO–DE focus on combining the traditional DE mutation strategies with diversity-based survival selection procedures, which are commonly used in other MOEAs. Development of new mutation schemes that are oriented exclusively toward MO is still a research niche. According to empirical studies conducted by Lim and Haron [44] and Mantere [45], mutation strategy plays a key role during the optimization process of DE. In this paper, two novel mutation strategies are specially designed to enhance both diversity and convergence in MO.

2) Direction-Guided Evolutionary Algorithms: Steering the optimization process in an effective and efficient way is desirable in the design of an EA. Among the numerous search schemes, direction-guided search has proved to be quite promising [46]. According to our survey (see the supplementary material), although additional archives are used in some direction-guided MOEAs, little attention is paid to the application of information across generations, which may imply the efficient searching directions. Another open problem with current direction-guided MOEAs is the insufficient analysis regarding objective space. Objective-based relationships among individuals are in fact critical in the estimation of converging directions (see Section III-B1 for definition). In the proposed NCG mutation, individuals from different generations are employed based on their relationships in objective space. Promising searching directions for specified solutions are then generated to guide the exploration.

3) Parameter Adaptation in Differential Evolution: The success of DE in solving a particular problem is very closely related to the fine-tuning of its control parameters. In the literature, a good volume of work has been undertaken to enhance the ultimate performance of DE via adaptive parameters (see the supplementary material for a detailed review). However, knowledge in objective space is neglected by most existing adaptation methods for MO–DE. The intrinsic difference between SO and MO is not revealed in the design of these schemes. To fill this research gap, a CGA mechanism is introduced to dynamically adapt the parameters from an objective-based perspective.

III. Proposed Algorithm

This section outlines the proposed ACGDE and discusses the details of each algorithmic component.

A. Overview of the Algorithm

This paper presents an adaptive DE algorithm for MO. Two novel mutation strategies and one new parameter adaptation mechanism are developed. The proposed ACGDE differs from the classical DE algorithms in the following aspects: first, information across different generations will be utilized to enhance both convergence and diversity; second, the relationships among individuals in objective space are considered during selection of parents; and third, adaptation of parameters operates from an objective-based perspective.

Algorithm 1 shows the general structure of ACGDE. Detailed explanations for the newly introduced mechanisms, parameters, and concepts will be provided in the following sections. Interested readers may also refer to the supplementary material for pseudocodes of each algorithmic component.

B. Cross-Generation Mutation

1) Converging and Searching Directions: To better explain the development of our idea, two kinds of directions are defined, namely converging and searching directions. Similar to convergence direction in [46], converging direction is defined as the direction that approaches the Pareto optimal front given a solution in the objective space, whereas searching

Algorithm 1 Procedure of ACGDE

Set the control parameters of ACGDE: population size N, Neighborhood size $T = 5\% \cdot N$, θ_F, θ_{Cr}, $F_{\max}$, $F_{\min}$, $Cr_{\max}$ and $Cr_{\min}$.
Set the generation number $g = 0$ and randomly initialize the population of N individuals $P_g = \{X_{1,g}, X_{2,g}, \ldots, X_{N,g}\}$ together with their associated parameters $\{F_{1,g}, F_{2,g}, \ldots, F_{N,g}\}$ and $\{Cr_{1,g}, Cr_{2,g}, \ldots, Cr_{N,g}\}$. The generated parameter values should be within $[F_{\min}, F_{\max}]$ and $[Cr_{\min}, Cr_{\max}]$, respectively.
for $g = 1$ to $G_{\max}$ **do**
  if $g = 1$ **then**
    $P_g = P_0$
  end if
  Calculate the sub-rank for each individual in P_g.
  for $i = 1$ to N **do**
    Determine the neighborhood of current generation and previous generation for individual $X_{i,g}$ in terms of sub-rank.
    Perform CGA mechanism to produce $F_{\text{spring},i,g}$ and $Cr_{\text{spring},i,g}$.
    if $rand[0, 1.0] \leq 0.5$ **then**
      Utilize NCG mutation with $F_{\text{spring},i,g}$ to generate the mutant vector $V_{i,g}$.
    else
      Utilize PCG mutation with $F_{\text{spring},i,g}$ to generate the mutant vector $V_{i,g}$.
    end if
    Perform binomial recombination with $Cr_{\text{spring},i,g}$ on $X_{i,g}$ and $V_{i,g}$ to generate the offspring $U_{i,g}$.
    Survival selection by the MO framework that ACGDE is implemented to.
  end for
  Generation number $g = g + 1$
  Update Current Population P_g
end for

direction is referred to the moving direction of an individual in the decision space. It is intuitive that the most efficient searching direction of an individual is the mapping of its converging direction. Thus, it would be beneficial if the correct converging directions could be learned during the optimization process. Although the exact converging directions are unavailable, it is possible to estimate the converging directions using the information across generations. Considering the nature of EAs, the solutions are getting better and better while the number of evolving generations increases. In MO, such selection pressure will move the population toward the true PF. The movement of solutions across generations may, therefore, provide some hints for estimating the converging directions. As shown in Fig. 1, the converging direction varies for the solutions from different regions in objective space. It is more reasonable to separately estimate converging directions within different regions. Defining neighborhood for each individual is an effective way to extract information over a certain limited area in objective space. In the following section, a simple yet efficient way to define the neighborhood will be discussed.

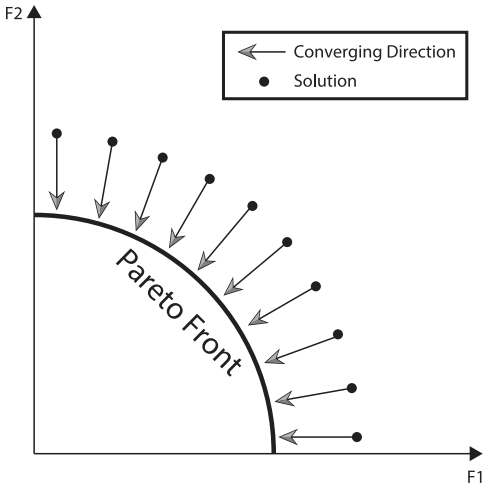


Fig. 1. Converging directions of different solutions in the objective space. The curve denotes the Pareto optimal front of the two-objective minimization problem. Each dot represents a solution and the arrows denote their converging directions.

2) Neighborhood Based on Sub-Rank: In MO, a common way to define the neighborhood of a solution in objective space is to measure the Euclidean distance between two solutions based on their objective values. The major weakness of this method is that the calculated distance may be biased by the different ranges of objective functions [47]–[49]. Under these circumstances, normalization is a necessary preprocessing step [50], [51]. Nevertheless, it cannot be guaranteed that the true range of each objective is known before optimization [47]. Normalization becomes difficult without the knowledge of true ranges. To circumvent this issue, a new term, called "sub-rank," is introduced. Similar to the ranking in [52], sub-rank is a vector comprising the ranks of an individual in each separate objective among current population. For instance, in a two-objective problem, if the sub-rank of one individual is $(1, 1)$, it means this individual has the best fitness value for both objectives. In the proposed algorithm, sub-rank 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 ranges is eliminated. In the proposed algorithm, first the Euclidean distance is calculated using sub-rank between any two individuals. Next, the neighborhood of each individual will be decided based on the calculated distance and the predefined neighborhood size T. The T nearest solutions are marked as the neighborhood of this individual.

3) Neighborhood-Based Cross-Generation Mutation: Classical DE mutation operators only make use of the information within the current generation to generate the mutant vectors [1], [53], [54]. However, based on the above discussions, the information across generations may reveal the trend of how solutions moved in the search space, and in turn help to guide searching directions. In order to estimate the converging directions for different regions in objective space and utilize

the corresponding searching directions to guide the evolutionary process, a novel NCG mutation strategy is proposed. The mutant vector is generated based on the difference between two individuals from consecutive generations.

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

$$V_{i,g} = X_{rn_1,g} + F \cdot \left(X_{rn_1,g} - X_{rn_2,g-1}\right) \tag{1}$$

where $V_{i,g}$ denotes the new generated individual, called the mutant vector, and i is the index of the main parent, g is the number of current generation. $X_{rn_1,g}$ is an individual randomly selected from the main parent's neighborhood of current generation, where index rn_1 is a randomly selected integer from $\{I_{i,g,1}, I_{i,g,2}, I_{i,g,3}, \ldots, I_{i,g,T}\}$, which records the indices of neighborhood, and T is the predefined neighborhood size. $X_{rn_2,g-1}$ is an individual randomly selected from the main parent's neighborhood of previous generation, where index rn_2 is a randomly selected integer from the recorded neighborhood indices set $\{I_{i,g-1,1}, I_{i,g-1,2}, I_{i,g-1,3}, \ldots, I_{i,g-1,T}\}$.

The underlying rationale of the designed strategy is that the movement of the solutions during evolutionary 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 main parent, the difference between them may reveal the moving trend of the solutions near the main parent in objective space. Since the essence of EA is to evolve solutions, the better solutions would have a higher chance to survive to next generation and the weaker individuals will be discarded. The moving directions of the solutions may be close to the true converging directions. Adding the difference between $X_{rn_1,g}$ and $X_{rn_2,g-1}$ as a perturbation to $X_{rn_1,g}$ is equivalent to making the current solution keep exploring based on the searching direction corresponding to the estimated converging direction. It is notable that proper selection of the neighborhood size T is critical to the overall performance of the algorithm. One extreme case is to set T as 1, then the algorithm may get stuck because of the overlap of $X_{rn_1,g}$ and $X_{rn_2,g-1}$. Conversely,

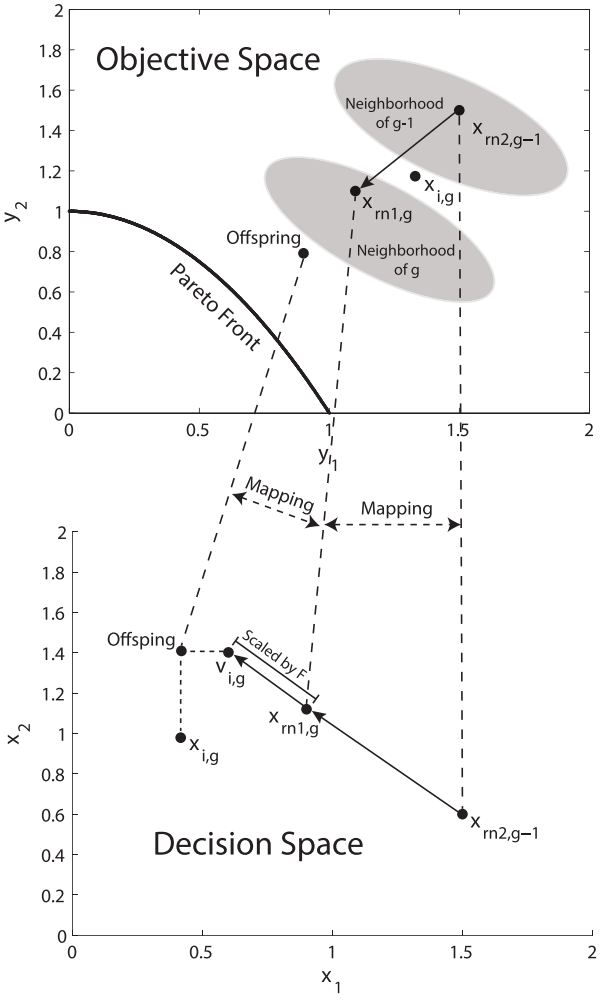


Fig. 2. Procedure of NCG mutation and the relationship among involved individuals. Both objective functions are minimized. The PF, variable values, and objective values are arbitrarily generated for convenience of illustration.

if the neighborhood size is too large, it will become difficult to estimate the converging direction near $X_{i,g}$ since $X_{rn_1,g}$ and $X_{rn_2,g-1}$ can be selected from distant regions in objective space. Based on our preliminary tests (see the supplementary material), it is recommended to choose neighborhood size T as 5% of population size. Fig. 2 illustrates the procedure of the proposed NCG mutation and the relationship among involved individuals. In objective space, the vector from $X_{rn_2,g-1}$ to $X_{rn_1,g}$ is the estimated converging direction. In decision space, the difference of the two individuals decides the corresponding searching direction. Subsequently, $X_{rn_1,g}$ will continue to explore following this search direction to generate the mutant vector $V_{i,g}$. The searching step is scaled by the parameter F. Offspring is produced via recombination of main parent $X_{i,g}$ and mutant vector $V_{i,g}$. The effectiveness of the obtained searching direction is validated through a contrast experiment in Section IV.

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 converging population [40], [42], [55], [56]. Since the

individuals involved in NCG mutation are selected from neighborhoods of the same main parent, the explorative ability of the algorithm is impaired. In order to compensate for the above-mentioned weakness of NCG mutation, another new mutation strategy is proposed, namely PCG mutation. The NCG and PCG mutations will be utilized simultaneously so that a reasonable tradeoff between exploitation and exploration can be achieved. The details of the PCG mutation operation are

$$V_{i,g} = X_{i,g} + F \cdot \left(X_{rp_1,g} - X_{rp_2,g-1}\right) \tag{2}$$

where $V_{i,g}$ denotes the new generated mutant vector, i is the index of the main parent, and g is the number of current generation. $X_{i,g}$ is the main parent. $X_{rp_1,g}$ is an individual randomly selected from the whole population in current generation, where index rp_1 is a randomly selected integer from $\{1, 2, 3, \ldots, N\}$ and N 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 $\{1, 2, 3, \ldots, N\}$.

Unlike the NCG mutation, the difference vector in PCG mutation is computed 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 difference will vary larger than the original DE mutation strategy. In this way, the explorative ability could be further enhanced. Another special modification of the population-based variant is that the difference vector would be added to the main parent directly. By these means, the searching will continue centering on the main parent instead of performing a purely stochastic exploration. This could lead to a more efficient optimization process.

In ACGDE, the above two strategies are employed in a half–half manner, which means they have the same probability (50%) to be utilized during each mutation operation. Parametric sensitivity of this ratio would be analyzed via experiments in Section IV. Simulation results show that the performance of the algorithm is better than, or at least comparable to, 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 main 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 main parent to generate the final offspring.

5) Enhancing Both Diversity and Convergence: When designing an MO algorithm, it is difficult to hasten the convergence speed without loss of diversity as if they were conflicting to each other. Nonetheless, the specially designed mechanism enables the proposed approach to enhance convergence along with the diversity. In the NCG mutation, 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 main parent in objective space so that the diversity of the population could be consistently maintained. Analogously, the parent-centric search in PCG mutation helps to preserve the diversity, whilst the more explorative mutation operator contributes to a higher converging speed to true PF.

C. Cross-Generation Adaptation Mechanism

1) Parametric Sensitivity in Multiobjective Optimization: Sensitivity to the parametric setting is a critical issue during the extension of DE into MO [22]–[24]. The performance of original DE may vary tremendously with different selections of parameters [2], [18]–[20]. This problem becomes more challenging in solving MOPs. Multiple objectives may have disparate requirements for the parametric setup, and even for a certain objective, the optimal setup of parameters may vary over 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, it is desirable to have an adaptation mechanism with the following features: first, the scheme is able to estimate the appropriate parameters for a specific region in objective space; second, there is a selection pressure to reserve more proper parameters and discard poor parameters; and third, as the current fitted parameters may not conform to the requirement of the next searching stage, complete convergence of parameters should be avoided. In order to achieve the above targets, a novel adaptation mechanism that makes use of the information across generations is proposed in this paper.

2) Proposed Adaptation Mechanism: In the proposed CGA mechanism, the two control parameters in DE, namely scaling factor F and crossover probability Cr, are applied at the individual level. Each individual in the population is associated with its own F and Cr. The neighborhood pools utilized in the NCG mutation also contributes to the reproduction of parameters for offspring.

During initialization, for each individual X_i, the associated F_i and Cr_i are randomly generated from the predefined interval $[F_{\min}, F_{\max}]$ and $[\mathrm{Cr}_{\min}, \mathrm{Cr}_{\max}]$. After that, the parameter values for any newly generated individual will be determined by

$$F_{\mathrm{spring},i,g} = F_{\mathrm{neighbor},i,g} + \theta_F \cdot \mathrm{Gaussian}(0, 1) \tag{3}$$

$$\mathrm{Cr}_{\mathrm{spring},i,g} = \mathrm{Cr}_{\mathrm{neighbor},i,g} + \theta_{\mathrm{Cr}} \cdot \mathrm{Gaussian}(0, 1) \tag{4}$$

where $F_{\mathrm{spring},i,g}$ and $\mathrm{Cr}_{\mathrm{spring},i,g}$ denote the parameter values assigned to the newly generated offspring, and i is the index of the main parent shared in the mutation operation, g is the current generation number. Gaussian$(0, 1)$ is a random number sampled from a Gaussian distribution accordingly with mean 0 and standard deviation 1. θ_F and θ_{Cr} are constants for scaling the outputs of Gaussian distribution. $F_{\mathrm{neighbor},i,g}$ and $\mathrm{Cr}_{\mathrm{neighbor},i,g}$ are produced in the following manner:

$$F_{\mathrm{neighbor},i,g} = \mathrm{mean}_A\left(NF_{i,g} \cup NF_{i,g-1}\right) \tag{5}$$

$$\mathrm{Cr}_{\mathrm{neighbor},i,g} = \mathrm{mean}_A\left(N\mathrm{Cr}_{i,g} \cup N\mathrm{Cr}_{i,g-1}\right) \tag{6}$$

with $NF_{i,g}$ denoting the set of scaling factors associated with all the individuals in main parent's neighborhood of current generation, and $NF_{i,g-1}$ comprising all the scaling factors for main parent's neighborhood of previous generation. Same rule is also applied for generating the Cr sets $N\mathrm{Cr}_{i,g}$ and $N\mathrm{Cr}_{i,g-1}$. mean_A stands for the simple arithmetic mean. If the calculated

$F_{\text{spring},i,g}$ or $\text{Cr}_{\text{spring},i,g}$ falls outside the interval $[F_{\min}, F_{\max}]$ or $[\text{Cr}_{\min}, \text{Cr}_{\max}]$, its value will be set as the nearest bound.

Following the above adaptation operation, the offspring will then be generated via mutation and crossover utilizing the obtained $F_{\text{spring},i,g}$ and $\text{Cr}_{\text{spring},i,g}$. In other words, the offspring itself is produced only after the generation of its associated parameters. Once an individual is generated, its associated parameters will not be altered until the individual fails to survive to the next generation.

3) Explanations of CGA Mechanism: By encoding the control parameters at the individual level, CGA mechanism is able to evaluate multiple sets of parameters simultaneously and preserve the favorable control values for each solution. The computational complexity does not increase compared to the fixed parametric scheme in original DE [19]. The fundamental principle behind CGA mechanism is that suitable parameter values tend to generate better individuals, and favorable individuals help to propagate their associated parameters. Based on the above essence, CGA mechanism tries to estimate the most proper parametric setting for the currently interested region in objective space. In SO, the target of optimization is to obtain the optimal solution for one objective, thus, the adaptation of parameters is performed without considering the individuals' location in objective space. However, when solving MOPs, multiple objectives may have disparate requirements for parametric setup and, hence, adjustment of parameters needs to be conducted in an objective space-based manner. Relationship in terms of the position in objective space plays an important role in CGA mechanism. Two neighborhood pools, which are identical to those in NCG mutation, are defined for each individual. Based on the neighborhood pools of current generation and previous generation, parameter sets $NF_{i,g}$, $NF_{i,g-1}$, $N\text{Cr}_{i,g}$, and $N\text{Cr}_{i,g-1}$ are generated. As stated before, good parameters are more likely to produce individuals that are able to survive. The associated parameters of the neighborhood members, therefore, reflect the required parametric setting for current region. Calculating the mean of their parameters is a reasonable way to approximate the optimal setup and moderate the effect of possible outliers.

Similar to NCG mutation, selection of the neighborhood size T is to achieve a balance between robustness and accuracy. If the T is too large, then the calculated mean is incapable of revealing the requirement of current region. In case that the T is too small, the estimated results will depend heavily on the parameters of a single or few individuals, which are unstable because of the applied perturbation operation. In this scenario, employment of the information across consecutive generations provides an advantage. With the neighborhood size T, actually $2T$ individuals are involved in the mean calculation. The information collected is twice what can be extracted from a single generation. The estimation becomes more reliable, and the influence of outliers is further reduced. In the main algorithm, the neighborhood pools are shared by CGA mechanism and NCG mutation with the recommended T as 5% of the population size.

Subsequently, a perturbation randomly sampled from a scaled Gaussian distribution is added to the computed mean $F_{\text{neighbor},i,g}$ and $\text{Cr}_{\text{neighbor},i,g}$ to finalize the parameter values of offspring. The purpose of introducing perturbations is to keep exploring new parameters throughout the evolutionary process. Premature convergence of parameters should be avoided because even for a single region in objective space, the optimal parameters may not stay consistent in different optimization stages. Owing to the scaled Gaussian distribution, exploration is centered on the estimated optimal values, whilst having a low probability to produce a dramatic variation. Values of $F_{\text{neighbor},i,g}$ and $\text{Cr}_{\text{neighbor},i,g}$ remain relatively stable unless most neighborhood members are updated simultaneously by individuals with new parameters. It indicates that the previous parametric setup is not suitable for the current searching stage, thereby the CGA mechanism proceeds to adapt the parameters. It is notable that the newly generated $F_{\text{spring},i,g}$ and $\text{Cr}_{\text{spring},i,g}$ will be employed in the mutation and crossover operators instead of the associated control parameters of parents. If the corresponding offspring succeeds to survive, then the $F_{\text{spring},i,g}$ and $\text{Cr}_{\text{spring},i,g}$ become its associated parameters.

D. Contributions of ACGDE Algorithm

The main task of ACGDE is actually to regenerate better individuals for the implemented MO framework (e.g., NSGA-II [41] and MOEA/D [42]). The mutation and crossover operation in original MO framework will be replaced by components of ACGDE. Survival selection is performed with the existing mechanisms, e.g., nondominated sorting in NSGA-II or scalar function in MOEA/D. As a result, ACGDE is flexible in responding to numerous optimization requirements. Depending on the properties of current problem, ACGDE is convenient to implement into different MO frameworks. Experimental results in Section IV show a significant improvement in NSGA-II and MOEA/D with ACGDE.

Compared with conventional DE operators, ACGDE introduces sub-rank calculations and neighborhood assignment. The sub-rank computations require sorting the population according to each objective function value in ascending order of magnitude. The complexity of this procedure is $O(MN \log N)$, where M is the number of objectives and N is the population size. Thereafter, for each solution we calculate the distance from that solution to all the other individuals in terms of sub-rank and determine the neighborhood members according to the computed distances. In a proper implementation, this has $O(MN^2)$ computational complexity. Thus, the overall complexity of ACGDE is $O(MN^2)$, which is identical to that of original NSGA-II [41].

IV. Empirical Study

This section evaluates the overall performance of ACGDE and effectiveness of each algorithmic component through experiments.

A. Comparison With State-of-the-Art MOEAs

To evaluate the optimization performance of the new algorithm, the proposed ACGDE is implemented into NSGA-II [41], which is a well-known powerful MO algorithm.

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

Problems (2-obj)	ACGDE-NSGA-II Mean(Std)	NSGA-II-DE Mean(Std)	NSGA-II(SBX) Mean(Std)	MOEA/D-DE Mean(Std)	MOEA/D(SBX) Mean(Std)	CCPSO Mean(Std)	MOEGS Mean(Std)	SPEA2 Mean(Std)
UF1	0.0528† (0.0148)	0.0603† (0.0162)	0.1230†‡ (0.0318)	**0.0475 (0.0372)**	0.1568†‡ (0.0652)	0.0483† (0.0108)	0.2207†‡ (0.1025)	0.1341†‡ (0.0407)
UF2	**0.0205 (0.0027)**	0.0429†‡ (0.0047)	0.0481†‡ (0.0125)	0.0426†‡ (0.0316)	0.0640†‡ (0.0310)	0.0481†‡ (0.0079)	0.0484†‡ (0.0091)	0.0626†‡ (0.0071)
UF3	**0.0947 (0.0139)**	0.1515†‡ (0.0271)	0.2179†‡ (0.0666)	0.1513†‡ (0.0688)	0.3064†‡ (0.0300)	0.3024†‡ (0.0390)	0.1318†‡ (0.0370)	0.3025†‡ (0.0410)
UF4	**0.0410 (0.0003)**	0.0723†‡ (0.0078)	0.0533†‡ (0.0018)	0.0866†‡ (0.0104)	0.0560†‡ (0.0034)	0.0639†‡ (0.0068)	0.1277†‡ (0.0105)	0.0682†‡ (0.0031)
UF5	**0.2870 (0.0932)**	0.8494†‡ (0.1698)	0.3257 (0.0943)	0.7643†‡ (0.1307)	0.4318†‡ (0.0812)	0.4535†‡ (0.0734)	0.8727†‡ (0.5201)	0.4741†‡ (0.0877)
UF6	**0.1576 (0.0849)**	0.4181†‡ (0.0819)	0.2302 (0.0680)	0.4386†‡ (0.2206)	0.4374†‡ (0.1509)	0.4701†‡ (0.0356)	0.6323†‡ (0.4519)	0.4609†‡ (0.1292)
UF7	**0.0262 (0.0071)**	0.0389† (0.0422)	0.2359†‡ (0.1447)	0.1018‡ (0.1648)	0.3536†‡ (0.1552)	0.0961†‡ (0.0381)	0.0833†‡ (0.1027)	0.1693†‡ (0.1285)
WFG1	**0.8084 (0.0316)**	1.2181†‡ (0.0052)	1.0790†‡ (0.0813)	1.1634†‡ (0.0138)	1.0483†‡ (0.0458)	0.9758†‡ (0.0359)	1.1818†‡ (0.0183)	1.0859†‡ (0.0185)
WFG2	**0.0139 (0.0008)**	0.0461†‡ (0.0253)	0.1604†‡ (0.0277)	0.1666†‡ (0.0880)	0.1871†‡ (0.0643)	0.5447†‡ (0.1463)	0.5816†‡ (0.1444)	0.2381†‡ (0.0281)
WFG3	**0.0201 (0.0009)**	0.0344†‡ (0.0017)	0.0211†‡ (0.0016)	0.0204 (0.0018)	0.0203† (0.0059)	0.1826†‡ (0.0521)	0.2399†‡ (0.0558)	0.1036†‡ (0.0305)
WFG4	0.0196†‡ (0.0014)	0.0927†‡ (0.0037)	0.0189†‡ (0.0011)	0.0811†‡ (0.0081)	**0.0167 (0.0015)**	0.0614†‡ (0.0328)	0.1842†‡ (0.0342)	0.0363†‡ (0.0052)
WFG5	**0.0682 (0.0015)**	0.0754†‡ (0.0017)	0.0705†‡ (0.0005)	0.0692†‡ (0.0003)	0.0691†‡ (0.0006)	0.0899†‡ (0.0134)	0.1698†‡ (0.1090)	0.0734†‡ (0.0012)
WFG6	0.1175†‡ (0.0205)	0.1079†‡ (0.0349)	**0.0640 (0.0068)**	0.1072†‡ (0.0319)	0.0820†‡ (0.0237)	0.1228†‡ (0.0489)	0.1335†‡ (0.0373)	0.0867†‡ (0.0162)
WFG7	**0.0166 (0.0007)**	0.0306†‡ (0.0019)	0.0170 (0.0010)	0.0190†‡ (0.0011)	0.0205 (0.0111)	0.2122†‡ (0.0730)	0.2250†‡ (0.1229)	0.0507†‡ (0.0184)
WFG8	**0.1089 (0.0036)**	0.1413†‡ (0.0101)	0.1371†‡ (0.0065)	0.1271†‡ (0.0128)	0.1270†‡ (0.0097)	0.2203†‡ (0.0489)	0.2502†‡ (0.0337)	0.1700†‡ (0.0107)
WFG9	0.1259†‡ (0.0004)	0.1106†‡ (0.0380)	0.0844‡ (0.0529)	**0.0597 (0.0287)**	0.0606‡ (0.0381)	0.1519†‡ (0.0534)	0.2660†‡ (0.1514)	0.1070† (0.0386)
UF8 (3-obj)	0.1383†‡ (0.0420)	0.1520†‡ (0.0300)	0.2194†‡ (0.0098)	**0.0911 (0.0124)**	0.148†‡ (0.0358)	0.257†‡ (0.0528)	0.1585†‡ (0.0319)	0.2781†‡ (0.0195)
UF9 (3-obj)	0.1776†‡ (0.1091)	0.1938†‡ (0.0646)	0.1635†‡ (0.0491)	**0.1065 (0.0452)**	0.134† (0.0624)	0.2873†‡ (0.0501)	0.199†‡ (0.0632)	0.3689†‡ (0.0535)
UF10 (3-obj)	0.6440†‡ (0.2715)	2.4308†‡ (0.1848)	0.3236† (0.0703)	0.5826†‡ (0.0716)	**0.2937 (0.1304)**	0.4859†‡ (0.0298)	6.9530†‡ (1.4222)	0.8201†‡ (0.2349)

† and ‡ indicate that the difference between the marked entry and the best entry is statistically significant using Wilcoxon rank sum test and unpaired two-sample student's t-test, respectively (both are at the 5% significance level).

The offspring in NSGA-II will be generated via ACGDE, and the survival selection part of the original framework is preserved including nondominated sorting and density estimation. The performance of ACGDE–NSGA-II is compared with seven state-of-the-art MOEAs.

1) NSGA-II (SBX) [41] with $p_c = 1.0,\ \eta_c = 20, \eta_m = 20,\ p_m = 1/N$.
2) NSGA-II–DE [40] with $F = 0.5,\ p_c = 1.0,\ \eta_m = 20,\ p_m = 1/N,\ \delta = 0.9,\ T = 20,\ n_r = 2$.
3) MOEA/D (SBX) [42] with $p_c = 1.0,\ \eta_c = 20,\ \eta_m = 20,\ p_m = 1/N$.
4) MOEA/D–DE [40] with $F = 0.5,\ p_c = 1.0,\ \eta_m = 20,\ p_m = 1/N,\ \delta = 0.9,\ T = 20,\ n_r = 2$.
5) CCPSO [57] with $p_c = 1.0,\ p_m = 1/N$, sub-population size: four for two-objective problems and 15 for three-objective problems, turbulence operator with probability of 0.1, inertia weight as 0.4, dynamic sharing for niche radius.
6) MOEGS [58] with $\kappa = 10$, number of trial solutions as 50, hypervolume (HYP) performance indicator for fitness assignment approach.
7) SPEA2 [59] with $p_c = 0.8,\ p_m = 1/N, n_{\text{bit}} = 15$.

Recommended parametric settings in the original literatures are utilized for the above MOEAs. Following the standard experimental settings [40], [41], for all the testing algorithms, the population size is fixed as 100 for two-objective benchmarks, 300 for three-objective benchmarks, and the maximum number of function evaluations is set to 5×10^4 for two-objective problems, 15×10^4 for three-objective problems. The control parameters of ACGDE are set as follows: $T = 5$ for two-objective problems, $T = 15$ for three-objective problems. $\theta_F = 0.4,\ \theta_{\text{Cr}} = 0.2,\ F_{\max} = 0.9,\ F_{\min} = 0.1,\ \text{Cr}_{\max} = 0.5,$

and $\mathrm{Cr}_{\min} = 0.2$. The selection of T is according to our preliminary tests (see the supplementary material), and the selection of other parameters is based on some empirical studies about parametric setup in DE [2], [19], [60].

In total, 19 frequently used MO benchmark problems were tested to evaluate the performance of the algorithm, among which UF1–UF10 are unconstrained problems from CEC-09 special session and competition [61] and WFG1–WFG9 are from from the Walking Fish Group (WFG) test suites [62]. 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. Inverted generational distance (IGD) [63] and hypervolume (HYP) [64] are selected as the performance metrics to quantitatively compare the performance of algorithms in terms of both convergence and diversity [65], [66]. The values of HYP are calculated by means of Monte Carlo simulation as in [67]. All the simulations were done on an Intel Core i7 machine with 16-GB RAM and 3.40-GHz speed. Microsoft Visual Studio 2012 Express is used to develop the coding and run the experiments. Tables I and S2 (see the supplementary material) show the mean and standard deviation of the IGD values and HYP values, respectively, for 30 independent runs of each algorithm on each benchmark. The best entries in terms of mean value are marked in boldface. In order to judge whether the results of the best performing algorithm differ from the results of competitors in a statistically significant way, Wilcoxon rank-sum test [68] and unpaired two-sample Student's t-test are conducted at the 5% significance level. The entries which are significantly different from the best entries are indicated by the symbols † and ‡.

From the comparative results, it clearly elucidates that ACGDE–NSGA-II is powerful in tackling multiobjective problems when compared to its competitors. For two-objective problems, it achieves the best results in 12 out of 16 benchmarks in terms of both IGD and HYP. For three-objective problems, ACGDE–NSGA-II performs best in two out of three benchmarks in terms of HYP, whereas in terms of IGD, the proposed algorithm is outperformed by MOEA/D variants. The explanation is that NSGA-II is a dominance-based MOEA, the efficiency of nondominated sorting will be deteriorated when the number of objectives increases. The ability of ACGDE in solving three-objective problems is further evaluated by implementation into MOEA/D in next section. To validate the effectiveness of ACGDE, it is reasonable to make a comparison between ACGDE–NSGA-II and original NSGA-II–DE. Separate statistical tests are performed between the two algorithms over all the benchmarks (partially shown in Tables I and S2). In terms of HYP, ACGDE–NSGA-II provides *significantly* better results in 16 out of 19 problems. In terms of IGD, ACGDE–NSGA-II *significantly* outperforms original NSGA-II–DE in 17 out of 19 problems, and the latter cannot give the best performance in any of the benchmarks. The only disparity between these two algorithms lies in the DE operator, from which we can draw the conclusion that the success of ACGDE–NSGA-II benefit from the newly developed cross-generation mutation operators and the parameter adaptation mechanism.

TABLE II
MEAN AND STANDARD DEVIATION OF THE IGD VALUES (30 RUNS)

Problems	ACGDE -MOEA/D Mean(Std)	MOEA/D -DE Mean(Std)	Problems	ACGDE -MOEA/D Mean(Std)	MOEA/D -DE Mean(Std)
UF1 (2-obj)	**0.0448 (0.0155)**	0.0475 (0.0372)	WFG5 (2-obj)	**0.0685 (0.0006)**	0.0692†‡ (0.0003)
UF2 (2-obj)	**0.0195 (0.0039)**	0.0426†‡ (0.0316)	WFG6 (2-obj)	**0.0998 (0.0347)**	0.1072† (0.0319)
UF3 (2-obj)	**0.1306 (0.0398)**	0.1513 (0.0688)	WFG7 (2-obj)	**0.0149 (0.0001)**	0.019†‡ (0.0011)
UF4 (2-obj)	**0.0436 (0.0014)**	0.0866†‡ (0.0104)	WFG8 (2-obj)	**0.1104 (0.0081)**	0.1271†‡ (0.0128)
UF5 (2-obj)	**0.4654 (0.0974)**	0.7643†‡ (0.1307)	WFG9 (2-obj)	0.1062†‡ (0.0404)	**0.0597 (0.0287)**
UF6 (2-obj)	**0.2893 (0.1694)**	0.4386† (0.2206)	WFG1 (3-obj)	**1.0470 (0.0427)**	1.4836†‡ (0.0064)
UF7 (2-obj)	**0.0521 (0.0882)**	0.1018 (0.1648)	WFG2 (3-obj)	**0.4011 (0.0142)**	0.4045† (0.0331)
UF8 (3-obj)	0.1045†‡ (0.0112)	**0.0911 (0.0124)**	WFG3 (3-obj)	**0.0484 (0.0010)**	0.0674†‡ (0.0076)
UF9 (3-obj)	**0.0760 (0.0401)**	0.1065†‡ (0.0452)	WFG4 (3-obj)	**0.3481 (0.0023)**	0.3797†‡ (0.009)
UF10 (3-obj)	**0.2481 (0.0660)**	0.5826†‡ (0.0716)	WFG5 (3-obj)	**0.1934 (0.0004)**	0.1952†‡ (0.0011)
WFG1 (2-obj)	**0.9680 (0.0128)**	1.1634†‡ (0.0138)	WFG6 (3-obj)	**0.2227 (0.0201)**	0.2611†‡ (0.0218)
WFG2 (2-obj)	**0.1297 (0.0674)**	0.1666† (0.088)	WFG7 (3-obj)	**0.1614 (0.0004)**	0.1735†‡ (0.0062)
WFG3 (2-obj)	**0.0159 (0.0008)**	0.0204†‡ (0.0018)	WFG8 (3-obj)	**0.2457 (0.0188)**	0.3392†‡ (0.0340)
WFG4 (2-obj)	**0.0226 (0.0025)**	0.0811†‡ (0.0081)	WFG9 (3-obj)	**0.2136 (0.0348)**	0.2233† (0.022)

† and ‡ indicate that the difference between ACGDE-MOEA/D and original MOEA/D-DE is statistically significant using Wilcoxon rank sum test and unpaired two-sample student's t-test, respectively (both are at the 5% significance level).

B. Implementation in MOEA/D

In this section, ACGDE is implemented into MOEA/D [42] by replacing the original mutation and crossover operators in MOEA/D–DE [40]. Besides the 19 benchmarks used in Section IV-D, three-objective versions of WFG1–WFG9 are included as well. The experimental and parametric setups are identical with those in Section IV-A. Table II presents the mean and standard deviation of the IGD values for 30 independent runs of both algorithms on each benchmark. The best entries in terms of mean value are marked in boldface.

According to the results, ACGDE–MOEA/D is able to provide better performance than original MOEA/D–DE in 26 out of 28 benchmark problems. Considering the statistical tests, ACGDE *significantly* improves the performance of MOEA/D in 11 out of 12 three-objective problems, and 12 out of 16 two-objective problems, respectively. To summarize, ACGDE is good at solving not only two-objective problems but also those with three objectives. Moreover, successful implementation in both NSGA-II and MOEA/D demonstrate the robustness of ACGDE during combination with different MO frameworks.

C. Effectiveness of Searching Directions in NCG Mutation

In the proposed NCG mutation, the exploration is guided by the searching direction corresponding to the estimated converging direction. For further investigation, one inevitable

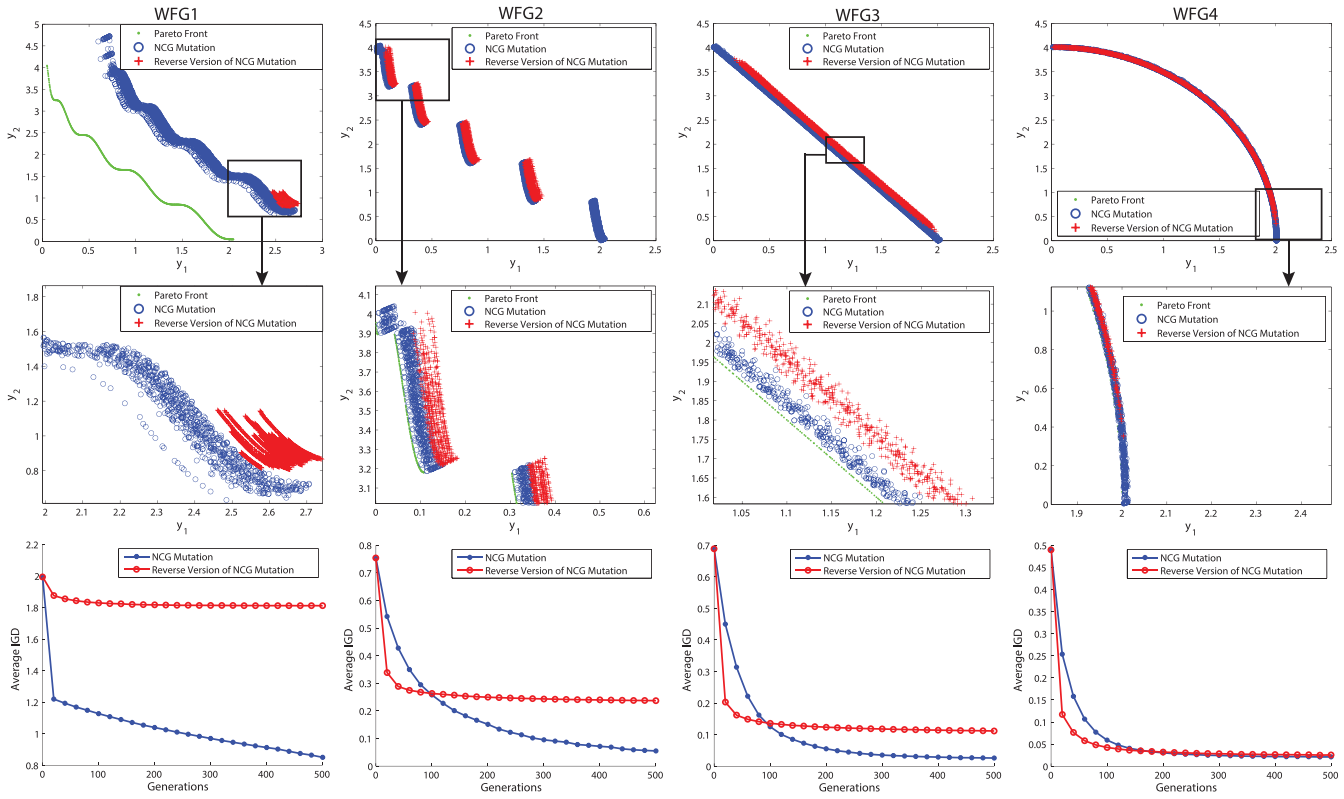


Fig. 3. Performance of the original NCG mutation compared with that of reverse NCG mutation on WFG1–WFG4. For each benchmark, the final solution sets obtained by both algorithms over 30 independent runs are plotted. The average IGD values of 30 runs over generations are shown as well.

question to address is: will these computed searching directions really help the optimization process? Or maybe they are only randomly generated perturbations? In order to clearly demonstrate the meaningfulness of calculated searching directions, a reverse version of NCG mutation will be employed for a contrast experiment.

Original NCG mutation

$$V_{i,g} = X_{rn_1,g} + F \cdot \left(X_{rn_1,g} - X_{rn_2,g-1}\right). \tag{7}$$

Reverse version of NCG mutation

$$V_{i,g} = X_{rn_1,g} + F \cdot \left(X_{rn_2,g-1} - X_{rn_1,g}\right). \tag{8}$$

As can be seen, all the individuals and parameters involved in the reverse version are identical to those in the original version. The only difference comes from the exchange of $X_{rn_1,g}$ and $X_{rn_2,g-1}$. Thus, the influences of all the other factors are eliminated, and the variation in performance, if any, only results from their different searching directions. With this reverse version of NCG mutation, it will be much easier to judge the effectiveness of searching directions in original NCG mutation: first, if the obtained searching direction is not different from a stochastic search, then the performance of the two mutation strategies should be almost the same and second, if the utilized searching directions in original version are meaningful and favorable, then an obvious improvement in performance should be observed compared with that of reverse version.

In this contrast experiment, the PCG mutation strategy and the CGA mechanism are removed from the original ACGDE. The simplified ACGDE only with NCG mutation or reverse NCG mutation are implemented into NSGA-II. For both algorithms, the scaling factor F is set as 0.5 and the crossover rate Cr is set as 0.35 based on the parametric setup in Section IV-A. WFG1–WFG8 from WFG test suites are utilized as benchmark problems. Other experimental and parametric setups are identical with those in Section IV-A. IGD is employed as the indicator to quantitatively evaluate the performance of each algorithm. Figs. 3 and S1 (see the supplementary material) present the final solution sets obtained by both algorithms for 30 independent runs and the average IGD values of 30 runs over generations.

From Figs. 3 (results on WFG1–WFG4) and S1 (results on WFG5–WFG8), the performance of the original NCG mutation is significantly better than the reverse version in six out of eight benchmark problems. For WFG1, the difference between the results of the two algorithms is fairly significant. The final solution sets obtained by the reverse version is very poor in both diversity and convergence. It reveals that the current searching directions are completely erroneous, and they have hampered the whole optimization process of the algorithm. In other words, the opposite searching directions, that utilized in the original NCG mutation, are close to the correct searching directions. Actually, the performance provided by the original NCG mutation in WFG1 is the best when compared with those of other state-of-the-art MO algorithms in Section IV-A. For WFG2, WFG3, WFG5, WFG7, and WFG8, the original NCG mutation outperforms the reverse version in terms of convergence enhancement and

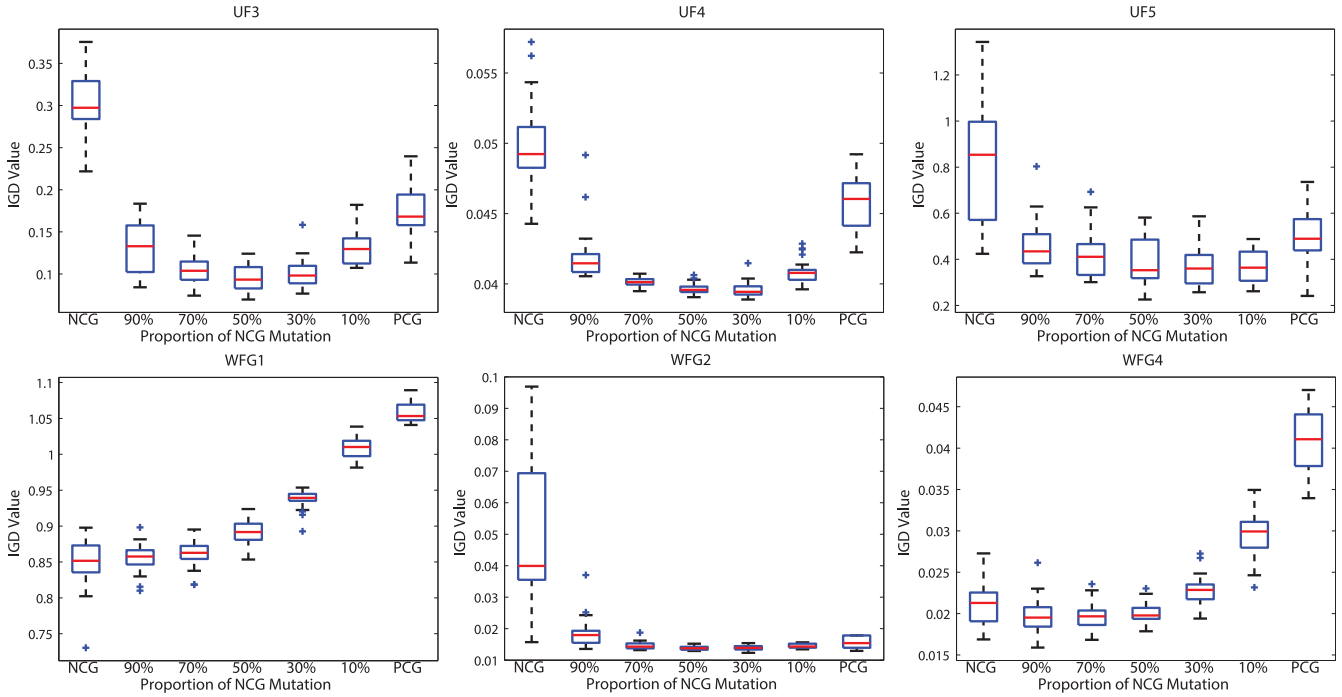


Fig. 4. Boxplots of the IGD values over 30 independent runs. The simulations are performed on UF3–UF5, WFG1, WFG2, and WFG4 separately. In each plot, the leftmost box represents the variant only using NCG mutation, and the rightmost box denotes the variant only utilizing PCG mutation. The percentages under each box indicate the probability to employ NCG mutation in this variant.

diversity maintenance. Since the solutions plotted are from 30 independent runs, the consistent improvement is able to prove the effectiveness of the searching directions in NCG mutation. For WFG4, the two tested algorithms gave similar results. However, as shown in the zoomed-in plot, the performance of the original NCG mutation is slightly better at the aspect of coverage of true PF. For WFG6, it can be observed that the reverse version performs better in convergence while the original version achieves greater diversity. If we look into the enlarged plot and the IGD values over generations, it is obvious that the original NCG mutation is trapped into a local optimum. This is a common issue for direction- or gradient-based approach [69]–[72]. Nevertheless, considering the significant improvement in most benchmark problems, the proposed NCG mutation is still promising for solving MOPs without complex local optima. Furthermore, the final version of ACGDE will include PCG mutation and CGA mechanism to enhance the explorative ability of the algorithm so that the optimization process is less likely to get stuck in local optimum.

D. Effectiveness of Combining NCG Mutation With PCG Mutation

NCG mutation is good at estimating the efficient searching direction but limited by its diminishing explorative ability. PCG mutation possesses a powerful explorative capability but is weak in exploitation. To strike a delicate balance between exploration and exploitation, NCG and PCG mutations are employed concurrently in ACGDE algorithm. A rate is predefined to decide the probability of utilizing NCG mutation scheme in each mutation operation, otherwise, PCG mutation is performed. In the current version of ACGDE, this rate is set as 50% so that both mutation strategies have the equal chance to be used. In order to investigate the influence of this rate and to validate the effectiveness of the combination, simulations have been run with different probabilities to utilize NCG mutation. The experimental setup and parametric setting is the same as those in previous section. Six representative benchmark problems are selected from WFG test suites and CEC-09, and each problem has been tested for 30 independent runs. Fig. 4 compares the performance of different variants in terms of IGD. Each box represents the distribution of IGD values obtained by the corresponding variant in 30 independent simulation runs. According to Fig. 4, the hybrid variants provide better results for five out of six benchmarks when compared with pure NCG variant and pure PCG variant. For WFG1, the searching directions estimated by NCG mutation are rather efficient, and there is no strong need for exploration. Thus, the performance of the algorithm declines with the decreased proportion of NCG mutation. For other problems, improvement over the variants with only one mutation strategy demonstrates the advantage of the combination. Among the several hybrid variants, the 50% version exhibits a relatively more robust performance. By contrast, the outcomes of 70% and 30% versions are more problem dependent. For other experiments in this paper, the 50% variant will always be applied.

E. Effectiveness of CGA Mechanism

This section aims at observing the behavior of parameter adaptation and verifying the effectiveness of the proposed CGA mechanism. Compared to crossover probability Cr,

the overall performance of DE depends more heavily on the selection of scaling factor F [73], [74]. As shown in the mutation formula, F has a big role in controlling the searching step of the algorithm. In order to investigate how CGA mechanism adapts F during the optimization process, the fluctuation of F values over generations are plotted for all the individuals in Fig. S2 (see the supplementary material). To better visualize the distribution of F values in objective space, before plotting, the individuals have already been sorted according to their fitness in the first objective (better individuals are assigned with smaller indices). Since the two objectives in the tested benchmarks are conflicting with each other, larger index means this individual focus more on optimizing the second objective. As a result, the order of the individuals in the plot actually follows their positions in objective space. In the simulation, ACGDE as stated in Algorithm 1 is implemented in NSGA-II. The experimental and parametric setups are identical with those in Section IV-A. Fig. S2 (see the supplementary material) visualizes the distribution of F values for the whole evolutionary process on three representative benchmark problems.

Based on Fig. S2, distinct patterns are observed in the distribution of F values while optimizing the three problems. For WFG3, F values below 0.5 are preferred in the early searching stage. From generation 300 onward, F values above 0.5 are accepted more frequently. The difference between the distributions in WFG5 and WFG8 is fairly obvious. Throughout the whole evolution, lower F values always hold a higher chance to survive in WFG8, whereas larger F values appear more in WFG5. From the perspective of objective space, in WFG5, the individuals which optimize the two objectives with similar weights (index around 50) are more likely to employ small F values in some searching stages (shown as the blue areas in the middle of the plot) although the whole population tend to reserve large F values. For WFG3 and WFG8, no evident pattern with regards to objective space emerged. Actually, in most existing benchmark problems, the requirements of parametric setup are roughly the same for its multiple objectives. From the testing results, it is explicit that CGA mechanism is intelligently adapting the parameters according to the current optimization task instead of a purely stochastic regeneration.

Next, a contrast experiment is performed to validate the effectiveness of CGA mechanism. CGA mechanism is removed from ACGDE, instead, fixed parameters are utilized as in original DE. The performance of the simplified version is compared with that of original ACGDE. Both algorithms are implemented in NSGA-II. In the simplified version, F is fixed as 0.5 and Cr is fixed as 0.35 based on the parameter intervals in CGA. Other experimental and parametric setups are identical with those in Section IV-A. UF1–UF10 from CEC-09 and WFG1–WFG9 from WFG test suites are tested. Table III shows the mean and standard deviation of the IGD values for 30 independent runs on each benchmark. The best entries in terms of mean value are marked in boldface. According to Table III, original ACGDE outperforms the simplified version without CGA mechanism in 17 out of 19 benchmark problems, from which we can conclude that the introduction of CGA mechanism has substantially improved the performance of the algorithm.

TABLE III
MEAN AND STANDARD DEVIATION OF THE IGD VALUES (30 RUNS)

Problems	with CGA	without CGA
	Mean ± Std	Mean± Std
UF1	**0.0528 ± 0.0148**	0.0729 ± 0.0410
UF2	**0.0205 ± 0.0027**	0.0264 ± 0.0069
UF3	**0.0947 ± 0.0139**	0.0953 ± 0.0158
UF4	0.0410 ± 0.0003	**0.0397 ± 0.0004**
UF5	**0.2870 ± 0.0932**	0.3852 ± 0.1044
UF6	**0.1576 ± 0.0849**	0.1918 ± 0.0797
UF7	**0.0262 ± 0.0071**	0.0467 ± 0.0465
UF8	**0.1383 ± 0.0420**	0.1965 ± 0.0813
UF9	**0.1776 ± 0.1091**	0.2914 ± 0.1385
UF10	**0.6440 ± 0.2715**	1.3504 ± 0.6490
WFG1	**0.8084 ± 0.0316**	0.8917 ± 0.0154
WFG2	**0.0139 ± 0.0008**	0.0387 ± 0.0558
WFG3	**0.0201 ± 0.0009**	0.0206 ± 0.0010
WFG4	**0.0196 ± 0.0014**	0.0200 ± 0.0011
WFG5	**0.0682 ± 0.0015**	0.0686 ± 0.0003
WFG6	**0.1175 ± 0.0205**	0.1200 ± 0.0179
WFG7	0.0166 ± 0.0007	**0.0164 ± 0.0009**
WFG8	**0.1089 ± 0.0036**	0.1094 ± 0.0044
WFG9	**0.1259 ± 0.0004**	0.1261 ± 0.0003

V. CONCLUSION

This paper has presented a new DE variant for MO, which employs information across generations to guide the searching directions. Two variants of cross-generation mutation operators have been proposed to enhance both convergence and diversity in the evolution. Furthermore, a CGA mechanism is introduced to tune the parameters dynamically from a perspective of objective space. Experimental results demonstrate that the proposed algorithmic components are effective and the new algorithm is able to significantly enhance the performance of NSGA-II and MOEA/D.

In future work, there are plans to introduce a dynamic mechanism to intelligently adapt the ratio between NCG and PCG mutations online. In addition, information from more than two generations will be examined. Finally, implementation into other multiobjective frameworks may also be considered.

REFERENCES

[1] R. Storn and K. V. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, 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," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
[3] R. Joshi and A. C. Sanderson, "Minimal representation multisensor fusion using differential evolution," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 29, no. 1, pp. 63–76, Jan. 1999.
[4] R. Storn, "System design by constraint adaptation and differential evolution," *IEEE Trans. Evol. Comput.*, vol. 3, no. 1, pp. 22–34, Apr. 1999.
[5] A. Qing, "Dynamic differential evolution strategy and applications in electromagnetic inverse scattering problems," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 1, pp. 116–125, Jan. 2006.
[6] D. Datta and S. Dutta, "A binary-real-coded differential evolution for unit commitment problem," *Int. J. Electr. Power Energy Syst.*, vol. 42, no. 1, pp. 517–524, 2012.
[7] D. Datta and J. R. Figueira, "A real-integer-discrete-coded differential evolution," *Appl. Soft Comput.*, vol. 13, no. 9, pp. 3884–3893, 2013.

[8] L. Tang, Y. Zhao, and J. Liu, "An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 209–225, Apr. 2014.

[9] J.-h. Zhong *et al.*, "A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem," *IEEE Trans. Evol. Comput.*, vol. 17, no. 4, pp. 512–527, Aug. 2013.

[10] S.-M. Guo and C.-C. Yang, "Enhancing differential evolution utilizing eigenvector-based crossover operator," *IEEE Trans. Evol. Comput.*, vol. 19, no. 1, pp. 31–49, Feb. 2015.

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

[12] C. A. C. Coello, G. Lamont, and D. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, vol. 5. New York, NY, USA: Springer, 2007.

[13] K. Miettinen, *Nonlinear Multiobjective Optimization*, vol. 12. New York, NY, USA: Springer, 1999.

[14] A. Ponsich, A. L. Jaimes, and C. A. C. Coello, "A survey on multiobjective evolutionary algorithms for the solution of the Portfolio optimization problem and other finance and economics applications," *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 321–344, Jun. 2013.

[15] L. Tang and X. Wang, "A hybrid multiobjective evolutionary algorithm for multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 17, no. 1, pp 20–45, Feb. 2013.

[16] K. Sindhya, K. Miettinen, and K. Deb, "A hybrid framework for evolutionary multi-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 17, no. 4, pp. 495–511, Aug. 2013.

[17] E. Mezura-Montes, M. Reyes-Sierra, and C. A. C. Coello, "Multi-objective optimization using differential evolution: A survey of the state-of-the-art," in *Advances in Differential Evolution* (Studies in Computational Intelligence), vol. 143. Berlin, Germany: Springer, 2008, pp. 173–196.

[18] A. K. Qin, V. L. Huang, and P. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.

[19] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.

[20] J. Zhang and A. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

[21] R. A. Sarker, S. M. Elsayed, and T. Ray, "Differential evolution with dynamic parameters selection for optimization problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 5, pp. 689–707, Oct. 2014.

[22] A. Zhou *et al.*, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32–49, Mar. 2011.

[23] D. Zaharie and D. Petcu, "Adaptive Pareto differential evolution and its parallelization," in *Proc. 5th Int. Conf. Parallel Process. Appl. Math.*, Częstochowa, Poland, 2003, pp. 261–268.

[24] H. A. Abbass, "The self-adaptive Pareto differential evolution algorithm," in *Proc. Congr. Evol. Comput.*, vol. 1. Honolulu, HI, USA, 2002, pp. 831–836.

[25] P. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 174–188, Apr. 2003.

[26] 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," *IEE Proc. Elect. Power Appl.*, vol. 146, no. 5, pp. 577–583, Sep. 1999.

[27] B. V. Babu and M. M. L. Jehan, "Differential evolution for multi-objective optimization," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 4. Canberra, ACT, Australia, 2003, pp. 2696–2703.

[28] N. K. Madavan, "Multiobjective optimization using a Pareto differential evolution approach," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 2. Honolulu, HI, USA, 2002, pp. 1145–1150.

[29] A. W. Iorio and X. Li, "Solving rotated multi-objective optimization problems using differential evolution," in *Proc. Adv. Artif. Intell.*, vol. 3339. Cairns, QLD, Australia, 2004, pp. 861–872.

[30] A. Basak, S. Das, and K. C. Tan, "Multimodal optimization using a biobjective differential evolution algorithm enhanced with mean distance-based selection," *IEEE Trans. Evol. Comput.*, vol. 17, no. 5, pp. 666–685, Oct. 2013.

[31] S. Kukkonen and J. Lampinen, "GDE3: The third evolution step of generalized differential evolution," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, vol. 1. Edinburgh, U.K., 2005, pp. 443–450.

[32] H. A. Abbass, R. Sarker, and C. Newton, "PDE: A Pareto-frontier differential evolution approach for multi-objective optimization problems," in *Proc. Congr. Evol. Comput.*, vol. 2. Seoul, Korea, 2001, pp. 971–978.

[33] L. V. Santana-Quintero and C. A. C. Coello, "An algorithm based on differential evolution for multi-objective problems," *Int. J. Comput. Intell. Res.*, vol. 1, no. 2, pp. 151–169, 2005.

[34] H. Li and Q. Zhang, "A multiobjective differential evolution based on decomposition for multiobjective optimization with variable linkages," in *Proc. 9th Int. Conf. Parallel Prob. Solv. Nat. (PPSN)*, vol. 4193. Reykjavik, Iceland, 2006, pp. 583–592.

[35] B. Xue, W. Fu, and M. Zhang, "Differential evolution (DE) for multi-objective feature selection in classification," in *Proc. Conf. Companion Genet. Evol. Comput. Companion*, Vancouver, BC, Canada, 2014, pp. 83–84.

[36] D. Datta and J. R. Figueira, "Graph partitioning by multi-objective real-valued metaheuristics: A comparative study," *Appl. Soft Comput.*, vol. 11, no. 5, pp. 3976–3987, 2011.

[37] B. Xue, W. Fu, and M. Zhang, "Multi-objective feature selection in classification: A differential evolution approach," in *Simulated Evolution and Learning* (LNCS 8886), 2014, pp. 516–528.

[38] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. C. Coello, "A survey of multiobjective evolutionary algorithms for data mining: Part I," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 4–19, Feb. 2014.

[39] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. C. Coello, "Survey of multiobjective evolutionary algorithms for data mining: Part II," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 20–35, Feb. 2014.

[40] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, Apr. 2009.

[41] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[42] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[43] X. Qiu, J. Xu, and K. C. Tan, "A novel differential evolution (DE) algorithm for multi-objective optimization," in *Proc. Congr. Evol. Comput. (CEC)*, Beijing, China, Jul. 2014, pp. 2391–2396.

[44] S. P. Lim and H. Haron, "Performance comparison of genetic algorithm, differential evolution and particle swarm optimization towards benchmark functions," in *Proc. IEEE Conf. Open Syst. (ICOS)*, Kuching, Malaysia, Dec. 2013, pp. 41–46.

[45] T. Mantere, "A min-max DE/GA hybrid for constrained problems," in *Proc. 4th Conf. New Explor. Technol.*, Seoul, Korea, 2007, pp. 188–191.

[46] L. T. Bui *et al.*, "DMEA: A direction-based multiobjective evolutionary algorithm," *Memet. Comput.*, vol. 3, no. 4, pp. 271–285, 2011.

[47] O. Grodzevich and O. Romanko, "Normalization and other topics in multi-objective optimization," in *Proc. Fields MITACS Ind. Prob. Workshop*, Toronto, ON, Canada, 2006, pp. 89–101.

[48] K. Miettinen, *Nonlinear Multiobjective Optimization*. New York, NY, USA: Springer, 1998.

[49] J. Branke, K. Deb, K. Miettinen, and R. Slowinski, *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Berlin, Germany: Springer, 2008.

[50] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," *Struct. Multidiscipl. Optim.*, vol. 26, no. 6, pp. 369–395, 2004.

[51] I. Kim and O. de Weck, "Adaptive weighted sum method for multiobjective optimization: A new method for Pareto front generation," *Struct. Multidiscipl. Optim.*, vol. 31, no. 2, pp. 105–116, 2006.

[52] S. Kukkonen and J. Lampinen, "Ranking-dominance and many-objective optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Singapore, Sep. 2007, pp. 3983–3990.

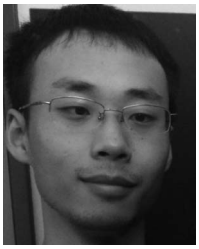
[53] K. V. Price, *New Ideas in Optimization* (An Introduction to Differential Evolution). Maidenhead, U.K.: McGraw-Hill, 1999, pp. 79–108.

[54] E. Mezura-Montes, J. Velazquez-Reyes, and C. A. C. Coello, "Modified differential evolution for constrained optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Vancouver, BC, Canada, 2006, pp. 25–32.

[55] S. Das, A. Abraham, U. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 526–553, Jun. 2009.

[56] M. Epitropakis, D. Tasoulis, N. Pavlidis, V. Plagianakos, and M. Vrahatis, "Enhancing differential evolution utilizing proximity-based mutation operators," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 99–119, Feb. 2011.

[57] C. Goh, K. Tan, D. Liu, and S. Chiam, "A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design," *Eur. J. Oper. Res.*, vol. 202, no. 1, pp. 42–54, 2010.
[58] C. Goh, Y. Ong, K. Tan, and E. Teoh, "An investigation on evolutionary gradient search for multi-objective optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Hong Kong, 2008, pp. 3741–3746.
[59] E. Ziztler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proc. Evol. Methods Design Optim. Control*, Athens, Greece, 2002, pp. 95–100.
[60] D. Zaharie, "Influence of crossover on the behavior of differential evolution algorithms," *Appl. Soft Comput.*, vol. 9, no. 3, pp. 1126–1138, Jun. 2009.
[61] Q. Zhang *et al.*, "Multiobjective optimization test instances for the CEC 2009 special session and competition," School Comput. Sci. Electr. Eng., Univ. Essex, Colchester, U.K., Tech. Rep. CES-487, 2008.
[62] L. Bradstreet, L. Barone, L. While, S. Huband, and P. Hingston, "Use of the WFG toolkit and PISA for comparison of MOEAs," in *Proc. IEEE Symp. Comput. Intell. Multicriteria Decis. Making*, Honolulu, HI, USA, 2007, pp. 382–389.
[63] C. A. C. Coello and N. C. Cortes, "Solving multiobjective optimization problems using an artificial immune system," *Genet. Program. Evolv. Mach.*, vol. 6, no. 2, pp. 163–190, 2005.
[64] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.
[65] D. Datta and J. R. Figueira, "Some convergence-based M-ary cardinal metrics for comparing performances of multi-objective optimizers," *Comput. Oper. Res.*, vol. 39, no. 7, pp. 1754–1762, 2012.
[66] G. G. Yen and Z. He, "Performance metric ensemble for multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 131–144, Feb. 2014.
[67] J. Bader and E. Zitzler, "HypE: An algorithm for fast hypervolume-based many-objective optimization," *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, Mar. 2011.
[68] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, no. 6, pp. 80–83, Dec. 1945.
[69] Y. Cai and J. Wang, "Differential evolution with neighborhood and direction information for numerical optimization," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 2202–2215, Dec. 2013.
[70] B. Li, Y.-S. Ong, M. N. Le, and C. K. Goh, "Memetic gradient search," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Hong Kong, Jun. 2008, pp. 2894–2901.
[71] A. Lara, C. A. C. Coello, and O. Schuetze, "Using gradient information for multi-objective problems in the evolutionary context," in *Proc. 12th Annu. Conf. Companion Genet. Evol. Comput. (GECCO)*, Portland, OR, USA, 2010, pp. 2011–2014.
[72] C.-M. Hong, C.-M. Chen, and H.-K. Fan, "A new gradient-based search method: Grey-gradient search method," in *Multiple Approaches to Intelligent Systems* (LNCS 1611). I. Imam, Y. Kodratoff, A. El-Dessouki, and M. Ali, Eds. Berlin, Germany: Springer, 1999, pp. 185–194.
[73] D. Zaharie, "On the explorative power of differential evolution," in *Proc. 3rd Int. Workshop Symbol. Numer. Algorithms Sci. Comput.*, Timisoara, Romania, Oct. 2001.
[74] S. Dasgupta, S. Das, A. Biswas, and A. Abraham, "On stability and convergence of the population-dynamics in differential evolution," *AI Commun.*, vol. 22, no. 1, pp. 1–20, 2009.



Xin Qiu received the B.E. degree in electrical engineering from Nanjing University, Nanjing, China, in 2012. He is currently working toward the Ph.D. degree with the Computational Intelligence Research Group, National University of Singapore, Singapore.

His research interests include differential evolution, multiobjective optimization, and applications of evolutionary algorithms.

Jian-Xin Xu (M'92–SM'98–F'11) received the Ph.D. degree from University of Tokyo, Tokyo, Japan, in 1989.

In 1991, he joined National University of Singapore, Singapore, where he is currently a Professor with the Department of Electrical Engineering. His research interests include fields of learning theory, intelligent system and control, nonlinear and robust control, robotics, and precision motion control. He has published over 200 journal papers and six books in the field of system and control. He has also supervised 30 Ph.D. students and 15 research associates, as well as conducted about 30 research grants.

Kay Chen Tan (SM'08–F'14) received the B.Eng. (First Class Hons.) degree in electronics and electrical engineering and the Ph.D. degree from University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively.

He is an Associate Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research interests include computational and artificial intelligence, with applications to multiobjective optimization, scheduling, automation, data mining, and games. He has published over 100 journal papers, over 100 papers in conference proceedings, and has co-authored five books.

Dr. Tan was a recipient of the 2012 IEEE Computational Intelligence Society Outstanding Early Career Award for his contributions to evolutionary computation in multiobjective optimization. He was the Editor-in-Chief of *IEEE Computational Intelligence Magazine* from 2010 to 2013, and is currently the Editor-in-Chief of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. He serves as an Associate Editor/Editorial Board Member on over 15 international journals, such as IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES, *Evolutionary Computation* (MIT Press), *European Journal of Operational Research*, *Journal of Scheduling*, and *International Journal of Systems Science*.

Hussein A. Abbass (SM'05) holds six academic degrees concluded with the Ph.D. degree from QUT, Brisbane QLD, Australia, in 2000.

He is a Professor of IT with University of New South Wales, Canberra, ACT, Australia. His research interests include autonomy with an aim to design next generation artificial intelligence systems that seamlessly integrate humans and machines, cognitive science, operations research, and artificial intelligence. He has published over 200 refereed articles.

Prof. Abbass is an Associate Editor of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, *IEEE Computational Intelligence Magazine*, and four other journals. He is a fellow of the U.K. Operational Research Society and the Australian Computer Society.