Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2021.017987

ARTICLE

Strengthened Initialization of Adaptive Cross-Generation Differential Evolution

Wei Wan[1], Gaige Wang[1,2,3,*] and Junyu Dong[1]

[1]Department of Computer Science of Technology, Ocean University of China, Qingdao, 266100, China

[2]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun, 130012, China

[3]Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Nanning, 530006, China

[*]Corresponding Author: Gaige Wang. Email: wgg@ouc.edu.cn

Received: 21 June 2021 Accepted: 26 September 2021

ABSTRACT

Adaptive Cross-Generation Differential Evolution (ACGDE) is a recently-introduced algorithm for solving multi-objective problems with remarkable performance compared to other evolutionary algorithms (EAs). However, its convergence and diversity are not satisfactory compared with the latest algorithms. In order to adapt to the current environment, ACGDE requires improvements in many aspects, such as its initialization and mutant operator. In this paper, an enhanced version is proposed, namely SIACGDE. It incorporates a strengthened initialization strategy and optimized parameters in contrast to its predecessor. These improvements make the direction of cross-generation mutation more clearly and the ability of searching more efficiently. The experiments show that the new algorithm has better diversity and improves convergence to a certain extent. At the same time, SIACGDE outperforms other state-of-the-art algorithms on four metrics of 24 test problems.

KEYWORDS

Differential Evolution (DE); multi-objective optimization (MO); opposition-based learning; parameter adaptation

1 Introduction

Differential Evolution, conceptualized by Storn and Price [1], is an important bionic intelligence method that is frequently used for numerical optimization. Due to its efficient and robust nature, several variants of DE have been proposed for solving single-objective optimization problems [2–4]. However, real-world problems are usually composed of multiple objectives, where increasing the value of one objective will inevitably lead to the loss of other objectives. Now many algorithms are improved in order to meet the needs of practical problems to solve the specific problem [5–8]. In multi-objective optimization problems, the difficulty of achieving Pareto optimality is attributed to the trade-off between multiple goals. Therefore, DE was proposed

to solve these problems. Initially, the DE algorithm was naturally developed for multi-objective optimization (MOO) [9–15] and practical problems [16–19] due to its superior performance.

DE has gained immense popularity for its simple structure and the ability of dealing with complex problems. However, the original DE is too simple to satisfy our current environment used in the actual problem [20–22]. In order to prevent further complications and to achieve more accurate optimization solutions, researchers have begun to improve the DE algorithm [23–25], mostly focusing on the mutant operators. DE uses three evolutionary operators, such as mutation, crossover, and selection, to guide the search to find optimal solutions. Therefore, studies typically attempted to improve these three operations to develop an improved DE, such as Generalized Differential Evolution (GDE3) [26], Neighborhood Based Differential Evolution (NDE) [27], and Composite Differential Evolution (CoDE) [28] algorithms, which all achieved excellent performance in the IEEE International Conference on Evolutionary Computation (CEC) [29]. These variants DE algorithms consistently demonstrate good performance in each years of CECs [30]. These improved variants prove that DE, despite being a marvelous metaheuristic algorithm, still has a lot of room for improvement, such as the convergence of the algorithm, the diversity, and density of the Pareto front (PF). Therefore, Adaptive Cross-Generation Differential Evolution Operators for multi-objective optimization (ACGDE) [31] was proposed in 2016.

However, few variants of the DE have considered the relationship between nondominated the sorting genetic algorithm II (NSGAII) [32]. NSGAII is an excellent algorithm for solving multi-objective problems. Its mutation GA plays a good search role in the evolution process. Like DE algorithm, NSGAII is a classical evolutionary algorithm and the way they generate new individuals has a certain randomness. And the selection of individuals does not need special processing, which makes the fusion of the two algorithms possible. The differences of population are required in DE evolution progress. However, the differences of individuals caused by DE mutation are not obvious in the early stage. Therefore, GA can be used to make up for the defects of DE algorithm. Combined with the characteristic that GA does not depend on the difference of population in the early stage, DE and NSGAII can complement each other effectively.

On the other hand, in order to enhance the difference between populations, we need to optimize the population in the early stage. Similar to most algorithms, the effect of initialization on DE performance is ignored. Specifically, ACGDE tries to estimate the promising searching directions by analyzing the differences between consecutive generations so that the mutation strategy does not work for the first few generations. And the parameters describing the relationship between individuals are not optimal, though the parameters of the algorithm are adaptive. These problems greatly affect the correctness of the solution.

Opposition-based learning (OBL), introduced by Tizhoosh [33], is inspired by the fact that everything has an opposite so that a theoretically new thing can be obtained. Since OBL is a reasonable choice between two opposing goals and removes a lot of randomness, it is suitable for use in the initialization phase of the metaheuristic algorithm and exhibits significant effects to improve convergence in the DE algorithm [34]. In the presented study, because the proposed algorithm needs a good initial stage generation more than most other evolutionary algorithms and, thus, used OBL to produce individuals in the initialization.

As an improved version, the ACGDE algorithm inherits the experiences of previous algorithms to improve DE and adds adaptive and cross-generational strategy. Two important mutation factors were considered simultaneously, such as population-based cross-generation (PCG) and neighborhood-based cross-generation (NCG), to expand the range of selected individuals while

using cross-generational crossover to realize previous generation information. At the same time, the scaling factor F and crossover probability Cr of DE were adjusted adaptively to ensure that the algorithm has a suitable evolution rate in different periods. The specific steps of ACGDE will be discussed in detail in Section 2.

Similar to most algorithms, the effect of initialization on DE performance is ignored. Specifically, ACGDE tries to estimate the promising searching directions by analyzing the differences between consecutive generations so that the mutation strategy does not work for the first few generations. And the parameters describing the relationship between individuals are not optimal, though the parameters of the algorithm are adaptive. These problems greatly affect the correctness of the solution.

In view of the nature of ACGDE, the solutions get increasingly better, while the number of evolving generations grows. The movement of solutions across generations may, therefore, provide the true PF for the next generation of evolution. However, this evolutionary strategy does not work for the early generations.

To address the above issues, we present Strengthened Initialization of Adaptive Cross-Generation Differential Evolution (SIACGDE) in attempt to eliminate the potential weaknesses of ACGDE, which lacks an obviously Pareto dominant relationship between early generations. In addition, the parameters are optimized in order to design a powerful and up-to-date version of DE. In SIACGDE, there are obviously individual differences between the two generations, that is, the offspring individuals better than the parent individuals. In this work, initial stage is divided into two parts. First, opposition-based computation learning is applied to the stochastic initial phase. Second, population-based cross-generation (PCG) and neighborhood-based cross-generation (NCG), proposed by ACGDE, are screened out in early generations. Then, we optimize the parameter neighbor size in the original ACGDE algorithm. Experiments show that parameter optimization is necessary in the paper of last section. And the results indicate that SIACGDE significantly outperforms its predecessor in terms of inverted generational distance (IGD) [35], Spread [36], and Coverage over Pareto front (CPF) [37] for 24 test problems.

This paper is further organized as follows: Section 2 presents a comprehensive review of related works. SIACGDE is described in Section 3, including the basic algorithmic framework and how the strengthened initialization promote ACGDE and individual exploration operations. Section 4 discusses the experimental settings and results, and Section 5 concludes this paper and points out directions for future work.

2 Related Works

In order to explain SIACGDE more deeply, we introduce the three algorithms mentioned in SIACGDE.

2.1 Differential Evolution

Differential Evolution (DE) was originally an excellent algorithm for single-object numerical optimization, inspired by the simulated annealing algorithm. DE is based on the principle of natural selection to make choices, like other evolutionary algorithms, using three unique operations (mutation, crossover, and selection) to guide the individuals to find promising solutions. Research has shown that DE show dominance in multi-objective problems [38]. And the ranks of competitions over the years proved its performance [30]. The main steps to produce new individuals in basic DE are as follows:

Mutation: DE will create a mutant vector V_j^t for each target vector X_j^t at generation t as:

$$V_j^t = X_{r1}^t + F \cdot \left(X_{r2}^t - X_{r3}^t\right) \tag{1}$$

where the scaling factor F is a certain constant, which is randomly selected from 0 to 1; and $r1, r2, r3 \in \{1, 2, \ldots, NP\}$ are randomly selected indexes of individual in the current generation. $j \in \{1, 2, \ldots, NP\}$ is the index of the parent.

Crossover: The purpose of crossover is to randomly generate trial vectors $U_j^t = \{u_{1,j}^t, u_{2,j}^t, \ldots, u_{D,j}^t\}$. The value of Cr, or crossover probability, is from 0 to 1. After mutation, crossover chooses a vector between the target vector $\{X_j^t = x_{1,j}^t, x_{2,j}^t, \ldots, x_{D,j}^t\}$ and mutant vector $V_j^t = \{v_{1,j}^t, v_{2,j}^t, \ldots, v_{D,j}^t\}$ using a crossover probability Cr.

$$u_{i,j}^t = \begin{cases} v_{i,j}^t & if \quad rand_j \le Cr \\ x_{i,j}^t & otherwise \end{cases} \tag{2}$$

where $i = \{1, 2, \ldots, D\}$; and $Cr \in [0, 1]$.

Selection: In DE, the strategy of greedy selection is adopted, that is, the better individuals are selected as the new individuals. In the minimization problem, X_j^{t+1} is defined by

$$X_j^{t+1} = \begin{cases} U_j^t & if \quad f\left(U_j^t\right) \le f\left(X_j^t\right) \\ x_j^t & otherwise \end{cases} \tag{3}$$

2.2 Overview of ACGDE Components

ACGDE has been improved in many aspects on the basis of DE, such as its mutant operators, which are subsequently discussed in detail.

1) CGA: The continued development of DE has been witnessed in its excellent performance and expanded application scenarios. Due to the fact that the converging direction and efficiency of the population evolution are greatly influenced by parameters, parameter adaptation is undoubtedly one of most important improvements of DE. Such adaptation mechanism makes the parameters of the algorithm more suitable for the current environment in the iteration process. Similar to all adaptive DE algorithms, two main control parameters, the scaling factor F and crossover probability Cr, are adaptive in ACGDE. This new adaptation mechanism is called CGA, which is defined as follows.

During initialization, F_i and Cr_i are randomly generated from $[F_{\min}, F_{\max}]$ and $[Cr_{\min}, Cr_{\max}]$ for each individual X_i, respectively. Whereafter, the parameter values of the offspring will be produced by the following formulas:

$$F_{spring,i,g} = F_{neighbor,i,g} + \theta_F \cdot Gaussian\,(0, 1) \tag{4}$$

$$Cr_{spring,i,g} = Cr_{neighbor,i,g} + \theta_{Cr} \cdot Gaussian\,(0, 1) \tag{5}$$

where $Gaussian$ (0, 1) is a random number from a Gaussian distribution, where the mean is 0 and the standard deviation is 1. The coefficients θ_F and θ_{Cr} are pre-input constants for controlling the outputs of Gaussian distribution. $F_{neighbor,i,g}$ and $Cr_{neighbor,i,g}$ are generated by:

$$F_{neighbor,i,g} = mean_A\left(NF_{i,g} \cup NF_{i,g-1}\right) \tag{6}$$

$$Cr_{neighbor,i,g} = mean_A\left(NCr_{i,g} \cup NCr_{i,g-1}\right) \tag{7}$$

According to the ranking of individual distances around individuals, X_i generates a small scale factor called neighbor. $NF_{i,g}$ and $NCr_{i,g}$ indicate the factors F of X_i 's neighbor and crossover probability Cr of X_i 's neighbor at generation g, respectively. If the result of F_{neighbor} or Cr_{neighbor} is beyond $[F_{\min}, F_{\max}]$ or $[Cr_{\min}, Cr_{\max}]$, its value is the nearest bound.

2) NCG and PCG mutant operators: In the past, DE variants hardly involved the relationship between generations, which newly generated offspring are likely better than the parents. The information difference between the two generations is suitable for DE algorithm mining. Hence, two strategies are proposed by ACGDE, namely NCG and PCG. First, NCG is defined as follows:

$$V_{i,g} = X_{rn1,g} + F \cdot \left(X_{rn1,g} - X_{rn2,g-1}\right) \tag{8}$$

where i and g are the index of the main parent and current generation, respectively; $V_{i,g}$ denotes the new generated vector; $X_{rn1,g}$ is a randomly selected individual from the current population neighborhood of its parent; index $rn1$ is randomly selected from $\{I_{i,g,1}, I_{i,g,2}, I_{i,g,3}, \ldots, I_{i,g,T}\}$, where T is the *Neighborhood* size; and $X_{rn2,g\text{-}1}$ is a randomly selected individual from the previous generation neighborhood of its parent. In a similar way, the index $rn2$ is randomly selected from $\{I_{i,g\text{-}1,1}, I_{i,g\text{-}1,2}, I_{i,g\text{-}1,3}, \ldots, I_{i,g\text{-}1,T}\}$.

The definition of PCG operation, which utilizes information from two successive generations but from a wider range than NCG uses in the same generation, is as follows:

$$V_{i,g} = X_{i,g} + F \cdot \left(X_{rp1,g} - X_{rp2,g-1}\right) \tag{9}$$

where $V_{i,g}$ donates the new generated vector; i and g are the number of the main parent and the index of current generation, respectively; the individual $X_{rp1,g}$ is randomly selected from the whole current population; and $rp1$ is an index randomly chosen randomly from 1 to N, where N is the population size. In a similar way, $X_{rp2,g\text{-}1}$ is randomly selected from the whole previous generation; and $rp2$ is an index chosen randomly from 1 to N.

2.3 Opposition-Based Learning (OBL)

In light of the concept of Yin and Yang originating from ancient Chinese philosophy [39], some physicists speculate the existence of white holes and antimatter considering the fact that such objects may have opposites. In the field of numerical optimization, the performance of opposition-based learning (OBL), proposed by Tizhoosh [33], has been well verified. In this work, OBL was added to the initialization to enhance the convergence of ACGDE. For OBL, opposite number, extended definition for D space, and opposition-based optimization are straightforwardly defined as follows.

Definition 1. (Opposite number) [32] Let $x \in [a, b]$ be a real number, then opposite $\breve{x}$ is defined as follows:

$$\breve{x} = a + b - x \tag{10}$$

Definition 2. (Opposite point in the D space) [33] Let $x = (x_1, x_2, \ldots, x_D)$ be a point in D space. $\breve{x} \in [a_i, b_i]$, i is from 1 to D. The opposite of x is defined as follows:

$$\breve{x}_i = a_i + b_i - x_i \tag{11}$$

Definition 3. (Opposition-based optimization) According to the definition of opposite point in the D space, the point $P = (x_1, x_2, \ldots, x_D)$of opposite point $\breve{P} = \left(\breve{x}_1, \breve{x}_2, \ldots, \breve{x}_D\right)$ is generated. $f(\cdot)$ is defined as the fitness function to measure the fitness of the candidate. Then, the point with the largest value between $f\left(\breve{P}\right)$ and $f(P)$ is chosen. In other words, according to fitness, we can select one point from each pair of opposite points.

3 Strengthened Initialization of Adaptive Cross-Generation Differential Evolution

In this section, an enhanced version of the ACGDE algorithm with strengthened initialization is proposed. The detailed initialization and the whole process of SIACGDE are introduced in the following subsections.

3.1 Initialization

As mentioned above, OBL can accelerate the evolutionary process without any prior knowledge. This advantage makes up for the slow progress of the ACGDE algorithm in the early generations. Therefore, two main steps are distinguishable for the original ACGDE, including initialization and the different mutant operator in the first five percent generations. Like most algorithms utilizing OBL for optimization, initialization in this algorithm is divided into four steps:

1) A population P with N individuals is randomly generated.

2) According to P produces its opposite population $\breve{P}$.

3) The candidate's fitness is measured, which is performed for all P and $\breve{P}$ individuals.

4) The first N individuals with the best fitness are selected as the first generation.

The corresponding pseudocode is given in Algorithm 1.

Algorithm 1: Opposition-Based Learning Initialization

Input: the individuals in population, N;
the problem dimension, D;
Output: Optimized POP_0;
1: Generate uniformly distributed random population POP_0;
2: **for** $i = 1{:}N$ **do**
3: **for** $j = 1{:}N$ **do**
4: $OPOP_{0_{i,j}} = a_j + b_j - POP_0$;
5: **end for**
6: **end for**
7: Select N fittest individuals from set the $\{POP_0, OPOP_0\}$ as initial population POP_0.

When the initialization provides better individuals that are closer to the Pareto frontier, we can iterate over them. Since ACGDE also implements nondominated the sorting genetic algorithm II (NSGAII) [32] in its multi-objective framework to select the next generation, the second step is the mutant operator of NSGAII, genetic algorithm (GA), replace the both mutant operators PCG and NCG in the first five percent generations. The experiment discussed in Section 4 will

show that this replacement step is very effective in improving the convergence speed and finding the Pareto front.

Although we have optimized the parameters of the algorithm at the same time, the experiments in the later section show that the strengthened initialization is the part with the biggest improvement. Therefore, the optimization of initialization is the most important step for SIACGDE.

3.2 Improving ACGDE with Strengthened Initialization

Here, the paper will focus on how the strengthened initialization and ACGDE mutant operator interact with each other. With better initial conditions, the cross-generation mutant operators, NCG and PCG, begin to play a stronger role than its predecessor in SIACGDE algorithm. The flow of SIACGDE is presented in Algorithm 2.

At the beginning, the individuals will be iterated rapidly in previous generations by NSGAII, and the obtained offspring can be used in SIACGDE. According to our experiment, it can be concluded that the iteration speed of NSGAII is obviously faster than the original algorithm in the initial stage. Furthermore, the addition of OBL makes the first few generations of individuals significantly better than the random initial individuals. In the process of evolution, GA mutates through two random parents, which does not depend on the individual differences between the two parents. This is actually an advantage in the early stage of evolution, compared with mutation NCG and PCG. If the previous operators are still used at early stage of evolution, the characteristics of these two operators will not play out. After several generations, the connection between populations is slowly established. At this time, this connection can not be used effectively by GA operator. Therefore, each generation is better than the previous generation, which can exactly match the cross-generation mechanism. This step is mainly for the two mutation NCG and PCG to obtain more obvious difference between generations.

NCG is a mutation operator in which all mutated individuals are selected from the range of the *Neighborhood*. NCG has a narrower search range than PCG so that the individuals produced by NCG are more targeted and sensitive to generational differences. On the other hand, PCG has a wider search space, maintaining population diversity while avoiding a purely stochastic exploration. Both NCG and PCG mutations have a 50% chance of being used, which achieves a tradeoff between exploitation and exploration.

Then, the process is repeated until the termination condition is reached. Algorithm 3 shows the steps of the original ACGDE.

Algorithm 2: SI-ACGDE

1: Generate uniformly distributed random population POP_0 with size N and set the problem dimension D;
2: $POP_0 = \mathrm{OBL}(N, D, POP_0)$;
3: **for** $g = 1{:}5\% \cdot G_{max}$ **do**
4: Utilize GA mutation to generate the new population POP_{spring};
5: Perform Survival selection of NSGAII;
6: $g = g + 1$;
7: Update Current Population POP_g;
8: **end for**
9: $POP = \mathrm{ACGDE}(g, POP_g, N, 20\% \cdot N)$.

Algorithm 3: Procedure of ACGDE

Input: the index of generations, g_0;
the population, POP;
the generation size, N;
the Neighborhood size, T;
Output: optimal solution POP;

Set the control parameters of ACGDE: θ_F, θ_{Cr}, F_{min}, F_{max}, Cr_{min}, and Cr_{max};
Set the associated parameters $\{F_{1,g_0}, F_{2,g_0}, \ldots, F_{N,g_0}\}$ and $\{Cr_{1,g_0}, Cr_{2,g_0}, \ldots, Cr_{N,g_0}\}$. The generated parameters values should be within $[F_{min}, F_{max}]$ and $[Cr_{min}, Cr_{max}]$, respectively;
for $g = g_0 : G_{max}$ **do**
  Calculate the sub-rank for each individual in POP_g;
  for $j = 1{:}N$ **do**
    Determine the neighborhood of current generation and previous generation for individual $X_{j,g}$ in terms of sub-rank;
    Perform CGS mechanism to produce $F_{spring,j,g}$ and $Cr_{spring,j,g}$;
    If $rand[0, 1.0] \leq 0.5$ **then**
      Utilize NCG mutation with $F_{spring,j,g}$ to generation the mutant vector $V_{j,g}$;
    else
      Utilize PCG mutation with $F_{spring,j,g}$ to generation the mutant vector $V_{j,g}$;
    end if
    Perform binomial recombination $Cr_{spring,j,g}$ on $X_{j,g}$ and $V_{j,g}$ to generate the offspring $U_{i,g}$;
  end for
  Perform the survival selection of the MO framework;
  $g = g + 1$;
  Update current population POP_g;
end for

4 Experiments

In this section, we discuss the performance of SIACGDE in the experiments. To fairly show performance comparisons, the same parameters are used in both algorithms, except for the change in the size of the *Neighborhood*. And the reason is described in the Subsection 3. Here, the experiment is presented in three subsections. First, the metrics used in this experiment are introduced in Subsection 1. Then, the comparison of seven common and state-of-the-art algorithms, including ACGDE, with the proposed SIACGDE algorithm on 24 frequently used MO benchmark problems (ZDT [40], DTLZ [41], VNT [42], and WFG [43]) is discussed in Subsection 2. The last subsection describes the tested effectiveness of the components of SIACGDE.

4.1 Performance Metrics

To date, there are a number of metrics to quantify the optimality of a solution set and the performance of multi-objective evolutionary algorithms (MOEAs) in multi-objective optimization (MOO). Each metric has its own set of performance criteria, which are divided into four categories: Capacity, Convergence, Diversity, and Convergence-Diversity. In this study, three metrics, including inverted generational distance (IGD) [35], Spread [36], Coverage over Pareto front (CPF) [37], and hypervolume (HV) [44], are selected to evaluate the performance of the algorithm in terms of direction diversity and convergence-diversity. Among them, IGD is the most common and effective metrics to evaluate the performance of the algorithm.

4.2 The Performance of Strengthened Initialization of Adaptive Cross-Generation Differential Evolution

To evaluate the competitiveness of the algorithm, we selected six state-of-the-art algorithms in addition to ACGDE for comparison. These include SIACGDE, ACGDE, and NSGAII, which were introduced earlier, and two common improved DE algorithms, GDE3 [26] and MOEADDE [45]. Other algorithms to be compared are LCSA [46], LMEA [47], and the improved version SPEA2SDE [48] of SPEA2 [49], which have been released in recent years. These seven algorithms were chosen for two reasons. First, some of algorithms, like MOEADDE, are variants that utilize the DE characteristics, and the effect of the comparison can more directly explain the validity of the proposed algorithm. Second, all improved algorithms are superior to their respective original algorithm in most problems. The experimental results of metrics are shown in Tables 1, 2 and Appendix A, respectively, where the best performance on each problem is bolded.

And the results were 30 independently run tests on each test problem for each MOEA algorithm. And the termination condition of eight algorithms is reaching maximum evolution of 10,000.

The values IGD of these seven algorithms for test functions can be shown in Table 1, from which it can be concluded that SIACGDE is the best algorithm on the most test functions. The relatively new algorithm LMEA follows as the second best. The convergence and diversity performance of the LMEA and LCSA algorithm are more excellent other algorithms except for SIACGDE which can be seen from the values of IGD. The reason why SIACGDE does not have a large advantage in some problems is mainly because the algorithm's ability to deal with more complex Pareto frontier problems or problems with too many decision variables is not far beyond the sate-of-the-art algorithms. However, SIACGDE has achieved a good balance between search ability and convergence ability. Its main goal is to effectively infer the Pareto frontier position based on the information of two generations in multi-objective problems. Therefore, in most problems, SIACGDE have achieved best value of IGD in eight algorithms.

Table 1: Mean and standard deviation of the values IGD

Problem	M	GDE3	MOEADDE	NSGAII	LCSA	LMEA	SPEA2SDE	ACGDENSGAII	SIACGDE
VNT1	3	1.4863e − 1 (5.82e − 3) −	1.8309e − 1 (3.72e − 3) −	1.5733e − 1 (8.92e − 3) −	1.6604e − 1 (8.76e − 3) −	1.6719e − 1 (1.04e − 2) −	1.4709e − 1 (9.11e − 3) −	1.6256e − 1 (9.78e − 3) −	**1.2463e − 1 (2.60e − 3)**
VNT2	3	1.9500e − 2 (1.33e − 3) −	4.5856e − 2 (2.02e − 3) −	2.1276e − 2 (1.35e − 3) −	4.6614e − 2 (1.73e − 2) −	3.1701e − 2 (3.18e − 3) −	3.3974e − 2 (4.82e − 3) −	2.2588e − 2 (1.53e − 3) −	**1.2613e − 2 (4.38e − 4)**
VNT3	3	3.5789e − 2 (1.62e − 3) −	7.4637e − 1 (2.06e − 1) −	4.1020e − 2 (3.03e − 3) −	4.0975e − 1 (2.29e − 1) −	4.7143e − 2 (2.86e − 3) −	3.1026e − 1 (3.91e − 1) −	4.7454e − 2 (2.75e − 3) −	**3.1989e − 2 (3.12e − 3)**
VNT4	3	1.6878e − 1 (1.17e − 2) +	1.7716e + 0 (1.83e − 2) −	**1.5837e − 1 (1.13e − 2) +**	2.4646e − 1 (6.33e − 2) +	1.1262e + 0 (5.87e − 2) −	1.2892e + 0 (9.71e − 2) −	1.6922e − 1 (1.36e − 2) +	1.0586e + 0 (5.48e − 2)
WFG1	3	1.7454e + 0 (3.37e − 2) −	1.5538e + 0 (2.89e − 2) =	9.7203e − 1 (1.01e − 1) +	1.4164e + 0 (6.24e − 2) +	2.1652e + 0 (8.51e − 2) −	**7.5561e − 1 (9.15e − 2) +**	1.5774e + 0 (7.65e − 2) =	1.5366e + 0 (8.60e − 2)
WFG2	3	2.8458e − 1 (1.34e − 2) −	3.5741e − 1 (2.17e − 2) −	2.2539e − 1 (1.08e − 2) −	1.9848e − 1 (1.17e − 2) =	3.3906e − 1 (6.72e − 2) −	2.0770e − 1 (1.12e − 2) −	2.4585e − 1 (1.47e − 2) −	**1.9261e − 1 (1.01e − 2)**
WFG3	3	3.6291e − 1 (3.04e − 2) −	2.8588e − 1 (4.29e − 2) −	1.2993e − 1 (1.48e − 2) +	2.2058e − 1 (2.62e − 2) =	**3.4083e − 2 (1.09e − 2) +**	7.4598e − 2 (7.14e − 3) +	2.2440e − 1 (3.38e − 2) =	2.2401e − 1 (2.00e − 2)
WFG4	3	3.5795e − 1 (1.09e − 2) −	3.9845e − 1 (1.39e − 2) −	2.8143e − 1 (9.94e − 3) −	2.4739e − 1 (3.71e − 3) =	2.8696e − 1 (3.73e − 2) −	3.2620e − 1 (1.52e − 2) −	3.1193e − 1 (1.03e − 2) −	**2.4542e − 1 (6.60e − 3)**
WFG5	3	2.7651e − 1 (1.28e − 2) −	3.3628e − 1 (3.59e − 3) −	2.8818e − 1 (1.07e − 2) −	2.4460e − 1 (2.26e − 3) −	2.4425e − 1 (1.29e − 2) −	3.2974e − 1 (1.76e − 2) −	2.7462e − 1 (9.73e − 3) −	**2.1954e − 1 (2.09e − 3)**
WFG6	3	3.8259e − 1 (1.95e − 2) −	4.3594e − 1 (2.22e − 2) −	3.3000e − 1 (1.78e − 2) =	3.3775e − 1 (2.70e − 2) =	**2.5708e − 1 (1.35e − 2) +**	3.5611e − 1 (1.91e − 2) −	3.9231e − 1 (2.16e − 2) −	3.3005e − 1 (1.69e − 2)
WFG7	3	3.7149e − 1 (1.19e − 2) −	3.8492e − 1 (1.49e − 2) −	2.8177e − 1 (1.14e − 2) −	2.5822e − 1 (8.61e − 3) −	**2.4190e − 1 (6.70e − 3) +**	3.2507e − 1 (1.58e − 2) −	3.3227e − 1 (1.89e − 2) −	2.5074e − 1 (7.07e − 3)
WFG8	3	4.8305e − 1 (1.67e − 2) −	4.8505e − 1 (2.60e − 2) −	3.7766e − 1 (1.11e − 2) −	3.5792e − 1 (1.07e − 2) −	3.5559e − 1 (1.94e − 2) −	3.6489e − 1 (1.13e − 2) −	4.1494e − 1 (1.28e − 2) −	**3.4255e − 1 (7.47e − 3)**
WFG9	3	3.9682e − 1 (2.39e − 2) −	3.4876e − 1 (1.05e − 2) −	2.8313e − 1 (1.36e − 2) −	2.7494e − 1 (2.94e − 2) =	**2.4636e − 1 (1.36e − 2) =**	3.0684e − 1 (1.85e − 2) −	3.8478e − 1 (2.16e − 2) −	2.7643e − 1 (4.90e − 2)
DTLZ1	3	4.7230e + 0 (2.37e + 0) −	8.0096e − 1 (1.00e + 0) +	2.8234e − 1 (2.80e − 1) +	1.1343e + 0 (5.72e − 1) +	**3.4378e − 2 (4.92e − 2) +**	1.7428e − 1 (1.54e − 1) +	1.4780e + 0 (1.08e + 0) =	1.6547e + 0 (8.79e − 1)
DTLZ2	3	8.2221e − 2 (2.87e − 3) −	7.7747e − 2 (1.08e − 3) −	6.9602e − 2 (2.05e − 3) −	5.8403e − 2 (1.05e − 3) −	5.7224e − 2 (1.89e − 3) −	7.6377e − 2 (2.11e − 3) −	7.0173e − 2 (2.59e − 3) −	**5.6362e − 2 (8.36e − 4)**
DTLZ3	3	2.8245e + 1 (3.01e + 1) =	2.4663e + 1 (2.68e + 1) =	8.0783e + 0 (3.58e + 0) +	3.5322e + 1 (1.04e + 1) −	**5.5230e + 0 (1.20e + 0) +**	8.3273e + 0 (4.94e + 0) +	3.9442e + 1 (1.45e + 1) −	2.8848e + 1 (1.12e + 1)
DTLZ4	3	1.0927e − 1 (1.96e − 2) −	1.7462e − 1 (7.13e − 2) −	8.4867e − 2 (8.63e − 2) −	5.9820e − 2 (1.39e − 3) −	3.5164e − 1 (1.07e − 1) −	1.3918e − 1 (1.61e − 1) −	7.0611e − 2 (2.34e − 3) −	**5.6427e − 2 (9.44e − 4)**
DTLZ5	3	9.0909e − 3 (8.54e − 4) −	1.4360e − 2 (2.83e − 4) −	6.0132e − 3 (2.35e − 4) +	1.3822e − 2 (1.57e − 3) −	**5.4543e − 3 (4.39e − 4) +**	1.0016e − 2 (1.27e − 3) −	6.8304e − 3 (4.47e − 4) =	7.0664e − 3 (6.95e − 4)
DTLZ6	3	1.0593e + 0 (6.49e − 1) −	1.4073e − 2 (1.95e − 4) −	6.0103e − 3 (3.89e − 4) −	1.9972e − 2 (1.84e − 3) −	7.9189e − 1 (3.67e − 1) −	1.0060e − 2 (2.18e − 3) −	6.7055e − 3 (4.11e − 4) −	**4.0875e − 3 (2.84e − 5)**
ZDT1	2	8.5193e − 1 (1.12e − 1) −	4.3540e − 1 (1.17e − 1) −	1.2231e − 2 (1.98e − 3) −	4.1716e − 3 (3.88e − 4) −	2.2423e + 0 (2.06e − 1) −	1.0466e − 2 (1.87e − 3) −	6.2812e − 3 (3.47e − 4) −	**3.7877e − 3 (3.71e − 5)**
ZDT2	2	1.8384e + 0 (2.12e − 1) −	7.1837e − 1 (1.39e − 1) −	2.8127e − 2 (5.00e − 2) −	1.1065e − 2 (1.22e − 2) −	3.5480e + 0 (2.61e − 1) −	1.6330e − 2 (4.44e − 3) −	6.3051e − 3 (3.80e − 4) −	**3.8543e − 3 (3.16e − 5)**
ZDT3	2	9.1497e − 1 (1.28e − 1) −	5.0854e − 1 (9.10e − 2) −	1.3545e − 2 (9.63e − 3) −	2.1206e − 2 (1.49e − 2) −	1.9628e + 0 (1.96e − 1) −	9.9197e − 3 (4.11e − 3) −	7.1517e − 3 (2.81e − 4) −	**4.7478e − 3 (1.60e − 4)**
ZDT4	2	3.2734e + 1 (5.52e + 0) −	3.2828e + 0 (1.42e + 0) −	2.7115e − 1 (1.22e − 1) +	2.7563e + 0 (1.27e + 0)−	5.4520e − 1 (1.82e − 1) +	**2.1149e − 1 (1.89e − 1) +**	2.1177e + 0 (1.08e + 0) =	1.7511e + 0 (7.61e − 1)
ZDT6	2	1.4138e − 1 (2.13e − 1) −	7.8136e − 2 (1.52e − 1) −	5.8193e − 2 (3.01e − 2) −	3.5168e − 3 (4.41e − 4) −	4.8868e − 1 (1.99e − 1) −	6.2477e − 2 (4.17e − 2) −	5.2366e − 3 (2.19e − 4) −	**3.0803e − 3 (2.18e − 5)**
+/-/=		0/23/1	1/21/2	7/16/1	3/16/5	7/16/1	5/19/0	1/18/5	

In order to show the performance of the algorithms in more angles, two metrics that are rarely considered in most experiments are also added in this experiment. As shown in Table A1 (the first table in Appendix A), the indicator Spread of SIACGDE performed best on 37 benchmarks, and SIACGDE has strong competitiveness on MOP. Finally, the experimental results from using the metric CPF in Table A2 (the second table in Appendix A) reveal that it is as dominant as the other two metrics in a total of 8 algorithms, also demonstrating strong performance of SIACGDE in coverage over the Pareto front.

The bold font represents the best performance on this problem. Where the symbols '+', '−' and '=' indicate the results of compared algorithm is significantly better, significantly worse, and statistically similar to that obtained by SIACGDED at a 0.05 level by the Wilcoxon's rank sum test, respectively.

Among the metrics that show convergence-diversity performance, the most effective and popular is HV. And we showed the performance comparison of eight algorithms in Table A3 (the third table in Appendix A). As the same of previous results of three metrics, SIACGDE is the best performing one of the eight algorithms. Especially in a two-dimensional function which the real Pareto front is a curve, the effect of SIACGDE is the most obvious. Due to its cross-generation search mechanism, the problem ZDT1-2 is solved very well. And the cross-generation mechanism will illustrate its advantages in the next subsection. From multiple perspectives, it can be concluded that SIACGDE is an algorithm with excellent solving ability.

In addition, we compared the runtime of eight algorithms on VNT1∼4 [42] in Table 2. Among them, the shortest runtime is NSGAII. This shows that NSGAII as an excellent classic algorithm, it is relatively simple operators, which greatly saves calculation time. But at the same time, the performance of NSGAII is no longer competitive. The runtime of GDE3, LCSA, and LMEA are in same order of magnitude, and their speed are slower than NSGAII. This is because the algorithm has a certain improvement in searching or selection, which makes the algorithm has better performance and leads to a long running time. The time complexity of SPEA2SDE of selection mechanism is too high, resulting in that SPEA2SDE is the slowest algorithm among the eight algorithms. Although the runtime of SIACGDE and ACGDE are different, the gap is not big. It is due to the fact that the two algorithms use same mutant operators in the middle and late stages of evolution, but the performance of the algorithm has been greatly improved due to the enhancement of the initial stage by SIACGDE. These can explain that under the same computing resources, SIACGDE can get better the convergence and diversity of the population.

In this experiment, the original algorithm ACGDE is based on NSGAII framework to select offspring. By comparing ACGDE with SIACGDE, we can see that the new algorithm has been greatly improved in terms of both diversity and finding solutions due to the difference between successive generations generated in the initial stage. In the first few generations of the original algorithm, the use of cross-generation information to make mutations could not find an effective way forward. At this point, the original mutation operator can be replaced by genetic operator to make better use of excellent individuals for searching. At the same time, opposition-based learning is applied to provide more excellent individuals for genetic operator and, thus, differences between individuals when the algorithm produces cross-generational replacement.

Table 2: Mean and standard deviation of the values HV

Problem	M	GDE3	MOEADDE	NSGAII	LCSA	LMEA	SPEA2SDE	ACGDENSGAII	SIACGDE
VNT1	3	3.3556e − 1	3.3358e − 1	3.3430e − 1	3.3308e − 1	3.3291e − 1	3.3297e − 1	3.3373e − 1	**3.3737e − 1**
		(4.11e − 4) −	(2.34e − 4) −	(5.93e − 4) −	(1.19e − 3) −	(1.20e − 3) −	(1.44e − 3) −	(7.81e − 4) −	**(2.69e − 4)**
VNT2	3	3.3216e − 1	3.3090e − 1	3.3173e − 1	3.3336e − 1	3.3071e − 1	**3.3403e − 1**	3.3136e − 1	3.3302e − 1
		(2.60e − 4) −	(9.65e − 5) −	(2.84e − 4) −	(2.58e − 3) =	(4.75e − 4) −	**(7.69e − 4) +**	(3.12e − 4) −	(9.14e − 5)
VNT4	3	**2.5811e − 1**	2.0967e − 1	2.5748e − 1	2.5650e − 1	2.4082e − 1	2.3497e − 1	2.5726e − 1	2.4129e − 1
		(2.37e − 4) +	(1.84e − 3) −	(4.57e − 4) +	(5.98e − 4) +	(3.61e − 3) =	(4.64e − 3) −	(4.34e − 4) +	(3.50e − 3)
DTLZ1	3	0.0000e + 0	2.5207e − 1	3.7401e − 1	2.5425e − 3	**7.9700e − 1**	4.8542e − 1	5.2865e − 2	1.9110e − 2
		(0.00e + 0) −	(3.42e − 1) +	(3.35e − 1) +	(8.73e − 3) =	**(1.16e − 1) +**	(3.24e − 1) +	(1.86e − 1) =	(6.68e − 2)
DTLZ2	3	4.8579e − 1	5.1497e − 1	5.2977e − 1	5.4412e − 1	5.4192e − 1	**5.6080e − 1**	5.2004e − 1	5.4712e − 1
		(5.74e − 3) −	(4.79e − 3) −	(4.26e − 3) −	(2.69e − 3) −	(6.12e − 3) −	**(9.20e − 4) +**	(5.25e − 3) −	(2.44e − 3)
DTLZ3	3	2.2170e − 2	**3.2756e − 2**	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0
		(8.54e − 2) =	**(9.86e − 2) +**	(0.00e + 0) =	(0.00e + 0) =	(0.00e + 0) =	(0.00e + 0) =	(0.00e + 0) =	(0.00e + 0)
DTLZ4	3	4.9464e − 1	5.0047e − 1	5.2509e − 1	5.4268e − 1	3.7115e − 1	5.3191e − 1	5.1937e − 1	**5.4753e − 1**
		(1.06e − 2) −	(2.48e − 2) −	(3.46e − 2) −	(3.60e − 3) −	(6.59e − 2) −	(7.38e − 2) −	(5.39e − 3) −	**(2.51e − 3)**
DTLZ6	3	3.4264e − 2	1.9504e − 1	1.9938e − 1	1.9226e − 1	6.5324e − 3	1.9915e − 1	1.9901e − 1	**2.0008e − 1**
		(7.16e − 2) −	(1.05e − 4) −	(1.56e − 4) −	(1.26e − 3) −	(3.49e − 2) −	(2.17e − 3) −	(2.26e − 4) −	**(3.62e − 5)**
ZDT1	2	2.3445e − 2	2.5080e − 1	7.0661e − 1	7.1991e − 1	0.0000e + 0	7.0881e − 1	7.1799e − 1	**7.2060e − 1**
		(2.02e − 2) −	(9.70e − 2) −	(2.73e − 3) −	(4.07e − 4) −	(0.00e + 0) −	(2.63e − 3) −	(3.67e − 4) −	**(2.90e − 5)**
ZDT2	2	0.0000e + 0	4.7134e − 3	4.1355e − 1	4.3691e − 1	0.0000e + 0	4.2470e − 1	4.4272e − 1	**4.4511e − 1**
		(0.00e + 0) −	(1.19e − 2) −	(4.54e − 2) −	(1.34e − 2) −	(0.00e + 0) −	(7.43e − 3) −	(3.37e − 4) −	**(3.82e − 5)**
ZDT3	2	3.5922e − 2	2.4763e − 1	6.0197e − 1	**6.0899e − 1**	0.0000e + 0	5.9310e − 1	5.9907e − 1	5.9979e − 1
		(3.78e − 2) −	(8.18e − 2) −	(2.73e − 2) +	**(3.72e − 2) +**	(0.00e + 0) −	(6.41e − 3) −	(1.44e − 4) −	(2.38e − 4)
ZDT4	2	0.0000e + 0	0.0000e + 0	4.7546e − 1	0.0000e + 0	1.6205e − 1	**4.9899e − 1**	1.5596e − 2	1.1261e − 2
		(0.00e + 0) −	(0.00e + 0) −	(1.12e − 1) +	(0.00e + 0) −	(1.03e − 1) +	**(1.73e − 1) +**	(5.06e − 2) =	(4.03e − 2)
ZDT6	2	3.2134e − 1	3.3490e − 1	3.1506e − 1	3.8824e − 1	5.6432e − 2	3.1161e − 1	3.8678e − 1	**3.8891e − 1**
		(1.01e − 1) −	(9.45e − 2) −	(3.70e − 2) −	(5.93e − 4) −	(8.44e − 2) −	(4.73e − 2) −	(2.26e − 4) −	**(1.55e − 5)**
+/-/=		1/11/1	2/11/0	4/8/1	2/8/3	2/9/2	4/8/1	1/9/3	

4.3 Optimization of Algorithm Parameters and Convergence Rate

In this subsection, we describe the experiments that were performed to verify the effectiveness of the cross-generation mutation proposed in SIACGDE. That is to say, we used the adjacent individuals of the parent and to select offspring. The validity of this strategy must be carried out under the premise that there is an obvious evolution in the two generations, so we suspended the experiment in the middle of evolution to observe. When SIACGDE ran the process of ZDT1 solution, we randomly selected two generations of experimental data. According to the results shown in Fig. 1, it is obvious that almost all individuals in the offspring are better than or equal to the parent individuals, and almost all vectors generated by the difference between them converge towards the Pareto front. Each generation moves forward, as described above, and eventually reaches the optimal solution.

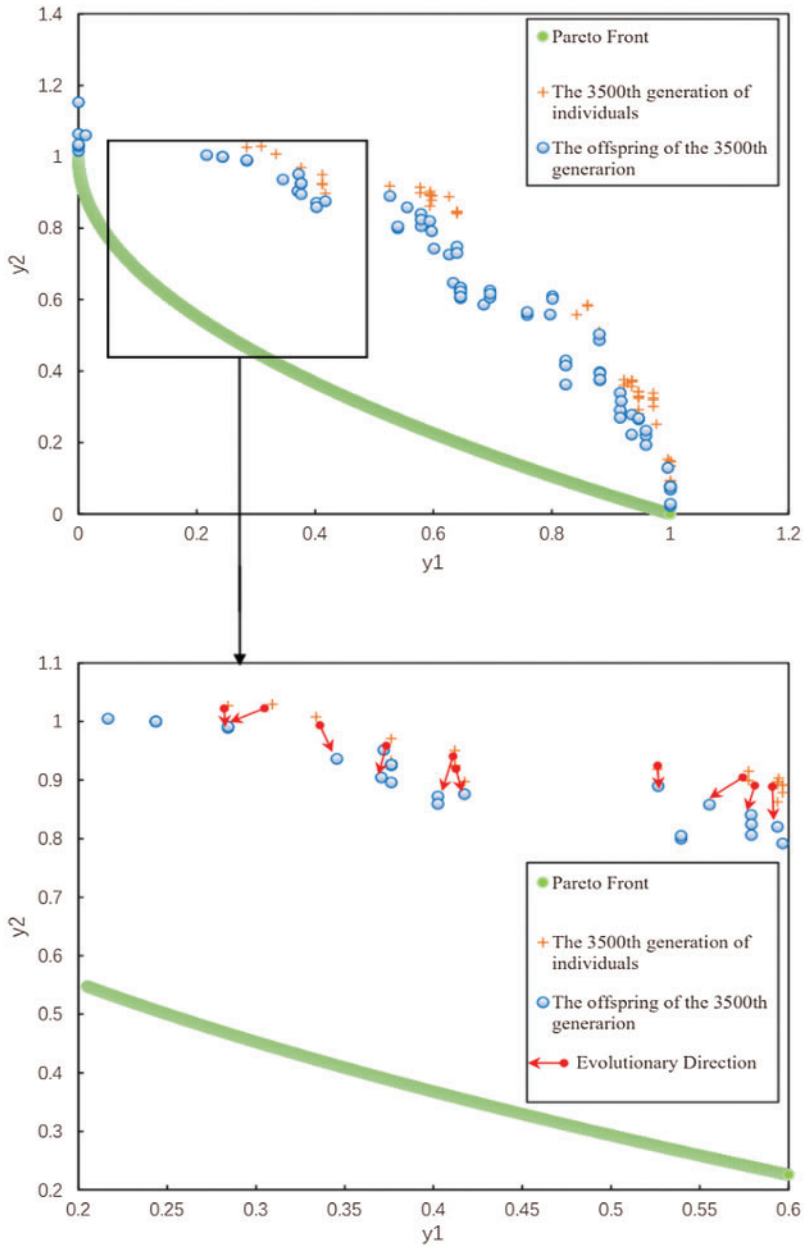


Figure 1: Procedure of SIACGDE and the relationship between parents and offspring individuals. The data include the 3500th generation of the algorithm running in ZDT1 and its next generation. The PF is the PF of ZDT1

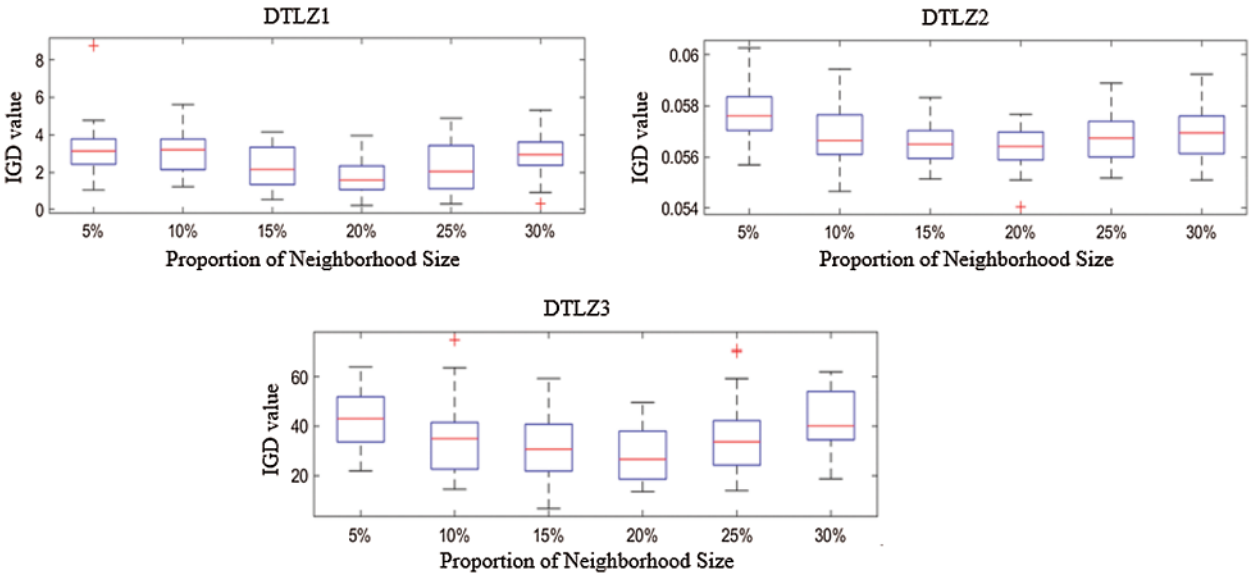
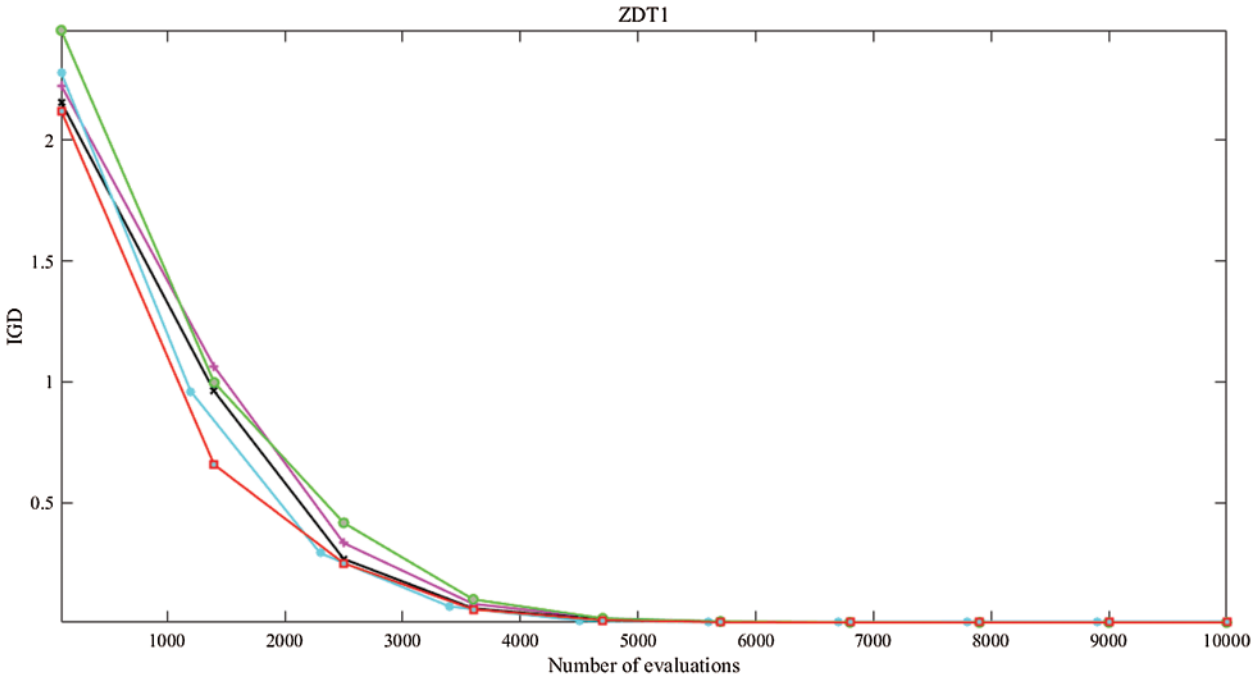


Figure 2: Boxplots of the IGD values over 30 independent runs. The simulations were performed on DTLZ1-DTLZ3 separately

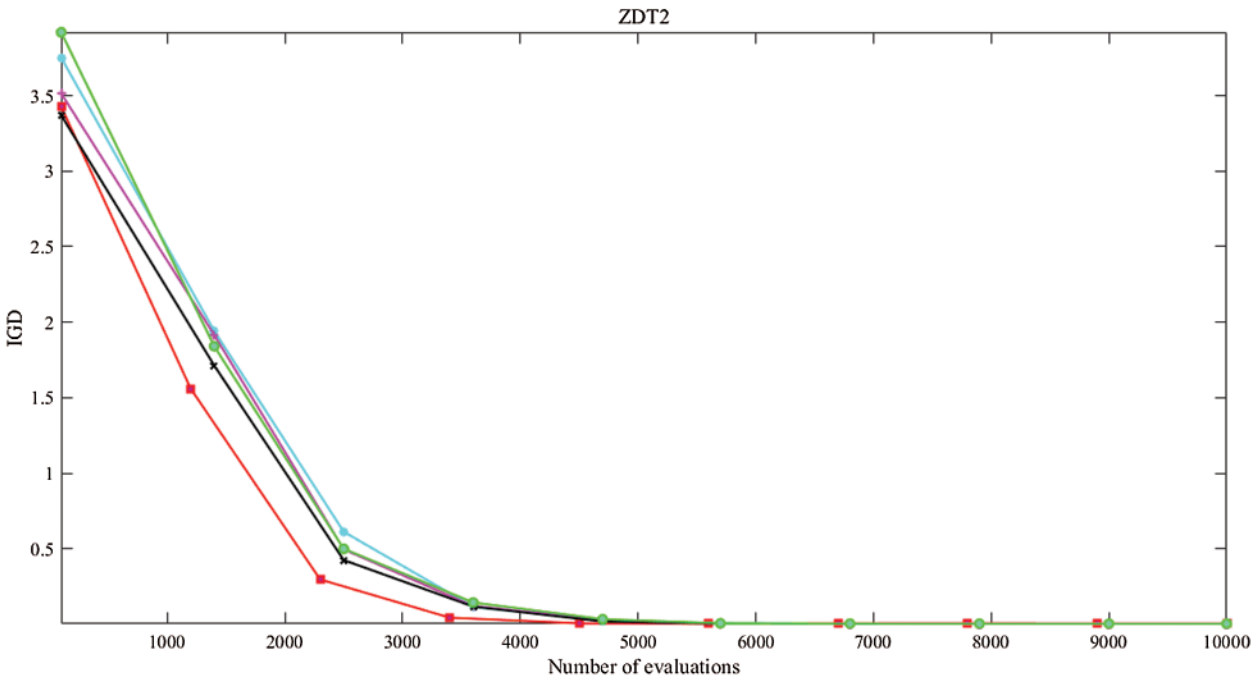
From the above experiments, we can see that the NCG mutation factor plays a key role in finding the optimal solution, and the size of the *Neighborhood* is the most important parameter in NCG. Therefore, we must find the best value of *Neighborhood* size to achieve the optimal performance of the algorithm. In the experimental, this value is determined by changing the size of *Neighborhood*, where the size refers to the percentage of the population size, with the other parameters remained unchanged. We can see from Fig. 2 that the optimal value of *Neighborhood* size in the three test functions, DTLZ1, DTLZ2, and DTLZ3, is 20%. Starting from a value of 20%, as the size becomes smaller, the IGD becomes larger and unstable, which means that the convergence and diversity of population are getting worse and worse. As the value of size increases towards 30%, greater error is observed. Therefore, the proportion of *Neighborhood* size is fixed at 20% and implemented into SIACGDE.

In the last subsection of the experiment, we tested the GA factor that has the greatest impact on the original ACGDE algorithm, the GA replaces the initial stage of mutation. The purpose of this experiment is to find how many generations in initial stage of GA will maximize the algorithm convergence of performance. Before the experiment, for the convenience of comparison, we named different GA proportions of SIACGDE: When the SIACGDE just used OBL in initialization, this model is SIACGDEI. When the proportion of GA in whole algorithm is 10%, this model is SIACGDEII. When the proportion of GA in whole algorithm is 15%, this model is SIACGDEIII. The SIACGDE algorithm, used in this paper, the proportion of GA in whole algorithm is 5%.

Fig. 3 illustrates the performance of the SIACGDE algorithms and the original ACGDE algorithm on ZDT1-ZDT3. It can be seen that in the three tests, only the first 5%N generation using GA achieved the best convergence. Therefore, it can be deduced that the convergence of the algorithm is affected by the proportion of GA.



(a)



(b)

Figure 3: (Continued)

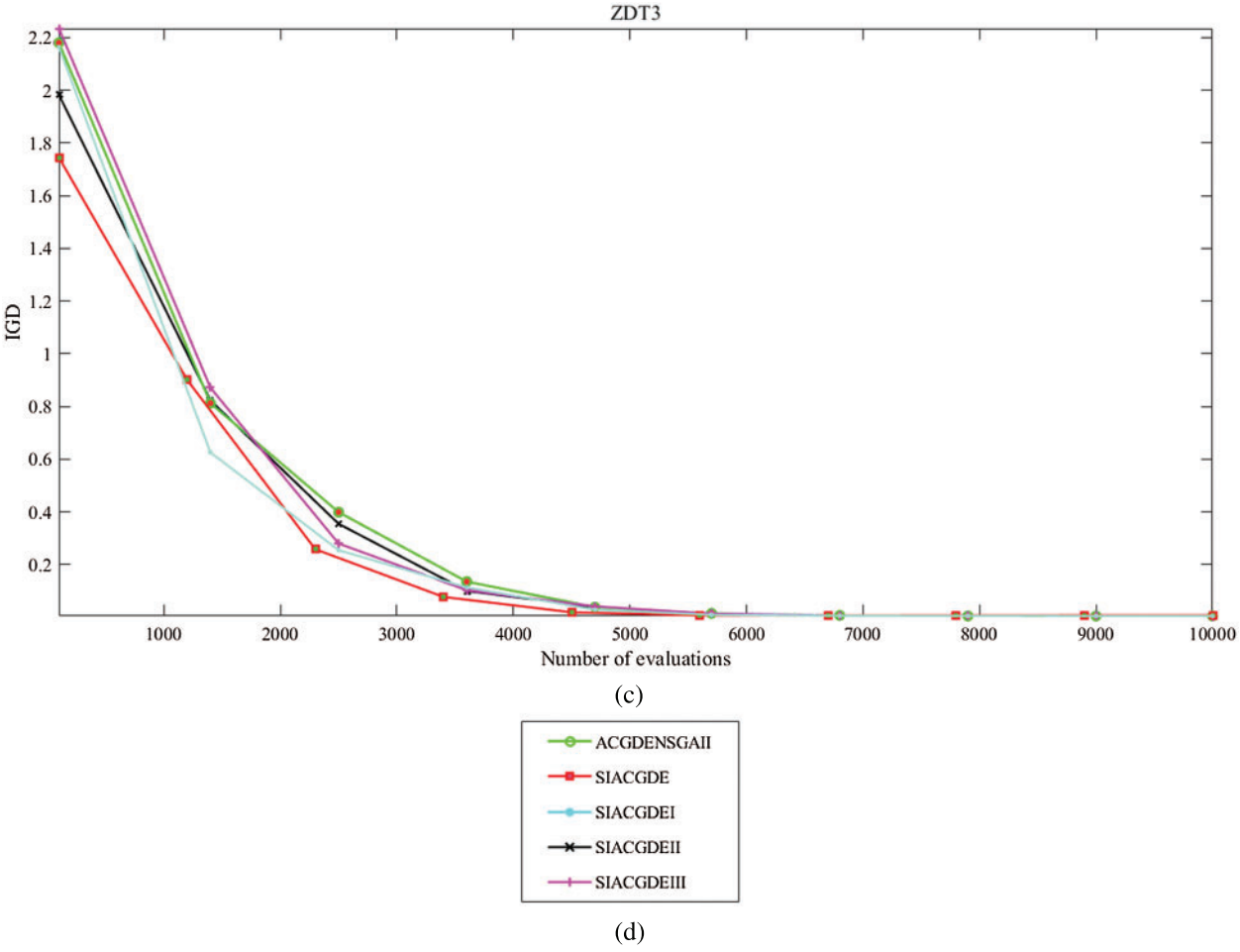


Figure 3: IGD values of four SIACGDE algorithms and ACGDE algorithm on ZDT1-ZDT2 (a) IGD of ZDT1 (b) IGD of ZDT2 (c) IGD of ZDT3 (d) Legend

5 Conclusions

This paper describes the improvement of the ACGDE algorithm by introducing the GA mutant operator in the initial stage, which corrects for the lack of search ability. SIACGDE specifically developed a new initialization method for two cross-generation mutations, NCG and PCG, so that the new alternative individuals are more suitable for the two mutations. Furthermore, the OBL mechanism is added to improve the efficiency of evolution. Experimental results demonstrate the significant improvements of two adjusted versions of ACGDE and that SIACGDE is obviously superior to the other six state-of-the-art algorithms. However, our proposed algorithm has its limitations. Firstly, when the real Pareto front is not continuous, the solving ability is reduced, which is also caused by the characteristics of cross-generation mutation. Secondly, the algorithm has no ability to deal with the problem of constraints.

In the future, the first plan is to find a method to deal with constraints matching the algorithm, so as to solve the real-world problems [50,51] and make SIACGDE more vitality. Then we plan to adopt a dynamic adaptive mechanism for the ratio of several mutant operators used in SIACGDE. We will also aim to develop a better strategy to apply in the early stage of the

algorithm so that it could be implemented into other many-objective frameworks to deal with higher dimensional problems. In addition, introducing more generations to participate in evolution will be considered. Finally, we will try to apply the concept of cross-generation evolution to other metaheuristic algorithms to examine its universe use.

Acknowledgement: The authors thank the Ocean University of China for their support in this work.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Storn, R. (1995). Differrential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report, 952.
2. Guo, S., Yang, C. (2014). Enhancing differential evolution utilizing eigenvector-based crossover operator. *IEEE Transactions on Evolutionary Computation, 19(1),* 31–49. DOI 10.1109/TEVC.2013.2297160.
3. Meng, Z., Chen, Y., Li, X., Lin, F. (2020). PaDE-NPC: Parameter adaptive differential evolution with novel parameter control for single-objective optimization. *IEEE Access, 8,* 139460–139478. DOI 10.1109/ACCESS.2020.3012885.
4. Ozer, A. B. (2010). CIDE: Chaotically initialized differential evolution. *Expert Systems with Applications, 37(6),* 4632–4641. DOI 10.1016/j.eswa.2009.12.045.
5. Li, G., Wang, G. G., Wang, S. (2021). Two-population coevolutionary algorithm with dynamic learning strategy for many-objective optimization. *Mathematics, 9(4),* 420. DOI 10.3390/math9040420.
6. Zhang, H., Wang, G. G., Dong, J., Gandomi, A. H. (2021). Improved NSGA-III with second-order difference random strategy for dynamic multi-objective optimization. *Processes, 9(6),* 911. DOI 10.3390/pr9060911.
7. Yi, J. H., Xing, L. N., Wang, G. G., Dong, J., Vasilakos, A. V. et al. (2020). Behavior of crossover operators in NSGA-III for large-scale optimization problems. *Information Sciences, 509(15),* 470–487. DOI 10.1016/j.ins.2018.10.005.
8. Zhang, Y., Wang, G. G., Li, K., Yeh, W. C., Jian, M. et al. (2020). Enhancing MOEA/D with information feedback models for large-scale many-objective optimization. *Information Sciences, 522(1),* 1–16. DOI 10.1016/j.ins.2020.02.066.
9. Gong, W., Cai, Z. (2013). Differential evolution with ranking-based mutation operators. *IEEE Transactions on Cybernetics, 43(6),* 2066–2081. DOI 10.1109/TCYB.2013.2239988.
10. Tang, L., Dong, Y., Liu, J. (2014). Differential evolution with an individual-dependent mechanism. *IEEE Transactions on Evolutionary Computation, 19(4),* 560–574. DOI 10.1109/TEVC.2014.2360890.
11. Gao, D., Wang, G. G. (2020). Solving fuzzy job-shop scheduling problem using DE algorithm improved by a selection mechanism. *IEEE Transactions on Fuzzy Systems, 28(12),* 3265–3275. DOI 10.1109/TFUZZ.2020.3003506.
12. Wang, G. G., Cai, X., Cui, Z., Min, G., Chen, J. (2020). High performance computing for cyber physical social systems by using evolutionary multi-objective optimization algorithm. *IEEE Transactions on Emerging Topics in Computing, 8(1),* 20–30. DOI 10.1109/TETC.2017.2703784.
13. Huang, Y., Li, W., Tian, F., Meng, X. (2020). A fitness landscape ruggedness multiobjective differential evolution algorithm with a reinforcement learning strategy. *Applied Soft Computing, 96(1),* 106693. DOI 10.1016/j.asoc.2020.106693.
14. Carvalho, J. P. G., Carvalho, É. C., Vargas, D. E., Hallak, P. H., Lima, B. S. et al. (2021). Multi-objective optimum design of truss structures using differential evolution algorithms. *Computers Structures, 252(10),* 106544. DOI 10.1016/j.compstruc.2021.106544.

15. Cheng, J., Yen, G. G., Zhang, G. (2016). A grid-based adaptive multi-objective differential evolution algorithm. *Information Sciences, 367(1),* 890–908. DOI 10.1016/j.ins.2016.07.009.
16. Zhang, Y., Gong, D., Gao, X., Tian, T., Sun, X. (2020). Binary differential evolution with self-learning for multi-objective feature selection. *Information Sciences, 507(2),* 67–85. DOI 10.1016/j.ins.2019.08.040.
17. Pan, Z., Fang, S., Wang, H. (2020). LightGBM technique and differential evolution algorithm-based multi-objective optimization design of DS-APMM. *IEEE Transactions on Energy Conversion, 10(1),* 1109. DOI 10.1109/TEC.2020.3009480.
18. Deng, W., Liu, H., Xu, J., Zhao, H., Song, Y. (2020). An improved quantum-inspired differential evolution algorithm for deep belief network. *IEEE Transactions on Instrumentation Measurement, 69(10),* 7319–7327. DOI 10.1109/TIM.2020.2983233.
19. Mininno, E., Neri, F., Cupertino, F., Naso, D. (2010). Compact differential evolution. *IEEE Transactions on Evolutionary Computation, 15(1),* 32–54. DOI 10.1109/TEVC.2010.2058120.
20. Liu, Y., Chen, Y., Yang, G. (2018). Developing multiobjective equilibrium optimization method for sustainable uncertain supply chain planning problems. *IEEE Transactions on Fuzzy Systems, 27(5),* 1037–1051. DOI 10.1109/TFUZZ.2018.2851508.
21. Wang, Y., Liu, Z. Z., Li, J., Li, H. X., Wang, J. (2018). On the selection of solutions for mutation in differential evolution. *Frontiers of Computer Science, 12(2),* 297–315. DOI 10.1007/s11704-016-5353-5.
22. Sun, G., Xu, G., Jiang, N. (2020). A simple differential evolution with time-varying strategy for continuous optimization. *Soft Computing, 24(4),* 2727–2747. DOI 10.1007/s00500-019-04159-0.
23. Cardenas-Montes, M. (2018). Weibull-based scaled-differences schema for differential evolution. *Swarm Evolutionary Computation, 38(4),* 79–93. DOI 10.1016/j.swevo.2017.06.004.
24. Mesejo, P., Ugolotti, R., Cunto, F. D., Giacobini, M., Cagnoni, S. (2013). Automatic hippocampus localization in histological images using differential evolution-based deformable models. *Pattern Recognition Letters, 34(3),* 299–307. DOI 10.1016/j.patrec.2012.10.012.
25. Awad, N. H., Ali, M. Z., Suganthan, P. N., Reynolds, R. G. (2017). CADE: A hybridization of cultural algorithm and differential evolution for numerical optimization. *Information Sciences, 378(6),* 215–241. DOI 10.1016/j.ins.2016.10.039.
26. Kukkonen, S., Lampinen, J. (2005). GDE3: The third evolution step of generalized differential evolution. *2005 IEEE Congress on Evolutionary Computation*. vol. 3, pp. 443–450. Edinburgh, UK.
27. Das, S., Abraham, A., Chakraborty, U. K., Konar, A. (2009). Differential evolution using a neighborhood-based mutation operator. *IEEE Transactions on Evolutionary Computation, 13(3),* 526–553. DOI 10.1109/TEVC.2008.2009457.
28. Wang, Y., Cai, Z., Zhang, Q. (2011). Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation, 15(1),* 55–66. DOI 10.1109/TEVC.2010.2087271.
29. Storn, R. (1996). On the usage of differential evolution for function optimization. *Proceedings of North American Fuzzy Information Processing*, pp. 519–523. Berkeley, USA.
30. Pant, M., Zaheer, H., Garcia-Hernandez, L., Abraham, A. (2020). Differential evolution: A review of more than two decades of research. *Engineering Applications of Artificial Intelligence, 90(1),* 103479. DOI 10.1016/j.engappai.2020.103479.
31. Qiu, X., Xu, J., Tan, K., Abbass, H. A. (2015). Adaptive cross-generation differential evolution operators for multiobjective optimization. *IEEE Transactions on Evolutionary Computation, 20(2),* 232–244. DOI 10.1109/TEVC.2015.2433672.
32. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *International Conference on Parallel Problem Solving from Nature*, pp. 849–858. Springer, Berlin, Heidelberg.
33. Tizhoosh, H. R. (2005). Opposition-based learning: A new scheme for machine intelligence. *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, pp. 695–701. Vienna, Austria.

34. Rahnamayan, S., Tizhoosh, H. R., Salama, M. M. (2008). Opposition-based differential evolution. *IEEE Transactions on Evolutionary Computation, 12(1),* 64–79. DOI 10.1109/TEVC.2007.894200.
35. Jiang, S., Ong, Y. S., Zhang, J., Feng, L. (2014). Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE Transactions on Cybernetics, 44(12),* 2391–2404. DOI 10.1109/TCYB.2014.2307319.
36. Wang, Y., Wu, L., Yuan, X. (2010). Multi-objective self-adaptive differential evolution with elitist archive and crowding entropy-based diversity measure. *Soft Computing, 14(3),* 193–209. DOI 10.1007/s00500-008-0394-9.
37. Tian, Y., Cheng, R., Zhang, X., Li, M., Jin, Y. (2019). Diversity assessment of multi-objective evolutionary algorithms: Performance metric and benchmark problems [Research Frontier]. *IEEE Computational Intelligence Magazine, 14(3),* 61–74. DOI 10.1109/MCI.2019.2919398.
38. Robič, T., Filipič, B. (2005). Differential evolution for multiobjective optimization. *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 520–533. Springer, Berlin, Heidelberg.
39. Mahdavi, S., Rahnamayan, S., Deb, K. (2018). Opposition based learning: A literature review. *Swarm Evolutionary Computation, 39(8),* 1–23. DOI 10.1016/j.swevo.2017.09.010.
40. Zitzler, E., Deb, K., Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation, 8(2),* 173–195. DOI 10.1162/106365600568202.
41. Jain, H., Deb, K. (2013). An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation, 18(4),* 602–622. DOI 10.1109/TEVC.2013.2281534.
42. Vlennet, R., Fonteix, C., Marc, I. (1996). Multicriteria optimization using a genetic algorithm for determining a Pareto set. *International Journal of Systems Science, 27(2),* 255–260. DOI 10.1080/00207729608929211.
43. Huband, S., Hingston, P., Barone, L., While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation, 10(5),* 477–506. DOI 10.1109/TEVC.2005.861417.
44. Zitzler, E., Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation, 3(4),* 257–271. DOI 10.1109/4235.797969.
45. Li, H., Zhang, Q. (2008). Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation, 13(2),* 284–302. DOI 10.1109/TEVC.2008.925798.
46. Zille, H. (2019). *Large-scale multi-objective optimisation: New approaches and a classification of the state-of-the-art (PhD Dissertation)*. Germany: Otto-vonGuericke-Universitaet Magdeburg
47. Zhang, X., Tian, Y., Cheng, R., Jin, Y. (2016). A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. *IEEE Transactions on Evolutionary Computation, 22(1),* 97–112. DOI 10.1109/TEVC.2016.2600642.
48. Li, M., Yang, S., Liu, X. (2013). Shift-based density estimation for Pareto-based algorithms in many-objective optimization. *IEEE Transactions on Evolutionary Computation, 18(3),* 348–365. DOI 10.1109/TEVC.2013.2262178.
49. Zitzler, E., Laumanns, M., Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. TIK-Report, 103.
50. Kumar, A., Wu, G., Ali, M. Z., Mallipeddi, R., Suganthan, P. N. et al. (2020). A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm Evolutionary Computation, 56,* 100693. DOI 10.1016/j.swevo.2020.100693.
51. Chamorro, H. R., Riano, I., Gerndt, R., Zelinka, I., Gonzalez-sLongatt, F. et al. (2019). Synthetic inertia control based on fuzzy adaptive differential evolution. *International Journal of Electrical Power Energy Systems, 105(1),* 803–813. DOI 10.1016/j.ijepes.2018.09.009.

Appendix A

Table A1: Mean and standard deviation of the values spread

Problem	M	GDE3	MOEADDE	NSGAII	LCSA	LMEA	SPEA2SDE	ACGDENSGAII	SIACGDE
VNT1	3	3.8583e − 1	6.9251e − 1	4.8022e − 1	6.4162e − 1	4.5108e − 1	4.5857e − 1	5.6532e − 1	**1.2825e − 1**
		(3.05e − 2) −	(2.73e − 2) −	(4.93e − 2) −	(6.97e − 2) −	(4.25e − 2) −	(3.65e − 2) −	(5.97e − 2) −	**(8.69e − 3)**
VNT2	3	3.4594e − 1	1.2971e + 0	5.4477e − 1	9.7740e − 1	9.6290e − 1	8.9665e − 1	8.2969e − 1	**1.9878e − 1**
		(3.44e − 2) −	(1.77e − 2) −	(5.32e − 2) −	(7.44e − 2) −	(6.95e − 2) −	(5.92e − 2) −	(8.36e − 2) −	**(2.85e − 2)**
VNT3	3	6.5924e − 1	1.1593e + 0	7.6379e − 1	1.2834e + 0	8.8243e − 1	1.1198e + 0	1.0612e + 0	**1.7299e − 1**
		(2.59e − 2) −	(1.06e − 1) −	(3.42e − 2) −	(1.10e − 1) −	(3.08e − 2) −	(1.10e − 1) −	(8.30e − 2) −	**(2.81e − 2)**
VNT4	3	4.6556e − 1	9.8453e − 1	6.0677e − 1	6.9888e − 1	7.1814e − 1	6.9797e − 1	8.5081e − 1	**4.3655e − 1**
		(4.45e − 2) −	(1.79e − 2) −	(4.61e − 2) −	(7.46e − 2) −	(3.19e − 2) −	(1.92e − 2) −	(6.69e − 2) −	**(1.62e − 2)**
WFG1	3	9.5443e − 1	8.6690e − 1	7.0383e − 1	8.1751e − 1	1.3197e + 0	**6.2643e − 1**	9.8055e − 1	9.7618e − 1
		(2.12e − 2) =	(5.53e − 2) +	(4.52e − 2) +	(7.00e − 2) +	(2.08e − 1) −	**(5.00e − 2) +**	(9.02e − 2) =	(1.68e − 1)
WFG2	3	4.9531e − 1	9.3141e − 1	5.2113e − 1	4.5339e − 1	7.7922e − 1	6.1243e − 1	5.5433e − 1	**1.8490e − 1**
		(3.78e − 2) −	(8.56e − 2) −	(5.85e − 2) −	(5.40e − 2) −	(1.06e − 1) −	(4.65e − 2) −	(4.68e − 2) −	**(3.81e − 2)**
WFG3	3	4.9286e − 1	6.7378e − 1	5.8718e − 1	6.9372e − 1	3.9573e − 1	2.7274e − 1	5.9517e − 1	**1.4476e − 1**
		(3.95e − 2) −	(6.09e − 2) −	(5.26e − 2) −	(5.80e − 2) −	(3.99e − 2) −	(3.17e − 2) −	(5.07e − 2) −	**(1.48e − 2)**
WFG4	3	4.2163e − 1	8.2134e − 1	4.4712e − 1	3.3669e − 1	3.4790e − 1	3.7447e − 1	4.3578e − 1	**1.1860e − 1**
		(4.03e − 2) −	(1.10e − 1) −	(3.00e − 2) −	(2.28e − 2) −	(2.97e − 2) −	(2.52e − 2) −	(3.80e − 2) −	**(1.48e − 2)**
WFG5	3	3.9367e − 1	7.8173e − 1	5.0027e − 1	3.8441e − 1	2.7601e − 1	3.7015e − 1	4.6857e − 1	**1.0866e − 1**
		(3.80e − 2) −	(4.60e − 2) −	(4.69e − 2) −	(2.42e − 2) −	(2.25e − 2) −	(2.91e − 2) −	(5.00e − 2) −	**(1.10e − 2)**
WFG6	3	3.9921e − 1	8.8743e − 1	4.9709e − 1	4.2747e − 1	2.8059e − 1	3.8476e − 1	4.9730e − 1	**1.3069e − 1**
		(3.39e − 2) −	(8.22e − 2) −	(3.92e − 2) −	(6.09e − 2) −	(3.23e − 2) −	(3.64e − 2) −	(4.95e − 2) −	**(1.42e − 2)**
WFG7	3	3.8759e − 1	6.7532e − 1	5.7243e − 1	3.8348e − 1	2.3517e − 1	3.7893e − 1	5.2843e − 1	**1.1005e − 1**
		(2.76e − 2) −	(8.11e − 2) −	(5.72e − 2) −	(4.50e − 2) −	(2.86e − 2) −	(2.97e − 2) −	(4.68e − 2) −	**(1.53e − 2)**
WFG8	3	4.1099e − 1	7.4598e − 1	5.3368e − 1	4.6012e − 1	3.2062e − 1	3.6914e − 1	4.4774e − 1	**1.3011e − 1**
		(3.03e − 2) −	(9.00e − 2) −	(4.13e − 2) −	(6.34e − 2) −	(3.31e − 2) −	(2.74e − 2) −	(3.87e − 2) −	**(1.37e − 2)**
WFG9	3	3.9499e − 1	7.8124e − 1	4.8120e − 1	3.7504e − 1	2.3126e − 1	3.5486e − 1	4.6679e − 1	**1.2357e − 1**
		(2.91e − 2) −	(6.41e − 2) −	(3.66e − 2) −	(3.91e − 2) −	(2.09e − 2) −	(2.55e − 2) −	(4.11e − 2) −	**(1.32e − 2)**
DTLZ1	3	6.1843e − 1	1.1651e + 0	6.1903e − 1	8.1300e − 1	2.7682e − 1	4.4943e − 1	5.4460e − 1	**2.3506e − 1**
		(1.11e − 1) −	(4.06e − 1) −	(1.53e − 1) −	(7.77e − 2) −	(7.55e − 2) −	(3.69e − 1) −	(9.53e − 2) −	**(4.16e − 2)**
DTLZ2	3	3.6511e − 1	7.2693e − 1	5.2172e − 1	3.2673e − 1	1.8948e − 1	3.2863e − 1	4.1585e − 1	**9.6738e − 2**
		(4.00e − 2) −	(4.66e − 2) −	(4.00e − 2) −	(4.67e − 2) −	(2.41e − 2) −	(2.53e − 2) −	(5.72e − 2) −	**(8.14e − 3)**
DTLZ3	3	9.9070e − 1	1.2035e + 0	9.1350e − 1	9.6831e − 1	**8.1804e − 1**	9.5081e − 1	1.0395e + 0	8.7245e − 1
		(2.67e − 1) =	(2.48e − 1) −	(1.14e − 1) =	(1.02e − 1) =	**(1.13e − 1) =**	(9.91e − 2) =	(1.57e − 1) −	(2.58e − 1)
DTLZ4	3	5.6974e − 1	1.0206e + 0	5.1083e − 1	4.6431e − 1	1.4167e + 0	3.9712e − 1	4.2756e − 1	**9.6301e − 2**
		(1.19e − 1) −	(8.45e − 2) −	(8.00e − 2) −	(1.13e − 1) −	(1.84e − 1) −	(1.69e − 1) −	(4.05e − 2) −	**(1.10e − 2)**
DTLZ5	3	3.0078e − 1	1.1410e + 0	5.0231e − 1	7.8707e − 1	3.5182e − 1	5.9300e − 1	5.2722e − 1	**1.9440e − 1**
		(3.21e − 2) −	(1.49e − 1) −	(5.54e − 2) −	(1.34e − 1) −	(3.87e − 2) −	(2.99e − 2) −	(5.90e − 2) −	**(2.12e − 2)**
DTLZ6	3	6.4761e − 1	1.3944e + 0	7.4365e − 1	1.5437e + 0	7.4778e − 1	6.0124e − 1	1.0561e + 0	**1.1689e − 1**
		(1.93e − 1) −	(9.18e − 2) −	(8.41e − 2) −	(8.84e − 2) −	(1.02e − 1) −	(5.57e − 2) −	(8.66e − 2) −	**(1.47e − 2)**
ZDT1	2	7.8192e − 1	1.0383e + 0	3.6500e − 1	3.3249e − 1	9.0033e − 1	3.7093e − 1	1.2118e + 0	**1.4548e − 1**
		(3.10e − 2) −	(6.89e − 2) −	(3.83e − 2) −	(4.52e − 2) −	(5.48e − 2) −	(2.35e − 2) −	(1.23e − 1) −	**(1.78e − 2)**
ZDT2	2	9.3303e − 1	1.0110e + 0	4.2942e − 1	5.0025e − 1	9.8143e − 1	4.4852e − 1	1.1787e + 0	**1.2535e − 1**
		(4.49e − 2) −	(1.98e − 2) −	(1.31e − 1) −	(3.44e − 1) −	(3.77e − 2) −	(6.64e − 2) −	(1.01e − 1) −	**(1.30e − 2)**
ZDT3	2	8.8362e − 1	1.0833e + 0	4.2857e − 1	8.1090e − 1	9.0184e − 1	5.4774e − 1	1.2219e + 0	**2.1335e − 1**
		(4.10e − 2) −	(5.39e − 2) −	(7.50e − 2) −	(1.01e − 1) −	(7.42e − 2) −	(3.87e − 2) −	(9.31e − 2) −	**(2.26e − 2)**
ZDT4	2	1.0061e + 0	1.7449e + 0	**8.8766e − 1**	9.4711e − 1	9.5781e − 1	9.0914e − 1	9.3729e − 1	8.9735e − 1
		(4.68e − 2) −	(2.14e − 1) −	**(1.00e − 1) =**	(7.36e − 2) −	(1.41e − 1) −	(2.41e − 1) =	(8.61e − 2) −	(7.22e − 2)
ZDT6	2	1.3583e + 0	1.1561e + 0	6.8469e − 1	5.5590e − 1	1.0641e + 0	6.9695e − 1	1.3267e + 0	**1.2395e − 1**
		(3.67e − 1) −	(6.25e − 1) −	(1.56e − 1) −	(4.27e − 1) −	(1.39e − 1) −	(1.12e − 1) −	(1.02e − 1) −	**(1.05e − 2)**
+/-/=		0/22/2	1/23/0	1/21/2	1/22/1	0/23/1	1/21/2	0/23/1	

The bold font represents the best performance on this problem. Where the symbols '+', '−' and '=' indicate the results of compared algorithm is significantly better, significantly worse, and

statistically similar to that obtained by SIACGDED at a 0.05 level by the Wilcoxon's rank sum test, respectively.

Table A2: Mean and standard deviation of the values CPF

Problem	M	GDE3	MOEADDE	NSGAII	LCSA	LMEA	SPEA2SDE	ACGDENSGAII	SIACGDE
VNT1	3	4.1061e − 1	3.4987e − 1	3.6818e − 1	3.9298e − 1	5.2968e − 1	4.4238e − 1	3.4988e − 1	**5.7357e − 1**
		(3.03e − 2) −	(2.02e − 2) −	(3.41e − 2) −	(4.91e − 2) −	(3.00e − 2) −	(2.85e − 2) −	(3.93e − 2) −	**(3.62e − 2)**
VNT2	3	1.4062e − 1	1.2871e − 1	1.2878e − 1	1.3418e − 1	1.6965e − 1	1.4616e − 1	1.0872e − 1	**2.1192e − 1**
		(1.99e − 2) −	(1.27e − 2) −	(1.85e − 2) −	(2.91e − 2) −	(1.52e − 2) −	(1.72e − 2) −	(1.74e − 2) −	**(1.78e − 2)**
VNT3	3	**6.0652e − 1**	1.4576e − 1	4.7590e − 1	1.3851e − 1	4.4576e − 1	2.2039e − 1	3.3171e − 1	4.8823e − 1
		(3.48e − 2) +	(1.12e − 2) −	(3.59e − 2) =	(3.13e − 2) −	(2.69e − 2) −	(4.02e − 2) −	(4.04e − 2) −	(3.72e − 2)
VNT4	3	4.4409e − 1	3.1403e − 1	3.8689e − 1	3.1331e − 1	4.7496e − 1	4.8042e − 1	3.1891e − 1	**6.4168e − 1**
		(3.32e − 2) −	(4.03e − 2) −	(3.03e − 2) −	(4.15e − 2) −	(4.26e − 2) −	(3.42e − 2) −	(3.39e − 2) −	**(4.74e − 2)**
WFG1	3	1.1993e − 2	6.0675e − 2	1.4661e − 1	5.5414e − 2	8.1154e − 3	**1.8281e − 1**	1.3821e − 2	2.0609e − 2
		(3.24e − 3) −	(2.11e − 2) +	(2.49e − 2) +	(1.39e − 2) +	(6.33e − 3) −	**(2.92e − 2) +**	(4.54e − 3) −	(8.75e − 3)
WFG2	3	4.2776e − 1	1.9787e − 1	4.2725e − 1	6.4917e − 1	2.9965e − 1	4.1425e − 1	4.3531e − 1	**7.2963e − 1**
		(3.56e − 2) −	(3.14e − 2) −	(4.15e − 2) −	(6.34e − 2) −	(5.09e − 2) −	(3.14e − 2) −	(4.03e − 2) −	**(4.19e − 2)**
WFG3	3	5.1338e − 1	4.8023e − 1	5.3908e − 1	5.1037e − 1	**9.1302e − 1**	4.7693e − 1	5.3531e − 1	5.3158e − 1
		(3.53e − 2) −	(3.49e − 2) −	(4.32e − 2) =	(4.33e − 2) −	**(3.51e − 2) +**	(2.89e − 2) −	(4.42e − 2) =	(2.54e − 2)
WFG4	3	3.8559e − 1	3.7957e − 1	3.7184e − 1	**6.6197e − 1**	4.9452e − 1	4.1885e − 1	3.9661e − 1	6.4911e − 1
		(3.04e − 2) −	(4.75e − 2) −	(2.94e − 2) −	**(1.78e − 2) +**	(5.01e − 2) −	(2.24e − 2) −	(3.18e − 2) −	(2.84e − 2)
WFG5	3	3.9421e − 1	3.5281e − 1	3.2584e − 1	6.1957e − 1	5.3951e − 1	4.1632e − 1	3.6193e − 1	**6.3700e − 1**
		(3.16e − 2) −	(1.31e − 2) −	(3.29e − 2) −	(2.11e − 2) −	(3.22e − 2) −	(2.40e − 2) −	(3.72e − 2) −	**(2.59e − 2)**
WFG6	3	3.9129e − 1	3.1384e − 1	3.2998e − 1	5.6264e − 1	5.2581e − 1	4.0052e − 1	3.4274e − 1	**5.9954e − 1**
		(3.82e − 2) −	(3.21e − 2) −	(2.92e − 2) −	(3.97e − 2) −	(3.24e − 2) −	(2.27e − 2) −	(4.09e − 2) −	**(2.59e − 2)**
WFG7	3	3.9701e − 1	3.8370e − 1	3.1329e − 1	6.1965e − 1	**6.5554e − 1**	4.1572e − 1	3.3888e − 1	6.2638e − 1
		(3.16e − 2) −	(3.61e − 2) −	(3.94e − 2) −	(3.55e − 2) =	**(2.37e − 2) +**	(1.98e − 2) −	(3.66e − 2) −	(2.83e − 2)
WFG8	3	3.6368e − 1	3.4912e − 1	3.1269e − 1	5.0110e − 1	4.4169e − 1	4.1418e − 1	3.4876e − 1	**5.8886e − 1**
		(2.90e − 2) −	(2.34e − 2) −	(3.13e − 2) −	(3.33e − 2) −	(3.83e − 2) −	(2.46e − 2) −	(3.15e − 2) −	**(2.19e − 2)**
WFG9	3	3.7370e − 1	3.4841e − 1	3.1711e − 1	5.6093e − 1	**6.1806e − 1**	4.1388e − 1	3.3280e − 1	5.9597e − 1
		(3.01e − 2) −	(2.38e − 2) −	(3.23e − 2) −	(3.28e − 2) −	**(3.10e − 2) +**	(3.17e − 2) −	(3.37e − 2) −	(2.45e − 2)
DTLZ1	3	1.7844e − 2	1.7073e − 1	1.3992e − 1	4.5853e − 2	**4.8978e − 1**	2.7596e − 1	8.5965e − 2	8.6788e − 2
		(2.11e − 2) −	(1.93e − 1) =	(7.72e − 2) +	(2.54e − 2) −	**(7.32e − 2) +**	(1.74e − 1) +	(7.37e − 2) =	(4.81e − 2)
DTLZ2	3	4.1335e − 1	4.9404e − 1	3.2196e − 1	6.5682e − 1	5.4981e − 1	4.3623e − 1	3.7830e − 1	**6.9676e − 1**
		(3.94e − 2) −	(3.14e − 2) −	(3.19e − 2) −	(2.10e − 2) −	(3.55e − 2) −	(1.82e − 2) −	(2.89e − 2) −	**(2.48e − 2)**
DTLZ3	3	2.4167e − 1	1.5296e − 1	1.3322e − 1	1.1064e − 1	**2.9395e − 1**	1.1361e − 1	2.3046e − 1	2.8561e − 1
		(6.42e − 2) =	(1.00e − 1) −	(6.27e − 2) −	(6.36e − 2) −	**(4.66e − 2) =**	(5.61e − 2) −	(7.72e − 2) −	(1.04e − 1)
DTLZ4	3	2.6485e − 1	1.6441e − 1	3.2718e − 1	6.3973e − 1	5.8842e − 2	3.7805e − 1	3.6914e − 1	**7.0162e − 1**
		(8.55e − 2) −	(1.24e − 1) −	(6.79e − 2) −	(3.35e − 2) −	(2.84e − 2) −	(1.43e − 1) −	(3.37e − 2) −	**(2.98e − 2)**
DTLZ5	3	9.2035e − 1	4.3657e − 1	7.8871e − 1	6.3602e − 1	8.7198e − 1	**9.8964e − 1**	7.6854e − 1	9.4030e − 1
		(1.89e − 2) −	(7.07e − 2) −	(3.89e − 2) −	(6.51e − 2) −	(2.66e − 2) −	**(1.48e − 3) +**	(4.37e − 2) −	(1.58e − 2)
DTLZ6	3	7.8487e − 1	2.1561e − 1	6.5308e − 1	2.3081e − 1	7.1408e − 1	**9.8970e − 1**	4.5902e − 1	9.2944e − 1
		(9.86e − 2) −	(5.15e − 2) −	(4.74e − 2) −	(4.83e − 2) −	(8.14e − 2) −	**(1.71e − 3) +**	(5.54e − 2) −	(9.98e − 3)
ZDT1	2	8.5357e − 2	1.8170e − 1	6.8342e − 1	7.9718e − 1	0.0000e + 0	7.0805e − 1	3.2313e − 1	**8.8143e − 1**
		(5.01e − 2) −	(8.47e − 2) −	(2.54e − 2) −	(4.02e − 2) −	(0.00e + 0) −	(1.78e − 2) −	(5.48e − 2) −	**(1.06e − 2)**
ZDT2	2	1.0558e − 1	9.4901e − 3	6.3183e − 1	6.2579e − 1	4.5989e − 2	6.3722e − 1	3.3576e − 1	**8.8377e − 1**
		(2.33e − 2) −	(2.20e − 2) −	(1.20e − 1) −	(2.53e − 1) −	(2.17e − 2) −	(4.41e − 2) −	(4.42e − 2) −	**(8.58e − 3)**
ZDT3	2	9.8039e − 4	4.9689e − 3	6.5800e − 1	4.4583e − 1	4.0421e − 2	6.4515e − 1	3.2265e − 1	**8.3967e − 1**
		(5.37e − 3) −	(8.14e − 3) −	(4.05e − 2) −	(8.72e − 2) −	(5.36e − 2) −	(2.95e − 2) −	(3.87e − 2) −	**(1.91e − 2)**
ZDT4	2	5.0000e − 2	0.0000e + 0	2.5385e − 1	0.0000e + 0	2.4095e − 1	**3.0455e − 1**	2.7614e − 2	3.4883e − 2
		(1.53e − 1) =	(0.00e + 0) −	(8.75e − 2) +	(0.00e + 0) −	(1.22e − 1) +	**(1.35e − 1) +**	(8.47e − 2) =	(7.86e − 2)
ZDT6	2	5.5017e − 1	6.2013e − 1	5.0975e − 1	7.5684e − 1	2.3176e − 1	5.0119e − 1	2.8401e − 1	**8.8768e − 1**
		(3.29e − 1) −	(3.42e − 1) −	(5.63e − 2) −	(9.69e − 2) −	(5.89e − 2) −	(6.09e − 2) −	(4.19e − 2) −	**(9.28e − 3)**
+/-/=		1/21/2	1/22/1	3/19/2	2/21/1	5/18/1	5/19/0	0/21/3	

Table A3: Mean and standard deviation of the runtime

Problem	M	GDE3	MOEADDE	NSGAII	LCSA	LMEA	SPEA2SDE	ACGDENSGAII	SIACGDE
VNT1	3	4.3520e − 1	1.4927e + 0	1.9050e − 1	5.4482e − 1	8.0590e − 1	4.5761e + 0	1.9391e + 0	1.9545e + 0
		(1.28e − 2)	(2.11e − 2)	(9.64e − 3)	(2.09e − 2)	(1.75e − 2)	(4.27e − 2)	(2.14e − 2)	(1.39e − 2)
VNT2	3	2.7341e − 1	1.4914e + 0	1.8379e − 1	5.5694e − 1	7.4865e − 1	3.3687e + 0	1.9318e + 0	1.9358e + 0
		(5.38e − 3)	(1.96e − 2)	(3.15e − 3)	(2.14e − 2)	(7.66e − 3)	(9.50e − 2)	(8.72e − 3)	(8.67e − 3)
VNT3	3	2.7389e − 1	1.5005e + 0	1.8355e − 1	5.5637e − 1	6.8591e − 1	2.3958e + 0	1.9467e + 0	1.9579e + 0
		(4.49e − 3)	(2.09e − 2)	(4.85e − 3)	(2.33e − 2)	(5.41e − 3)	(1.96e − 2)	(1.52e − 2)	(3.16e − 3)
VNT4	3	3.4775e − 1	1.5355e + 0	1.9118e − 1	5.6945e − 1	8.1982e − 1	4.4627e + 0	1.9897e + 0	2.1074e + 0
		(9.33e − 3)	(1.58e − 2)	(3.73e − 3)	(2.36e − 2)	(1.53e − 2)	(1.78e − 1)	(2.42e − 2)	(2.35e − 2)
Rank		2	5	1	3	4	8	6	7