Research Article

A Hybrid Mutation Chemical Reaction Optimization Algorithm for Global Numerical Optimization

Ransikarn Ngambusabongsopa,¹ Zhiyong Li,¹ and Esraa Eldesouky^{1,2}

¹College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China

²Faculty of Computers and Informatics, Suez Canal University, Ismailia 41522, Egypt

Correspondence should be addressed to Zhiyong Li; zhiyong.li@hnu.edu.cn

Received 10 August 2015; Accepted 29 September 2015

Academic Editor: Julien Bruchon

Copyright © 2015 Ransikarn Ngambusabongsopa et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper proposes a hybrid metaheuristic approach that improves global numerical optimization by increasing optimal quality and accelerating convergence. This algorithm involves a recently developed process for chemical reaction optimization and two adjustment operators (turning and mutation operators). Three types of mutation operators (uniform, nonuniform, and polynomial) were combined with chemical reaction optimization and turning operator to find the most appropriate framework. The best solution among these three options was selected to be a hybrid mutation chemical reaction optimization algorithm for global numerical optimization. The optimal quality, convergence speed, and statistical hypothesis testing of our algorithm are superior to those previous high performance algorithms such as RCCRO, HP-CRO2, and OCRO.

1. Introduction

Various optimization approaches have been used to solve existing global optimization problems in science, engineering, and related fields. Evolutionary algorithms (EAs), inspired by the biological evolution and social behavior of organisms, are the most powerful among optimization frameworks. Several EAs have been successfully used to solve global optimization problems. Genetic algorithms (GAs) [1, 2] were developed based on the natural process of evolution. Particle swarm optimization (PSO) [3] was inspired by the social behavior of bird flocking or fish schooling. Memetic algorithms (MAs) [4] are similar to GAs except that MAs initiate a local search before undergoing the evolutionary process [5]. Ant colony optimization [6] resembles PSO based on the capability of ants to find the shortest route between their nest and a food source. Artificial bee colony [7, 8] was inspired by the echolocation behavior of bees. Harmony search [9] was based on the natural musical performance of a musician searching for an optimal state of harmony.

A recent study has proposed a new chemical reaction-inspired EA named “chemical reaction optimization” [10, 11]. CRO is a low-latency algorithm that mimics molecular

interactions in chemical reactions to reach a low-energy stable state. An effective version of CRO is the real-coded chemical reaction optimization (RCCRO) [12], which can process a large set of continuous problems [13–17]. HP-CRO, a hybrid algorithm based on PSO and CRO, produces more distinguished results compared with RCCRO [18]. However, orthogonal chemical reaction optimization (OCRO) algorithms for global numerical optimization problems [19] are the best when compared with RCCRO and HP-CRO. OCRO is efficient in solving high-dimension functions but not in solving low-dimensional functions.

The current study proposes a hybrid mutation chemical reaction optimization algorithm for global numerical optimization (MCRO). This strategy can be used to solve high-dimension and low-dimension functions better and faster. The proposed framework combines CRO with a turning operator and a mutation operator. Our algorithm was determined by choosing the best solution that estimates the performance of the combination with three types of mutation operators (uniform, nonuniform, and polynomial).

This paper is organized as follows. Section 2 provides a brief review of preliminary studies on CRO and mutation

operator. Section 3 presents the proposed approach: MCRO. Section 4 describes the evaluation methods and simulation results and discusses the comparisons of algorithms. Section 5 concludes this paper and suggests proposals for future work.

2. Preliminary

2.1. Optimization. In science, mathematics, engineering, and other related fields, optimization is the selection of an optimal solution from a set of feasible alternatives. In general, an optimization problem includes minimizing or maximizing a function by systematically selecting input values from a given feasible set [20]. The problem function $f(x)$ is scalar, where a variable x demonstrates a particular solution and x is usually a vector of n components. The n components specify the dimensions of f . An optimization problem can be subjected to a vector of constraints $C = \{c_1, c_2, \dots, c_m\}$ that limits the feasible region, where m corresponds to the total number of constraints [12]. The aim of optimization is to generate the optimal solution and manage the convergence speed of a problem function. Our research focuses on function minimization; the goal of minimum optimization is to find the minimum solution $\hat{s} \in S$ and $(\hat{s}) \leq f(s)$, $\forall s \in S$.

2.2. CRO. In principle, chemical reactions are governed by two rules of thermodynamics. First, energy cannot be created or destroyed but can be transformed from one entity to another; second, the entropy of a system tends to increase, where entropy is the measure of the degree of disorder [21]. CRO was inspired by the nature of chemical reactions and mimics the interactions of molecules in a chemical reaction to reach a low-energy stable state. Potential energy is the energy stored in a molecule with respect to its molecular configuration; the system becomes disordered when potential energy is converted to other forms [11]. Molecules stored in a container are vital to the manipulation of agents. Each molecule contains a profile that includes several attributes, such as molecular structure (ω), current potential energy (PE), and current kinetic energy (KE). In CRO reaction, the initial reactants in high-energy states incur a sequence of collisions. Molecules collide either with other molecules or with the walls of the container, pass through energy barriers, and become the final products in low-energy stable states. Four types of elementary CRO reactions capture the transition of high-energy molecules to stable states [10].

On-wall ineffective collision is the reaction created when a molecule hits the wall of a container and then bounces back. This reaction only slightly changes the molecular structure ω when $PE_\omega + KE_\omega \geq PE_{\hat{\omega}}$ occurs. The modification of molecular structure $\hat{\omega}$ is processed by neighborhood search operator as $\hat{\omega} \setminus \text{in neighborhood}(\omega)$. The central energy buffer is updated by extracting and storing a certain portion of its KE. The profile of the molecule is updated as $KE_{\hat{\omega}} = (PE_\omega - PE_{\hat{\omega}} + KE_\omega) \times r$, where r is a random number that $r \in [KELossRate, 1]$. The new PE is calculated using the new ω or $\hat{\omega}$.

Intermolecular ineffective collision refers to two or more molecules that collide with each other and then separate. The profiles of the molecules and the central energy buffer are updated when $PE_{\hat{\omega}_1} + PE_{\hat{\omega}_2} + KE_{\hat{\omega}_1} + KE_{\hat{\omega}_2} \geq PE_{\hat{\omega}_1} + PE_{\hat{\omega}_2}$. The number of molecules is unaltered after the collision, but the molecular structures are produced from their own neighborhoods by neighborhood search operator. Similar to on-wall ineffective collision, this reaction also slightly changes the molecular structure.

Decomposition occurs when a molecule hits the wall of a container and then splits into two or more molecules. This elementary reaction is applied to finish local search and to explore other regions. The profiles of the molecules and the central energy buffer are updated when $PE_\omega + KE_\omega \geq PE_{\hat{\omega}_1} + PE_{\hat{\omega}_2}$ and when the energy buffer is sufficient. This reaction significantly alters the molecular structures of the resultant molecules.

Synthesis represents the situation when two or more molecules collide and mingle to a single molecule. The profiles of the molecules and the central energy buffer are updated when $PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2} \geq PE_{\hat{\omega}}$. This reaction strongly and significantly alters the resultant molecular structure.

The algorithm of CRO is shown as Algorithm 1 [12]. On-wall and intermolecular ineffective collisions are local searches, whereas decomposition and synthesis are global searches.

2.3. Mutation Operator. Mutation is a natural process that changes a DNA sequence to a new variation endowing individual survival advantages in biology. Mutation operator, which has been applied to GAs, simulates the natural biological evolution of species by exchanging information and producing offspring chromosomes [1, 2]. In GAs, mutation operator is designed to work on an individual chosen; this individual is randomly selected or chosen by defined condition. The new gene value is modified based on user-definable probability or other conditions such as a distribution. GAs containing mutation can widely maintain offspring chromosomes, instead of being limited to the genes available in the initial population. Therefore GAs with mutation can generate better solution and prevent premature convergence compared to GAs without mutation.

The concept of MCRO is initiated by supposing that advantage of mutation operator would empower global numerical optimization while combining with CRO. Then we reformed the mutation operator algorithm in accordance with CRO infrastructure, such as replacing an individual in GAs by a molecule (M), assigning probability and distribution similar to user-definable probability or condition in GAs, and changing gene value of GAs with molecular structure (ω). In CRO, a molecular structure contains amount number of moles and a pair of moles is presented as a set of $\omega = \{\omega_1, \omega_2, \dots, \omega_v\}$, where v is the number of moles. Mutation operator algorithm generates the new molecular structure ($\hat{\omega}$) based on mutation operator formula relating to mutation type. There are many types of mutation and these types depend on the representation itself. Several mutation

Input: Objective function f , constraints, and the dimensions of the problem

```

(1) /* initialization phase */
(2) Assign parameter value to PopSize, InitialKE, StepSize, buffer, KELossRate, On-wallColl,
(3) DecThres, SynThres, The set of molecules in this Container are molecules 1, 2, ..., PopSize
(4) for each of the molecule do
(5)   Assign a random solution to the molecular structure  $\omega$ . Calculate the PE by  $f(\omega)$  and evaluate.
(6)   Assign the KE with InitialKE
(7) end for
(8) /* Iterations phase */
(9) while (the stopping criteria not met) do
(10)   Get  $r$  randomly in interval [0, 1]
(11)   if ( $r > \text{On-wallColl}$ ) then
(12)     select a molecule  $M$  from Container Randomly
(13)     if (Decomposition criterion met) then
(14)       Decomposition
(15)       if Success then Add a new molecule  $M'$  to Container
(16)     else
(17)       On-wallIneffectiveCollision
(18)     end if
(19)   else
(20)     select two molecule  $M_1$  and  $M_2$  from Container Randomly
(21)     if (Synthesis criterion met) then
(22)       Synthesis
(23)       if Success then Remove molecule  $M_2$  from Container
(24)     else
(25)       Inter-molecularIneffectiveCollision
(26)     end if
(27)   end if
(28)   evaluate and keep any new minimum solution
(29) end while
(30) /* The Output phase */
(31) Output the best solution and its function value.

```

ALGORITHM 1: Chemical reaction optimization (CRO).

Input: a molecular structure of solution M

```

(1) Assign probability  $P$ , distribution  $D$ ,  $\nu$  = Number of moles
    $U$  = function upper bound;  $L$  = function lower Bound
(2) For  $i = 0$  to  $\nu$  do
(3)   Get  $r$  randomly in interval [0, 1]
(4)   If  $r < (P/\nu)$  then
(5)      $\hat{\omega}_i$  = do mutation operator Formula
(6)     If  $\hat{\omega}_i < L$  then  $\hat{\omega}_i = L$ 
(7)     If  $\hat{\omega}_i > U$  then  $\hat{\omega}_i = U$ 
(8)   End if
(9) End For
(10) Output a new molecular structure =  $\hat{\omega}$ 

```

ALGORITHM 2: Mutation operator algorithm.

types are proposed such as uniform, nonuniform [22], and polynomial mutations [23, 24]. The mutation operator algorithm is demonstrated as Algorithm 2.

3. Proposed Approach: MCRO

Mutation chemical reaction optimization (MCRO), which is the improved hybrid of CRO and two adjustment operators,

generates various molecular structures to produce high-quality results and accelerate convergence. The turning operator and the mutation operator will be explained in this section.

3.1. Turning Operator. Turning operator is a new operator that is merged into a subalgorithm named neighborhood search operator or $N(\cdot)$. Neighborhood search operator

is used in three types of elementary reactions (on-wall ineffective collision, intermolecular ineffective collision, and decomposition) to transform the molecular structure from the neighborhood of the operand [12]. The new molecular structures of the solution are calculated as Formula (1), and turning operator is generated once for an objective function (f_i) by using Formula (2). This new operator can highly improve the optimal quality and reliability of the algorithm:

NewStructure

$$= (\text{previousStructure} + \text{RandomGaussian} \cdot \text{stepSize}) \quad (1)$$

· Turning operator of f_i ,

Turning operator of f_i

$$= \{ \text{random number interval } [1, 1E - 300] \}, \quad (2)$$

$(1 \leq i \leq 23).$

3.2. Mutation Operator. Mutation operator is migrated into MCRO to improve solution diversity and accelerate convergence by generating various molecular structures. These structures are generated by using the processing mutation formula. This action increases the probability of finding the optimal solution and avoids being trapped into a local optimal solution; therefore, it accelerates convergence. In our research, the best mutation operator that matches our algorithm was selected from three types of mutation operators (uniform, nonuniform, and polynomial).

Uniform and nonuniform mutations are presented in basic GAs and their extending development of algorithm [22, 25]. Polynomial mutation is a popular one that was first introduced in Nondominated Sorting Genetic Algorithm (NSGA) and NSGA-II [23, 24]. Uniform, nonuniform, and polynomial mutations are calculated as Formulas (3), (4), and (5), respectively. As mutation operator algorithm mentioned in Section 2.3, input data is a set of molecular structures presented as $\omega = \{\omega_1, \omega_2, \dots, \omega_v\}$, where v is the number of moles, ω_i is a member of ω , $1 \leq i \leq v$. Likewise output data is a set of molecular structures presented as $\hat{\omega}$. According to opportuneness, the variable names such as ω and $\hat{\omega}$ maybe present differently when they appear in other algorithms such as $temp\omega$, $temp\hat{\omega}$, $temp\omega_1$, and $temp\omega_2$:

$$\hat{\omega}_i = [(\text{random number } [0-1] - 0.5) \times \text{perturbation}] + \omega_i, \quad (3)$$

$$\hat{\omega}_i = \left[(\text{UL} - \omega_i) \cdot (1.0 \times \text{random number } [0-1]^{1.0 - \text{FE}/\text{FELimit}})^{\text{perturbation}} \right] + \omega_i$$

$$\text{UL} = \begin{cases} \text{upper bound,} & \text{random number } [0-1] < 0.5 \\ \text{lower bound,} & \text{random number } [0-1] \geq 0.5 \end{cases} \quad (4)$$

$$\hat{\omega}_i = \hat{\omega}_i + \Delta Q^2$$

$$\Delta Q = \begin{cases} (2.0 \times \text{rnd} + (1.0 - 2.0 \times \text{rnd}) \times (1.0 - \Delta 1)^D)^{(1.0/(D+1.0))}, & \text{rnd : random number } [0-1] < 0.5 \\ 1.0 - (2.0 \times (1.0 - \text{rnd}) + 2.0 \times (\text{rnd} - 0.5) \times (1.0 - \Delta 2)^{(D+1.0)})^{(1.0/(D+1.0))}, & \text{rnd : random number } [0-1] \geq 0.5, \end{cases} \quad (5)$$

$$\text{where } \Delta 1 = \frac{(\hat{\omega}_i - \text{lower bound})}{\text{distance (upper bound and lower bound)}}, \quad \Delta 2 = \frac{(\text{upper bound} - \hat{\omega}_i)}{\text{distance (upper bound and lower bound)}}.$$

Generally, the optimization method consists of three phases: initialization, iteration, and output, which are all included in our proposed approach. We applied mutation operator to the initialization and iteration phases of the algorithm. A mutation operator was applied to each new molecule in the initialization phase. In the iteration phase, we processed mutation operator in all types of elementary reaction, namely, on-wall ineffective collision, intermolecular ineffective collision, decomposition, and synthesis. Mutation operator was incorporated to change the molecular structure. We compared the results before and after the performance of mutation operator and selected the better result. We believe mutation is one potential component to improve the performance of MCRO, since mutation operator can

spread the search space by randomly sampling new points. Furthermore this mutation operator algorithm increases the chance of generating more powerful result not less than twice of original CRO for every elementary reaction. Following are the four elementary reactions' explanations.

In on-wall ineffective collision of MCRO, a molecule M is chosen. The first new molecular structure ($\hat{\omega}$) of M is generated by neighborhood search operator which includes turning operator as mentioned in Section 3.1 and then the first fitness or $\text{PE}_{\hat{\omega}}$ is calculated by $f(\hat{\omega})$. The new KE_{ω} is generated and *buffer* is updated when the condition $\text{PE}_{\omega} + \text{KE}_{\omega} \geq \text{PE}_{\hat{\omega}}$ occurs. Next the second new molecular structure ($temp\omega$) is produced by computing mutation operator algorithm and then the second fitness or $temp\text{PE}$ is calculated

Input: a molecule M and buffer

- (1) Obtain $\hat{\omega} = \text{Neighbor}(\omega)$ with turning operator
- (2) Calculate $\text{PE}_{\hat{\omega}}$ by $f(\hat{\omega})$
- (3) **if** $\text{PE}_{\omega} + \text{KE}_{\omega} \geq \text{PE}_{\hat{\omega}}$ **then**
- (4) Get r randomly in interval $[\text{KELossRate}, 1]$
- (5) $\text{KE}_{\hat{\omega}} = (\text{PE}_{\omega} + \text{KE}_{\omega} - \text{PE}_{\hat{\omega}}) \times r$
- (6) Update $\text{buffer} = \text{buffer} - \text{KE}_{\hat{\omega}}$
- (7) **do** Mutation of ω to $\text{temp}\omega$
- (8) **Calculate** the tempPE by $f(\text{temp}\omega)$
- (9) **if** tempPE better than $\text{PE}_{\hat{\omega}}$ **then** Replace $\hat{\omega}$ with $\text{temp}\omega$, $\text{PE}_{\hat{\omega}}$ with tempPE
- (10) Update the profile of M by $\omega = \hat{\omega}$, $\text{PE}_{\omega} = \text{PE}_{\hat{\omega}}$ and $\text{KE}_{\omega} = \text{KE}_{\hat{\omega}}$
- (11) **end if**

Output M and buffer

ALGORITHM 3: On-wall ineffective collision of MCRO.

Input: molecules M_1, M_2

- (1) Obtain $\hat{\omega}_1 = \text{Neighbor}(\omega_1)$ and $\hat{\omega}_2 = \text{Neighbor}(\omega_2)$ with turning operator
- (2) Calculate $\text{PE}_{\hat{\omega}_1}$ and $\text{PE}_{\hat{\omega}_2}$
- (3) $\text{temp}_2 = (\text{PE}_{\hat{\omega}_1} + \text{PE}_{\hat{\omega}_2} + \text{KE}_{\hat{\omega}_1} + \text{KE}_{\hat{\omega}_2}) - (\text{PE}_{\hat{\omega}_1} + \text{PE}_{\hat{\omega}_2})$
- (4) **if** $\text{PE}_{\hat{\omega}_1} + \text{PE}_{\hat{\omega}_2} + \text{KE}_{\hat{\omega}_1} + \text{KE}_{\hat{\omega}_2} \geq \text{PE}_{\hat{\omega}_1} + \text{PE}_{\hat{\omega}_2}$ **then**
- (5) Get r randomly in interval $[0, 1]$
- (6) $\text{KE}_{\hat{\omega}_1} = \text{temp}_2 \times p$ and $\text{KE}_{\hat{\omega}_2} = \text{temp}_2 \times (1 - p)$
- (7) **end if**
- (8) **do** Mutation of ω_1 to $\text{temp}\omega_1$ and ω_2 to $\text{temp}\omega_2$
- (9) **Calculate** the tempPE_1 by $f(\text{temp}\omega_1)$ and tempPE_2 by $f(\text{temp}\omega_2)$
- (10) **If** tempPE_1 better than $\text{PE}_{\hat{\omega}_1}$: Replace $\hat{\omega}_1$ with $\text{temp}\omega_1$ and $\text{PE}_{\hat{\omega}_1}$ with tempPE_1
- (11) **If** tempPE_2 better than $\text{PE}_{\hat{\omega}_2}$: Replace $\hat{\omega}_2$ with $\text{temp}\omega_2$ and $\text{PE}_{\hat{\omega}_2}$ with tempPE_2
- (12) Update the profile of M_1 by $\omega_1 = \hat{\omega}_1$, $\text{PE}_{\hat{\omega}_1} = \text{PE}_{\hat{\omega}_1}$ and $\text{KE}_{\hat{\omega}_1} = \text{KE}_{\hat{\omega}_1}$
- (13) Update the profile of M_2 by $\omega_2 = \hat{\omega}_2$, $\text{PE}_{\hat{\omega}_2} = \text{PE}_{\hat{\omega}_2}$ and $\text{KE}_{\hat{\omega}_2} = \text{KE}_{\hat{\omega}_2}$
- (14) **Output** M_1 and M_2

ALGORITHM 4: Intermolecular ineffective collision of MCRO.

by $f(\text{temp}\omega)$. The profile of molecule M is updated as the best solution by comparing the first fitness ($\text{PE}_{\hat{\omega}}$) and the second fitness (tempPE). Therefore this action contains two probabilities and achieves the best solution for each time while there is only one choice for original CRO. The on-wall ineffective collision of MCRO is shown in Algorithm 3.

Intermolecular ineffective collision of MCRO algorithm starts by selecting two molecules M_1 and M_2 . Subsequently, neighborhood search operator with turning operator transforms the molecular structures of M_1 and M_2 from ω_1 and ω_2 to $\hat{\omega}_1$ and $\hat{\omega}_2$, respectively, and evaluates $\text{PE}_{\hat{\omega}_1}$ by $f(\hat{\omega}_1)$ and $\text{PE}_{\hat{\omega}_2}$ by $f(\hat{\omega}_2)$. The new KE are generated as $\text{KE}_{\hat{\omega}_1}$ when $\text{PE}_{\hat{\omega}_1} + \text{PE}_{\hat{\omega}_2} + \text{PE}_{\hat{\omega}_1} + \text{KE}_{\hat{\omega}_2} \geq \text{PE}_{\hat{\omega}_1} + \text{PE}_{\hat{\omega}_2}$ occurs. Furthermore, the algorithm executes mutation operator to build two new molecular structures, $\text{temp}\omega_1$ and $\text{temp}\omega_2$, and processes fitness of two molecules tempPE_1 by $f(\text{temp}\omega_1)$ and tempPE_2 by $f(\text{temp}\omega_2)$. Meanwhile, the best fitness between $\text{PE}_{\hat{\omega}_1}$ and tempPE_1 is selected to be the optimal solution of M_1 , similar to the optimal solution of M_2 is the best fitness between $\text{PE}_{\hat{\omega}_2}$ and tempPE_2 , and then the profiles of molecules M_1 and M_2 are updated. Prove that there are two choices to estimate the most outstanding result for this

reaction that acted to a molecule; therefor this elementary reaction of MCRO consists of four chances to meet the optimum solution for a time, while the same reaction of original CRO is just giving two choices (one choice for a molecule). Algorithm 4 illustrates intermolecular ineffective collision of MCRO algorithm.

Decomposition of MCRO algorithm begins by selecting a molecules M from container. Copy a molecular structure (ω) of M to two new variables ω_1 and ω_2 . Next the molecular structures ω_1 and ω_2 are transformed to $\hat{\omega}_1$ and $\hat{\omega}_2$, respectively, by computing neighborhood search operator (with turning operator), and furthermore evaluate the fitness $\text{PE}_{\hat{\omega}_1}$ as $f(\hat{\omega}_1)$ and the fitness $\text{PE}_{\hat{\omega}_2}$ as $f(\hat{\omega}_2)$. The profiles of molecules are updated when $\text{PE}_{\omega} + \text{KE}_{\omega} \geq \text{PE}_{\hat{\omega}_1} + \text{PE}_{\hat{\omega}_2}$ as follows: set $\text{KE}_{\hat{\omega}_1} = (\text{PE}_{\omega} + \text{KE}_{\omega} - \text{PE}_{\hat{\omega}_1} - \text{PE}_{\hat{\omega}_2}) \times k$ and $\text{KE}_{\hat{\omega}_2} = \text{PE}_{\omega} + \text{KE}_{\omega} - \text{PE}_{\hat{\omega}_1} - \text{PE}_{\hat{\omega}_2} \times (1 - k)$ when k is randomly in interval $[0, 1]$ and a new molecule M' is created. In case of *not* ($\text{PE}_{\omega} + \text{KE}_{\omega} \geq \text{PE}_{\hat{\omega}_1} + \text{PE}_{\hat{\omega}_2}$) and the energy buffer is sufficient set $\text{KE}_{\hat{\omega}_1} = (\text{temp1} + \text{buffer}) \times r_1 \times r_2$ and $\text{KE}_{\hat{\omega}_2} = (\text{temp1} + \text{buffer} - \text{KE}_{\hat{\omega}_1}) \times r_3 \times r_4$, when r_1, r_2, r_3 , and r_4 are randomly in interval $[0, 1]$; in addition the central energy buffer is updated and a new molecule M' is created. If there exists a new molecule M'

Input: A molecule M and buffer

- (1) Obtain ω_1 and ω_2 from ω
- (2) Obtain $\dot{\omega}_1 = \text{Neighbor}(\omega_1)$ and $\dot{\omega}_2 = \text{Neighbor}(\omega_2)$ with turning operator
- (3) Calculate $\text{PE}_{\dot{\omega}_1}$ and $\text{PE}_{\dot{\omega}_2}$
- (4) $\text{temp}_1 = \text{PE}_{\omega} + \text{KE}_{\omega} - \text{PE}_{\dot{\omega}_1} - \text{PE}_{\dot{\omega}_2}$
- (5) $\text{Success} = \text{TRUE}$
- (6) **if** $\text{PE}_{\omega} + \text{KE}_{\omega} \geq \text{PE}_{\dot{\omega}_1} + \text{PE}_{\dot{\omega}_2}$ **then**
- (7) Get k randomly in interval $[0, 1]$
- (8) $\text{KE}_{\dot{\omega}_1} = \text{temp}_1 \times k$ and $\text{KE}_{\dot{\omega}_2} = \text{temp}_1 \times (1 - k)$
- (9) Create new molecules M'
- (10) **else if** $\text{temp}_1 + \text{buffer} \geq 0$ **then**
- (11) Get r_1, r_2, r_3 , and r_4 randomly in interval $[0, 1]$
- (12) $\text{KE}_{\dot{\omega}_1} = (\text{temp}_1 + \text{buffer}) \times r_1 \times r_2$
- (13) $\text{KE}_{\dot{\omega}_2} = (\text{temp}_1 + \text{buffer} - \text{KE}_{\dot{\omega}_1}) \times r_3 \times r_4$
- (14) Update $\text{buffer} = \text{temp}_1 + \text{buffer} - \text{KE}_{\dot{\omega}_1} - \text{KE}_{\dot{\omega}_2}$
- (15) Create new molecules M'
- (16) **end if**
- (17) **else**
- (18) $\text{Success} = \text{FALSE}$
- (19) **end if**
- (20) **If** $\text{Success} = \text{TRUE}$
- (21) **do** Mutation of $\dot{\omega}_1$ to $\text{temp}\dot{\omega}_1$ and $\dot{\omega}_2$ to $\text{temp}\dot{\omega}_2$
- (22) **Calculate** the tempPE_1 by $f(\text{temp}\dot{\omega}_1)$ and tempPE_2 by $f(\text{temp}\dot{\omega}_2)$
- (23) **If** tempPE_1 better than $\text{PE}_{\dot{\omega}_1}$ **then** Replace $\dot{\omega}_1$ with $\text{temp}\dot{\omega}_1$ and $\text{PE}_{\dot{\omega}_1}$ with tempPE_1
- (24) **If** tempPE_2 better than $\text{PE}_{\dot{\omega}_2}$ **then** Replace $\dot{\omega}_2$ with $\text{temp}\dot{\omega}_2$ and $\text{PE}_{\dot{\omega}_2}$ with tempPE_2
- (25) Assign $\dot{\omega}_1, \text{PE}_{\dot{\omega}_1}, \text{KE}_{\dot{\omega}_1}$ to the profile of M
- (26) Assign $\dot{\omega}_2, \text{PE}_{\dot{\omega}_2}, \text{KE}_{\dot{\omega}_2}$ to the profile of M'
- (27) **end if**
- (28) **Output** M and $M', \text{Success}$ and buffer

ALGORITHM 5: Decomposition of MCRO.

($\text{Success} = \text{TRUE}$), the two new molecular structures $\text{temp}\dot{\omega}_1$ and $\text{temp}\dot{\omega}_2$ are established by operated mutation operator algorithm and then the fitness of tempPE_1 and tempPE_2 are evaluated by $f(\text{temp}\dot{\omega}_1)$ and $f(\text{temp}\dot{\omega}_2)$, respectively. The first minimum fitness chooses the best one from $\text{PE}_{\dot{\omega}_1}$ or tempPE_1 , and the second minimum fitness selects the best one between $\text{PE}_{\dot{\omega}_2}$ and tempPE_2 . Furthermore profiles of M and M' are updated. This concludes that there are four opportunities to reach the best result in decomposition of MCRO similar to intermolecular ineffective collision of MCRO; in other words, they are twice the same reaction in original CRO. Decomposition of MCRO is represented as Algorithm 5.

In synthesis of MCRO algorithm, two molecules M_1 and M_2 are chosen. Secondly, neighborhood search operator (with turning operator) generated the two molecular structures of M_1 and M_2 from ω_1 and ω_2 to $\dot{\omega}_1$ and $\dot{\omega}_2$, respectively, and the first fitness or $\text{PE}_{\dot{\omega}_1}$ is calculated by $f(\dot{\omega}_1)$. Subsequently, the new $\text{KE}_{\dot{\omega}_1}$ is computed when $\text{PE}_{\omega_1} + \text{PE}_{\omega_2} + \text{KE}_{\omega_1} + \text{KE}_{\omega_2} \geq \text{PE}_{\dot{\omega}_1} + \text{PE}_{\dot{\omega}_2} + \text{KE}_{\dot{\omega}_1} + \text{KE}_{\dot{\omega}_2} - \text{PE}_{\dot{\omega}_1}$. Afterward mutation operator algorithm generates $\dot{\omega}_1$ to $\text{temp}\dot{\omega}_1$, and the fitness generated by mutation operator algorithm or tempPE is calculated by $f(\text{temp}\dot{\omega}_1)$. Finally the profile of molecule M_1 is updated by choosing the best solution between the fitness ($\text{PE}_{\dot{\omega}_1}$) and the fitness generated by mutation operator (tempPE). There are two probabilities included in this elementary reaction while synthesis

of original CRO provides only one. Synthesis of MCRO algorithm is presented as Algorithm 6.

4. Simulation Results and Discussion

4.1. Benchmarks and Parameters. The proposed MCRO was simulated to solve the 23 objective functions based on RCCRO [12]. The benchmark objective functions are classified into three categories, as shown in Table 1.

Category I (High-Dimensional Unimodal Functions). Category I is composed of high-dimensional functions f_1 – f_7 and includes 30 molecules in each container. These functions are easy to solve because each objective function has only one global minimum.

Category II (High-Dimensional Multimodal Functions). Category II provides high-dimensional functions f_8 – f_{13} and includes 30 molecules in each container. These functions consist of many local minimums and can solve the most difficult problem.

Category III (Low-Dimensional Multimodal Functions). Category III includes low-dimensional functions f_{14} – f_{23} . In each container of functions, f_{14} , f_{16} , f_{17} , and f_{18} contain 2 molecules; f_{15} , f_{21} , f_{22} , and f_{23} contain 4 molecules; f_{19}

TABLE 1: The benchmark objective functions.

Objective function	Number of molecules	Lower-upper bound	$f_{\min}$	Name
<i>Category I</i>				
$f_1(x) = \sum_{i=1}^n x_i^2$	30	$[-100, 100]^n$	0	Sphere model
$f_2(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	30	$[-10, 10]^n$	0	Schwefel's problem 2.22
$f_3(x) = \sum_{i=1}^n \left(\sum_{j=1}^i x_j \right)^2$	30	$[-100, 100]^n$	0	Schwefel's problem 1.2
$f_4(x) = \max_i \{ x_i , 1 \leq i \leq n\}$	30	$[-100, 100]^n$	0	Schwefel's problem 2.21
$f_5(x) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right)$	30	$[-30, 30]^n$	0	Generalized Rosenbrock's function
$f_6(x) = \sum_{i=1}^n (x_i + 0.5)^2$	30	$[-100, 100]^n$	0	Step function quartic
$f_7(x) = \sum_{i=1}^n ix_i^4 + \text{random}[0, 1]$	30	$[-1.28, 1.28]^n$	0	Function with noise
<i>Category II</i>				
$f_8(x) = -\sum_{i=1}^n \left(x_i \sin \left(\sqrt{ x_i } \right) \right)$	30	$[-500, 500]^n$	-12569.5	Generalized Schwefel's problem 2.26
$f_9(x) = \sum_{i=1}^n \left(x_i^2 - \cos(2\pi x_i) + 10 \right)$	30	$[-5.12, 5.12]^n$	0	Generalized Rastrigin's function
$f_{10}(x) = -20 \exp(-0.2) \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} - \exp \left(\frac{1}{n} \sum_{i=1}^n \cos 2\pi x_i \right) + 20 + e$	30	$[-32, 32]^n$	0	Ackley's function
$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos \left(\frac{x_i}{\sqrt{i}} \right) + 1$	30	$[-600, 600]^n$	0	Generalized Griewank function
$f_{12}(x) = \frac{\pi}{n} \left\{ 10 \sin^2(\pi y_1) + \sum_{i=1}^{29} (y_i - 1)^2 \left[1 + 10 \sin^2(\pi y_{i+1}) \right] + (y_n - 1)^2 \right\} + \sum_{i=1}^{30} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4} (x_i + 1)$	30	$[-50, 50]^n$	0	Generalized penalized functions
$f_{13}(x) = 0.1 \left\{ \sin^2(\pi 3x_1) + \sum_{i=1}^{29} (x_i - 1)^2 \left[1 + \sin^2(\pi 3x_{i+1}) \right] + (x_n - 1)^2 \left[1 + \sin^2(2\pi 30) \right] \right\} + \sum_{i=1}^{30} u(x_i, 5, 100, 4)$	30	$[-50, 50]^n$	0	
<i>Remark:</i> in f_{12} and f_{13} , $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leq x_i \leq a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$				

TABLE I: Continued.

Objective function	Number of molecules	Lower-upper bound	$f_{\min}$	Name
<i>Category III</i>				
$f_{14}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})^6} \right]^{-1}$	2	$[-65.536, 65.536]^n$	1	Shekel's foxholes function
$f_{15}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	4	$[-5, 5]^n$	0.0003075	Kowalik's function
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]^n$	-1.0316285	Six-hump camel-back function
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi^2}x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10$	2	$[-5, 10] \times [0, 15]$	0.398	Branin function
$f_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2) \right] \times [30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 32x_1x_2 + 27x_2^2)]$	2	$[-2, 2]^n$	3	Goldstein-Price function
$f_{19}(x) = - \sum_{i=1}^4 c_i \exp \left[- \sum_{j=1}^4 a_{ij} (x_j - p_{ij})^2 \right]$	3	$[0, 1]^n$	-3.86	Hartman's family
$f_{20}(x) = - \sum_{i=1}^4 c_i \exp \left[- \sum_{j=1}^6 a_{ij} (x_j - p_{ij})^2 \right]$	6	$[0, 1]^n$	-3.32	
$f_{21}(x) = - \sum_{i=1}^5 [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	$[0, 10]^n$	-10	Shekel's family
$f_{22}(x) = - \sum_{i=1}^7 [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	$[0, 10]^n$	-10	
$f_{23}(x) = - \sum_{i=1}^{10} [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	$[0, 10]^n$	-10	


```

Input: molecules  $M_1, M_2$ 
(1) Obtain  $\omega_1$  from  $\omega_1$  and  $\omega_2$  from  $\omega_2$ 
(2) Calculate  $PE_{\omega_1}$ 
(3) if  $PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2} \geq PE_{\omega_1}$  then
(4)    $Success = TRUE$ 
(5)    $KE_{\omega_1} = PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2} - PE_{\omega_1}$ 
(6)   do Mutation of  $\omega_1$  to  $temp\omega_1$ 
(7)   Calculate the  $tempPE$  by  $f(temp\omega_1)$ 
(8)   if  $tempPE$  better than  $PE_{\omega_1}$  then Replace  $\omega_1$  with  $temp\omega_1$  and  $PE_{\omega_1}$  with  $tempPE$ 
(9)   Assign  $\omega_1$   $PE_{\omega_1}$  and  $KE_{\omega_1}$  to the profile of  $M_1$ 
(10) else
(11)    $Success = FALSE$ 
(12) end if
(13) Output  $M_1$  and  $Success$ 

```

ALGORITHM 6: Synthesis of MCRO.

TABLE 2: MCRO parameters.

Parameter	<i>popsize</i>	<i>InitialKE</i>	<i>Buffer</i>	<i>KELossRate</i>	<i>On-wallColl</i>	<i>DecThres</i>	<i>SynThres</i>
Category I (f_1-f_7)	10	1000	0	0.1	0.2	150000	10
Category II (f_8-f_{13})	20	10000000	100000	0.1	0.2	150000	10
Category III ($f_{14}-f_{23}$)	100	1000	0	0.1	0.2	150000	10

contains 3 molecules; and f_{20} contains 6 molecules. These functions comprise several local minimums.

Several parameters contained in MCRO include the main algorithm and the mutation operator parameters. Table 2 illustrates the main parameters in our simulations: *popsize*, *InitialKE*, *buffer*, *KELossRate*, *On-wallColl*, *DecThres*, *SynThres*, and *StepSize*.

The mutation operator parameters in our experiment are *probability* and *DistributionIndex*. We set *probability* = 0.6 and *DistributionIndex* = 0.9 for every objective function.

4.2. MCRO Solution. To obtain the most appropriate MCRO solution, we considered three types of mutation operators: polynomial, uniform, and nonuniform. Our approach estimates start with alloyed CRO with turning operator, consequently integrating mutation operator to the algorithm. The first, second, and third options are the algorithms with polynomial, uniform, and nonuniform mutations, respectively. We generated the results of these three solutions by running each objective function 25 times. The optimal quality of the solution of each function is the mean of the best global minimum at 25 repeats. If the means of the results are equal, the second key for comparison is the standard deviation. All simulations included in Section 4.3 are performed on the same personal computer and same computing environment with computer Intel Core i7, CPU 3.10 GHz, and Ram 4 GB.

Table 3 compares the results of Category I functions for the three solutions. All of the solutions yield good results for f_1 , f_2 , f_3 , f_4 , and f_6 . The algorithm with polynomial mutation is the leader for f_5 , and the algorithm with uniform mutation generates the best result for f_7 . The average result of Category I functions is the same for the three solutions.

Table 4 compares the results of Category II functions for the three solutions. The results of the three solutions are as good as those of the two other solutions in f_9 , f_{10} , and f_{11} . The solution with polynomial mutation performs best for f_{12} and f_{13} . The solution with nonuniform mutation generates the best result for f_8 . The average result of high-dimensional multimodal functions is led by the solution combined with polynomial mutation, nonuniform mutation, and then uniform mutation.

Table 5 compares the results of Category III functions. The solution with polynomial mutation computes the best result for f_{14} , f_{16} , f_{17} , f_{18} , and f_{19} . Meanwhile, the solution with uniform mutation is the leader for f_{15} , f_{20} , f_{21} , f_{22} , and f_{23} , and the solution with nonuniform mutation performs best for f_{15} . The polynomial solution does not generate the best result for the remaining functions, but it produces close results to those of the leader for the first five functions. The average ranking of Category III functions for the three solutions in order of solution is polynomial mutation, uniform mutation, and nonuniform mutation.

The overall ranking is shown in the bottom of Table 5. The solution with polynomial mutation derives the most excellent results, followed successively by uniform mutation and nonuniform mutation. Hence, we prefer polynomial mutation operator for MCRO.

4.3. Experimental Method and Results. The aim of the present experiment is to evaluate the performance of MCRO by comparing it with three previously published algorithms: RCCRO4 [12], the best version of RCCRO; HP-CRO2 [18], the best version of HP-CRO and a hybrid of CRO and PSO; and OCRO [19], which represents the best results among

TABLE 3: Comparisons of the result of the three solutions for Category I (f_1-f_7) (the best results are marked in bold).

Function	Result of	Algorithm with mutation		
		Polynomial	Uniform	Nonuniform
f_1	Mean	0	0	0
	StDev.	0	0	0
	Rank	1	1	1
f_2	Mean	0	0	0
	StDev.	0	0	0
	Rank	1	1	1
f_3	Mean	0	0	0
	StDev.	0	0	0
	Rank	1	1	1
f_4	Mean	0	0	0
	StDev.	0	0	0
	Rank	1	1	1
f_5	Mean	2.85E + 01	2.85E + 01	2.85E + 01
	StDev.	1.05E - 02	1.51E - 02	1.38E - 01
	Rank	1	3	2
f_6	Mean	0	0	0
	StDev.	0	0	0
	Rank	1	1	1
f_7	Mean	2.57E - 04	5.44E - 06	1.94E - 05
	StDev.	2.07E - 04	6.80E - 06	1.71E - 05
	Rank	3	1	2
Average rank	(f_1-f_7)	1.29	1.29	1.29
Rank	(f_1-f_7)	1	1	1

the three algorithms. We generated the result for MCRO, OCRO, HP-CRO2, and RCCRO4 by running each objective function 25 times.

We provided three experiment methods to evaluate the capability of comparison algorithms. These three methods were evaluated for optimal solution quality, convergence speed, and statistical hypothesis testing.

(1) *Optimal Solution Quality Evaluation.* For each objective function, the optimal solution or the best global minimum is the most important result in each runtime. In a computing time, each round of iteration contains the local minimum as a result. Consequently, if the current global minimum is worse than the local minimum, then the current global minimum is replaced by the local minimum. The best global minimum at each computing time is generated when the program meets the stop condition of MCRO and other optimization algorithms. The mean of the best global minimum at 25 computing times is the optimal solution of each function. If the optimal solutions of the competitors are equal, then the second key for comparison is the standard deviation. The following optimal solution quality evaluations are demonstrated.

TABLE 4: Comparisons of the result of the three solutions for Category II (f_8-f_{13}) (the best results are marked in bold).

Function	Result of	Algorithm with mutation		
		Polynomial	Uniform	Nonuniform
f_8	Mean	-1.24E + 04	-6.26E + 03	-1.24E + 04
	StDev.	6.87E + 01	2.94E + 02	5.77E + 01
	Rank	2	3	1
f_9	Mean	0	0	0
	StDev.	0	0	0
	Rank	1	1	1
f_{10}	Mean	4.44E - 16	4.44E - 16	4.44E - 16
	StDev.	0	0	0
	Rank	1	1	1
f_{11}	Mean	1	1	1
	StDev.	0	0	0
	Rank	1	1	1
f_{12}	Mean	2.51E - 06	4.61E - 03	3.22E - 06
	StDev.	1.55E - 06	1.22E - 03	3.61E - 06
	Rank	1	3	2
f_{13}	Mean	3.03E - 06	1.08E - 04	3.34E - 06
	StDev.	2.15E - 06	6.33E - 05	2.28E - 06
	Rank	1	3	2
Average rank	(f_8-f_{13})	1.17	2.00	1.33
Rank	(f_8-f_{13})	1	3	2

Table 6 presents the optimal solution quality evaluation of MCRO, OCRO, HP-CRO2, and RCCRO4 for high-dimensional unimodal functions or Category I (f_1-f_7). MCRO performs best for f_1, f_2, f_3, f_4, f_6 , and f_7 but not for f_5 . MCRO and OCRO are the best for f_1, f_2, f_3 , and f_6 ; HP-CRO2 and RCCRO4 derive the same results as MCRO and OCRO for f_6 . RCCRO4 generates the best result for f_5 . The ranking of optimal solution quality for Category I functions in descending order is as follows: MCRO, OCRO, HP-CRO2, and RCCRO4.

Table 7 compares the optimal solution quality for Category II functions. MCRO performs best for f_9, f_{10}, f_{11} , and f_{12} ; OCRO generates the same results as MCRO for f_9, f_{10} , and f_{11} ; HP-CRO2 is the best for f_{13} ; RCCRO4 has the best result for f_8 . The ranking of optimal solution quality for Category II functions in descending order is the same as that for Category I.

Table 8 compares the results of Category III functions. MCRO is the best in this category because it generates the best results for 8/10 functions: $f_{14}, f_{16}, f_{17}, f_{18}, f_{19}, f_{21}, f_{22}$, and f_{23} . MCRO is followed by OCRO, which is the best for f_{15} . HP-CRO2, which generates the best result for f_{20} , is in third rank, followed by RCCRO4 in fourth rank. The ranking of optimal solution quality for Category III functions in descending order is the same as that for Categories I and II.

The overall ranking presented at the bottom of Table 8 shows that MCRO performs best in the optimal solution

TABLE 5: Comparisons of the result of the three solutions for Category III (f_{14} – f_{23}) (the best results are marked in bold).

Function	Result of	Algorithm with mutation		
		Polynomial	Uniform	Nonuniform
f_{14}	Mean	9.98E – 01	9.99E – 01	1.05E + 00
	StDev.	8.60E – 11	1.62E – 03	1.94E – 01
	Rank	1	2	3
f_{15}	Mean	3.92E – 04	3.79E – 04	3.42E – 04
	StDev.	3.11E – 05	2.34E – 05	1.64E – 05
	Rank	3	2	1
f_{16}	Mean	–1.03E + 00	–1.03E + 00	–1.03E + 00
	StDev.	4.51E – 03	4.81E – 03	4.68E – 03
	Rank	1	3	2
f_{17}	Mean	3.98E – 01	3.98E – 01	3.98E – 01
	StDev.	9.86E – 05	6.16E – 04	4.51E – 04
	Rank	1	3	2
f_{18}	Mean	3.00E + 00	3.00E + 00	3.00E + 00
	StDev.	1.14E – 04	1.75E – 03	1.53E – 03
	Rank	1	3	2
f_{19}	Mean	–3.86E + 00	–3.86E + 00	–3.86E + 00
	StDev.	2.42E – 03	3.95E – 03	4.82E – 03
	Rank	1	2	3
f_{20}	Mean	–3.31E + 00	–3.31E + 00	–3.26E + 00
	StDev.	5.33E – 03	4.67E – 03	2.55E – 02
	Rank	2	1	3
f_{21}	Mean	–1.01E + 01	–1.01E + 01	–1.01E + 01
	StDev.	3.06E – 02	1.56E – 02	1.06E – 01
	Rank	2	1	3
f_{22}	Mean	–1.04E + 01	–1.04E + 01	–1.03E + 01
	StDev.	3.14E – 02	1.39E – 02	7.40E – 02
	Rank	2	1	3
f_{23}	Mean	–1.05E + 01	–1.05E + 01	–1.05E + 01
	StDev.	1.32E – 02	1.28E – 02	6.52E – 02
	Rank	2	1	3
Average rank	(f_{14} – f_{23})	1.60	1.90	2.50
Rank	(f_{14} – f_{23})	1	2	3
Overall rank	(f_1 – f_{23})	1.39	1.74	1.83
Rank	(f_1 – f_{23})	1	2	3

quality evaluation. MCRO is followed successively by OCRO, HP-CRO2, and RCCRO4.

(2) *Convergence Speed Evaluation.* Convergence speed is an essential issue that indicates the performance of algorithms in optimization. In our experiment, the convergence speed of the algorithm was calculated by counting the number of iterations (FEs) before the algorithm converges with the accepted result. The accepted result is the results given there are the average FEs needed to reach the threshold expressed as acceptable solutions for each objective function such as acceptable solution of f_1 is $1E - 60$, the robust algorithm has

to generate the result at least not worse than accepted result. The algorithm with fewer FEs is more powerful than that with greater FEs. Table 9 compares the convergence speeds of the algorithms. MCRO demonstrates the fastest convergence speed for all functions, except for f_8 , where HP-CRO2 is the best, and for f_{19} and f_{21} , where OCRO is the best. In addition, the distance between MCRO and the second best of each function is significant. For $f_1, f_3, f_4, f_{10}, f_{11}$, and f_{14} , MCRO converges the accepted results with 691, 819, 338, 896, 1056, and 314 FEs, respectively, whereas OCRO, respectively, has 23253, 25053, 31245, 65200, 64947, and 2818 FEs for the same functions. For f_{22} , MCRO and RCCRO4 undergo 28 and 4721 FEs, respectively. The convergence speed ranking of comparison algorithms for all objective functions from fastest to slowest is as follows: MCRO, OCRO, HP-CRO2, and RCCRO4.

After evaluating the convergence speed on the basis of iteration number, we evaluated the same parameter in MCRO, OCRO, HP-CRO2, and RCCRO4 by drawing a convergence curve for specific functions ($f_4, f_7, f_9, f_{10}, f_{11}$, and f_{23}) in a particular run. MCRO remains the most advantageous among the four algorithms.

(3) *Statistical Hypothesis Testing.* Friedman test [26], a well-known nonparametric statistical test, was used to assess the statistical hypothesis. Friedman test evaluates the differences or equality of results among competitor algorithms. We calculated the Friedman statistic according to χ^2 distribution with $k - 1$ degrees of freedom. Then, we estimated the p value according to normal approximations. The Friedman statistic F_f is presented as Formula (6). Parameter n is the number of objective functions, which is 23 in our test. k represents the number of comparison algorithms. In this case, we compared our proposed algorithm MCRO with three previous algorithms (OCRO, HP-CRO2, and RCCRO4); hence $k = 4$. Friedman rank or R_j is searching by transforming the quality results of MCRO, OCRO, HP-CRO2, and RCCRO4 to ranks for each objective function (i). In cases of equal ranks, the average ranks are assigned:

$$F_f = \frac{12n}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right]. \quad (6)$$

Table 10 demonstrates the results of Friedman rank test in terms of the comparison of optimal solution quality and convergence speed. MCRO achieves the best rank in the statistic test comparing both the optimal solution quality and the convergence speed on the basis of the corresponding Friedman ranks 1.59 and 1.22, the statistic F_f values 21.76398 and 35.96694, and the p values $7.30387E - 05$ and $7.60987E - 08$. In addition, the results explicitly display significant differences across the competing algorithms. In multiple comparisons, the post hoc test was utilized to compare a set of algorithms with a control algorithm (best algorithm). The post hoc test starts with finding the test statistic z of the i th

TABLE 6: Optimal solution quality for Category I (f_1-f_7) (the best results are marked in bold).

Function	FEs	Result of	MCRO	OCRO	HP-CRO2	RCCRO4
f_1	150000	Mean	0	0	$2.01E-53$	$2.20E-07$
		StDev.	0	0	$9.83E-53$	$3.83E-08$
		Rank	1	1	3	4
f_2	150000	Mean	0	0	$7.14E-06$	$2.076E-03$
		StDev.	0	0	$2.81E-05$	$3.206E-04$
		Rank	1	1	3	4
f_3	250000	Mean	0	0	$1.55E-102$	$2.41E-07$
		StDev.	0	0	$5.91E-102$	$6.01E-08$
		Rank	1	1	3	4
f_4	150000	Mean	0	$5.73E-21$	$4.43E-04$	$8.521E-03$
		StDev.	0	$1.83E-20$	$2.56E-04$	$2.348E-03$
		Rank	1	2	3	4
f_5	150000	Mean	$2.85E+01$	$2.75E+01$	$2.22E+01$	$2.61E+01$
		StDev.	$1.05E-02$	$1.60E-01$	$3.50E+01$	$5.61E+01$
		Rank	4	3	1	2
f_6	150000	Mean	0	0	0	0
		StDev.	0	0	0	0
		Rank	1	1	1	1
f_7	150000	Mean	$2.57E-04$	$2.08E-03$	$8.18E-03$	$3.82E-03$
		StDev.	$2.07E-04$	$9.96E-04$	$2.87E-03$	$1.39E-03$
		Rank	1	2	4	3
Average	Rank	(f_1-f_7)	1.43	1.57	2.5771	3.14
Rank		(f_1-f_7)	1	2	3	4

TABLE 7: Optimal solution quality for Category II (f_8-f_{13}) (the best results are marked in bold).

Function	FEs	Result of	MCRO	OCRO	HP-CRO2	RCCRO4
f_8	150000	Mean	$1.24E+04$	$-1.00E+04$	$1.24E+04$	$-1.26E+04$
		StDev.	$6.87E+01$	$1.61E+02$	$1.61E+02$	$7.86E+01$
		Rank	3	4	2	1
f_9	250000	Mean	0	0	$6.20E-04$	$1.57E-03$
		StDev.	0	0	$1.31E-03$	$4.18E-04$
		Rank	1	1	3	4
f_{10}	150000	Mean	$4.44E-16$	$4.44E-16$	$6.88E-12$	$2.18E-03$
		StDev.	0	0	$2.33E-11$	$2.60E-04$
		Rank	1	1	3	4
f_{11}	150000	Mean	1	1	$1.00E+00$	$1.00E+00$
		StDev.	0	0	$1.04E-14$	$1.393E-07$
		Rank	1	1	3	4
f_{12}	150000	Mean	$2.51E-06$	$9.80E-04$	$1.08E-01$	$6.223E-02$
		StDev.	$1.55E-06$	$1.75E-04$	$2.09E-01$	$1.312E-01$
		Rank	1	3	4	2
f_{13}	150000	Mean	$3.03E-06$	$1.12E-02$	$1.04E-14$	$6.46E-06$
		StDev.	$2.15E-06$	$2.76E-03$	$1.59E-14$	$4.56E-06$
		Rank	2	4	1	3
Average	Rank	(f_8-f_{13})	1.50	2.33	2.67	3.00
Rank		(f_8-f_{13})	1	2	3	4

TABLE 8: Optimal solution quality for Category II ($f_{14}-f_{23}$) (the best results are marked in bold).

Function	FES	Result of	MCRO	OCRO	HP-CRO2	RCCRO4
f_{14}	7500	Mean	9.98E - 01	9.98E - 01	9.98E - 01	9.98E - 01
		StDev.	8.60E - 11	8.71E - 10	6.46E - 04	2.62E - 07
		Rank	1	2	4	3
f_{15}	250000	Mean	3.92E - 04	3.80E - 04	2.05E - 03	5.27E - 04
		StDev.	3.11E - 05	6.12E - 05	5.62E - 03	9.87E - 05
		Rank	2	1	4	3
f_{16}	1250	Mean	-1.03E + 00	-1.01E + 00	-1.01E + 00	-9.67E - 01
		StDev.	4.51E - 03	1.89E - 02	2.19E - 02	9.50E - 02
		Rank	1	2	3	4
f_{17}	5000	Mean	3.98E - 01	3.99E - 01	3.98E - 01	3.98E - 01
		StDev.	9.86E - 05	9.79E - 04	3.96E - 04	4.35E - 02
		Rank	1	4	2	3
f_{18}	10000	Mean	3.00E + 00	3.00E + 00	3.00E + 00	3.04E + 00
		StDev.	1.14E - 04	1.85E - 04	6.14E - 04	4.33E - 02
		Rank	1	2	3	4
f_{19}	4000	Mean	-3.86E + 00	-3.86E + 00	-3.86E + 00	-3.86E + 00
		StDev.	2.42E - 03	2.80E - 03	3.73E - 03	5.42E - 03
		Rank	1	2	3	4
f_{20}	7500	Mean	-3.31E + 00	-3.25E + 00	-3.32E + 00	-3.32E + 00
		StDev.	5.33E - 03	2.41E - 02	1.78E - 03	2.13E - 03
		Rank	3	4	1	2
f_{21}	10000	Mean	-1.01E + 01	-9.57E + 00	-9.50E + 00	-9.03E + 00
		StDev.	3.06E - 02	7.37E - 01	3.85E - 01	2.59E + 00
		Rank	1	2	3	4
f_{22}	10000	Mean	-1.04E + 01	-9.96E + 00	-9.75E + 00	-1.03E + 01
		StDev.	3.14E - 02	4.83E - 01	4.77E - 01	2.55E - 02
		Rank	1	3	4	2
f_{23}	10000	Mean	1.05E + 01	-1.00E + 01	-9.91E + 00	-9.562E + 00
		StDev.	1.32E - 02	6.05E - 01	5.31E - 01	2.213E + 00
		Rank	1	2	3	4
Average	Rank	($f_{14}-f_{23}$)	1.30	2.40	3.00	3.30
Rank		($f_{14}-f_{23}$)	1	2	3	4
Average rank		(f_1-f_{23})	1.39	2.13	2.78	3.17
Overall Rank		(f_1-f_{23})	1	2	3	4

and j th algorithms, which is processed using Formula (7). Consider

$$z = \frac{(R_i - R_j)}{\sqrt{k(k+1)/6}}, \quad (7)$$

where R_i and R_j are the Friedman ranks of the compared algorithms.

The unadjusted p value was calculated using the z value from the table of normal distribution $N(0, 1)$. The z value was then compared with an appropriate level of significance α . In the present study, the following three computing procedures were applied to obtain the adjusted p values (APVs): Bonferroni-Dunn procedure [27] presented as Formula (8), Holm procedure [28] assigned as Formula (9), and Hochberg procedure [29] expressed as Formula (10). Subscripts i and j

are indexes that each corresponds to a concrete hypothesis, and p_j is the p -value obtained for the j th hypothesis.

The Bonferroni-Dunn procedure adjusts the value α through a single step:

$$\text{Bonferroni APV}_i: \min \{v, 1\}, \quad \text{where } v = (k-1) p_i. \quad (8)$$

The Holm procedure adjusts the value α in a step-down manner:

$$\text{Holm APV}_i: \min \{v, 1\}, \quad (9)$$

$$\text{where } v = \max \{(k-j) p_j : 1 \leq j \leq i\}.$$

The Hochberg procedure adjusts the value α in a step-up direction:

$$\text{Hochberg APV}_i: \min \{(k-j) p_j : (k-1) \geq j \geq i\}. \quad (10)$$

TABLE 9: Convergence speed (the best results are marked in bold).

Function	Accept	MCRO	OCRO	HP-CRO2	RCCRO	Function	Accept	MCRO	OCRO	HP-CRO2	RCCRO
f_1	1E - 60	691	23253	138711	x	f_{13}	1E - 02	29767	95685	51597	81992
	Rank	1	2	3	4		Rank	1	4	2	3
f_2	1E - 10	334	2620	105625	x	f_{14}	9.99E - 01	314	2818	2996	3354
	Rank	1	2	3	4		Rank	1	2	3	4
f_3	1E - 110	819	25053	237038	x	f_{15}	4E - 03	2075	44535	56843	90405
	Rank	1	2	3	4		Rank	1	2	3	4
f_4	1E - 04	338	31245	128103	x	f_{16}	-1.02	166	651	752	937
	Rank	1	2	3	4		Rank	1	2	3	4
f_5	5E + 01	1251	6732	23608	120316	f_{17}	3.99E - 01	1544	3849	3419	2935
	Rank	1	2	3	4		Rank	1	4	3	2
f_6	1E - 06	704	4695	29235	97733	f_{18}	3.0001	5239	8444	7555	x
	Rank	1	2	3	4		Rank	1	3	2	4
f_7	1E - 02	349	9471	107379	48223	f_{19}	-3.85	2324	2011	2331	2703
	Rank	1	2	4	3		Rank	2	1	3	4
f_8	-1E + 04	6741	53688	4306	11499	f_{20}	-3.30	2469	6233	5643	3662
	Rank	2	4	1	3		Rank	1	4	3	2
f_9	1E - 03	909	3928	103567	138790	f_{21}	-9	8232	6143	7278	6220
	Rank	1	2	3	4		Rank	4	1	3	2
f_{10}	1E - 13	895	65200	74078	x	f_{22}	-9	28	5419	7604	4721
	Rank	1	2	3	4		Rank	1	3	4	2
f_{11}	1	1056	64947	x	x	f_{23}	-9	3796	5974	7210	4745
	Rank	1	2	3	3		Rank	1	3	4	2
f_{12}	1E - 03	15711	75733	28990	89666						
	Rank	1	3	2	4						
Average	(Rank)			1.22		2.43			2.91		3.35
Rank				1		3			4		2

TABLE 10: Result of Friedman test.

Algorithm	Friedman ranks Optimal result quality	Friedman ranks Convergence
MCRO	1.59	1.22
OCRO	2.33	2.43
HP-CRO2	2.85	2.93
RCCRO4	3.24	3.41
Statistic F_f	21.76398	35.96694
p value	7.30387E - 05	7.60987E - 08

The results of the APVs for the Friedman test are shown in Table 11, where the control algorithm is the proposed algorithm MCRO. These results comprise the z value, unadjusted p values, and APVs that were extracted using the post hoc procedures and computed using the three procedures. We compared the results for both optimal solution quality and convergence speed. MCRO is superior to RCCRO4, HP-CRO2, and OCRO in all post hoc procedures considered.

(4) *Computation Time.* Normally, the numbers of computation times are often proportional to capability of computer. Unlike the other results such as optimal solution quality and

convergence speed, although the simulations perform on various personal computers, these results are very approximate. Hence our experiments disregard using the computation time as an evaluation method. However, the computation times appearing in Table 12 indicate that, in our simulated situation, the computer times for these four comparison algorithms are closed.

4.4. *Discussion.* Section 4.2 recommends that the best MCRO solution is the combination of CRO, turning operator, and polynomial mutation operator. The solution with polynomial mutation obtains the most appropriate results over uniform and nonuniform mutations.

The experimental results in Section 4.3 substantiate that MCRO is more powerful than OCRO, HP-CRO2, and RCCRO4. MCRO can generate the best result on most of the 23 objective functions (see Figure 1). We determined several issues, such as optimal solution quality, convergence speed, and statistical hypothesis testing, to evaluate algorithm performance. Table 6 shows that MCRO can generate the best optimal solution for all high-dimensional unimodal functions. We can obtain the optimal solution 0 (equal $f_{\min}$) and StdDev 0 for five functions (f_1 , f_2 , f_3 , f_4 , and f_6). MCRO achieves the best average rank in solving high-dimensional multimodal functions (Table 7) and performs the optimal

FIGURE 1

TABLE 11: Adjusted p values for the Friedman test (control algorithm: MCRO).

Compare method	Algorithm	z	Unadjusted p value	Bonferroni-Dunn	Holm	Hochberg
Result quality (StDev.)	RCCRO4	4.334195427	$1.46294E-05$	$4.38883E-05$	$4.38883E-05$	$4.38883E-05$
	HP-CRO2	3.309749235	0.000933796	0.002801388	0.001867592	0.001867592
	OCRO	1.943820979	0.051917043	0.155751129	0.051917043	0.051917043
Convergence	RCCRO4	5.752659385	$8.78504E-09$	$2.63551E-08$	$2.63551E-08$	$2.63551E-08$
	HP-CRO2	4.491802534	$7.06229E-06$	$2.11869E-05$	$1.41246E-05$	0.000014125
	OCRO	3.178409980	0.001480852	0.004442556	0.001480852	0.001480852

TABLE 12: Computation times.

Function	Time to reach the best global optimum (sec.)			
	MCRO	OCRO	HP-CRO2	RCCRO4
f_1	0.0668	0.0050	0.0052	0.0063
f_2	0.0069	0.0066	0.0056	0.0068
f_3	0.0202	0.0073	0.0086	0.0095
f_4	0.0070	0.0076	0.0065	0.0123
f_5	0.0244	0.0102	0.0112	0.0165
f_6	0.0105	0.0098	0.0149	0.0088
f_7	0.0123	0.0274	0.0074	0.0553
f_8	0.0133	0.0528	0.0621	0.0433
f_9	0.0263	0.0324	0.0494	0.0335
f_{10}	0.0162	0.0195	0.0295	0.0443
f_{11}	0.0231	0.472	0.0547	0.0335
f_{12}	0.0238	0.0275	0.0288	0.0553
f_{13}	0.0255	0.0287	0.0301	0.0295
f_{14}	0.0007	0.0009	0.0009	0.0308
f_{15}	0.0165	0.0151	0.0156	0.0152
f_{16}	<0.0001	0.0005	0.0005	0.0005
f_{17}	<0.0001	0.0005	0.0009	0.0005
f_{18}	0.0002	0.0007	0.0007	0.0007
f_{19}	0.0001	0.0010	0.0010	0.0009
f_{20}	0.0005	0.0012	0.0011	0.0013
f_{21}	0.0005	0.0005	0.0005	0.0006
f_{22}	0.0006	0.0008	0.0008	0.0006
f_{23}	0.0008	0.0008	0.0009	0.0007

solution 0 (equal $f_{\min}$) and StdDev 0 for two functions (f_9 and f_{11}). For low-dimensional functions (Table 8), the results of MCRO are close to the true optimal, indicating that this algorithm is outstanding and consistent. Moreover, some comparison algorithms can generate the same results as MCRO in solving several objective functions, but MCRO converges faster than the other algorithms (Table 10).

5. Conclusions and Future Work

This paper presents a hybrid optimization algorithm MCRO that solves global numerical optimization problems by combining CRO with turning and mutation operators. The most excellent MCRO framework is the utilization of the solution with polynomial mutation. Therefore, we provide polynomial

mutation operator into our proposed approach. Our algorithm is advantageous because the nature of CRO is suitable for global numerical optimization, and turning operator is a perfect method to improve optimal solution quality and algorithm reliability. Furthermore, polynomial mutation operator accelerates algorithm convergence. To analyze the evaluation results, we compared the results of MCRO and other previously published algorithms (OCRO, HP-CRO2, and RCCRO4) in terms of optimal solution quality, convergence speed, and statistical hypothesis testing. These results indicate that MCRO is superior to other algorithms in solving global numerical optimization problems. Several potential parameters such as *Popsize*, *InitialKE*, *buffer*, *KELossRate*, *On-wallColl*, *DecThres*, *SynThre*, *StepSize*, *probability*, and *DistributionIndex* were also discussed.

MCRO is useful in solving global numerical optimization problems with high-dimensional unimodal, high-dimensional multimodal, and low-dimensional functions. However, whether or not MCRO will also be excellent for solving other problem types is unclear. Hence, the applicability of MCRO to other problem types requires further research and consideration.

Our future work will focus on investigating the applicability of MCRO to other numerical optimization problems and practical engineering optimization problems. We also intend to develop a new hybrid metaheuristic approach to solve optimization problems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Grant nos. 61173107 and 91320103), the National High Technology Research and Development Program of China (Grant no. 2012AA01A301-01), the Special Project on the Integration of Industry, Education and Research of Guangdong Province, China (Grant no. 2011A091000027), and the Project on the Integration of Industry, Education and Research of Guangdong Province, China.

References

- [1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman, Boston, Mass, USA, 1989.
- [2] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, MIT Press, Cambridge, Mass, USA, 1992.
- [3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, December 1995.
- [4] R. Dawkins, *The Selfish Gene*, Oxford University Press, Oxford, UK, 1976.
- [5] P. Merz and B. Freosleben, "A genetic local search approach to the quadratic assignment problem," in *Proceedings of the 7th International Conference on Genetic Algorithms*, pp. 465–472, San Diego, Calif, USA, 1997.
- [6] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep. TR06, Computer Engineering Department, Engineering Faculty, Erciyes University, 2005.
- [7] M. Dorigo, V. Maniezzo, and A. Coloni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.
- [8] W.-F. Gao, S.-Y. Liu, and L.-L. Huang, "A novel artificial bee colony algorithm based on modified search equation and orthogonal learning," *IEEE Transactions on Cybernetics*, vol. 43, no. 3, pp. 1011–1024, 2013.
- [9] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.
- [10] A. Y. S. Lam and V. O. K. Li, "Chemical-reaction-inspired meta-heuristic for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 381–399, 2010.
- [11] A. Y. S. Lam and V. O. K. Li, "Chemical reaction optimization: a tutorial," *Memetic Computing*, vol. 4, no. 1, pp. 3–17, 2012.
- [12] A. Y. S. Lam, V. O. K. Li, and J. J. Q. Yu, "Real-coded chemical reaction optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 3, pp. 339–353, 2012.
- [13] A. Y. S. Lam, J. Xu, and V. O. K. Li, "Chemical reaction optimization for population transition in peer-to-peer live streaming," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '10)*, pp. 1–8, IEEE, Barcelona, Spain, July 2010.
- [14] J. Xu, A. Y. S. Lam, and V. O. K. Li, "Chemical reaction optimization for the grid scheduling problem," in *Proceedings of the IEEE International Conference on Communications (ICC '10)*, pp. 1–5, Cape Town, South Africa, May 2010.
- [15] J. Xu, A. Y. Lam, and V. O. Li, "Parallel chemical reaction optimization for the quadratic assignment problem," in *Proceedings of the International Conference on Genetic and Evolutionary Methods (GEM '10)*, pp. 125–131, Las Vegas, Nev, USA, July 2010.
- [16] J. Xu, A. Y. S. Lam, and V. O. K. Li, "Chemical reaction optimization for task scheduling in grid computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 10, pp. 1624–1631, 2011.
- [17] J. J. Q. Yu, A. Y. S. Lam, and V. O. K. Li, "Evolutionary artificial neural network based on chemical reaction optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '11)*, pp. 2083–2090, IEEE, New Orleans, La, USA, June 2011.
- [18] T. T. Nguyen, Z. Y. Li, S. W. Zhang, and T. K. Truong, "A Hybrid algorithm based on particle swarm and chemical reaction optimization," *Expert Systems with Applications*, vol. 41, no. 5, pp. 2134–2143, 2014.
- [19] Z. Y. Li, Z. Li, T. T. Nguyen, and S. M. Chen, "Orthogonal chemical reaction optimization algorithm for global numerical optimization problems," *Expert Systems with Applications*, vol. 42, no. 6, pp. 3242–3252, 2015.
- [20] Wikipedia, "Mathematical optimization," http://en.wikipedia.org/wiki/Numerical_optimization.
- [21] E. Guggenheim, *Thermodynamics: An Advanced Treatment for Chemists and Physicists*, Wiley, North Holland, 5th edition, 1967.
- [22] X. Zhao, X.-S. Gao, and Z.-C. Hu, "Evolutionary programming based on non-uniform mutation," *Applied Mathematics and Computation*, vol. 192, no. 1, pp. 1–11, 2007.
- [23] M. Hamdan, "On the disruption-level of polynomial mutation for evolutionary multi-objective optimisation algorithms," *Computing and Informatics*, vol. 29, no. 5, pp. 783–800, 2010.
- [24] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
- [25] S. Cathabard, P. K. Lehre, and X. Yao, "Non-uniform mutation rates for problems with unknown solution lengths," in *Proceedings of the 11th Workshop Proceedings on Foundations of Genetic Algorithms (FOGA '11)*, pp. 173–180, ACM, Schwarzenberg, Austria, January 2011.
- [26] J. Derrac, S. Garcia, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–19, 2011.
- [27] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, 1961.
- [28] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979.
- [29] Y. Hochberg, "A sharper Bonferroni procedure for multiple tests of significance," *Biometrika*, vol. 75, no. 4, pp. 800–802, 1988.