Research Article

An Advanced Chemical Reaction Optimization Algorithm Based on Balanced Local and Global Search

Min Zhang , Liang Chen, and Xin Chen

College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

Correspondence should be addressed to Min Zhang; zhangmin@nuaa.edu.cn

Received 21 September 2017; Accepted 21 January 2018; Published 16 April 2018

Academic Editor: Antonino Laudani

Copyright © 2018 Min Zhang 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.

An advanced chemical reaction optimization algorithm based on balanced local search and global search is proposed, which combines the advantages of adaptive chemical reaction optimization (ACRO) and particle swarm optimization (PSO), to solve continuous optimization problems. This new optimization is mainly based on the framework of ACRO, with PSO's global search operator applied as part of ACRO's neighborhood search operator. Moreover, a "finish" operator is added to the ACRO's structure and the search operator is evolved by an adaptive scheme. The simulation results tested on a set of twenty-three benchmark functions, and a comparison was made with the results of a newly proposed hybrid algorithm based on chemical reaction optimization (CRO) and particle swarm optimization (denoted as HP-CRO). The final comparison results show the superior performance improvement over HP-CRO in most experiments.

1. Introduction

We often encounter optimization problems in scientific and technological research and development. Over the past decades, a series of optimization algorithms have been proposed: genetic algorithm (GA) (see, e.g., [1]), simulated annealing (SA) (see, e.g., [2]), ant colony optimization (ACO) (see, e.g., [3]), particle swarm optimization (PSO) (see, e.g., [4]), chemical reaction optimization (CRO) (see, e.g., [5]), and others. The optimization algorithms mentioned above are with the same objective, that is, to find a best (or optimal) solution.

In general, when faced with an optimization problem, we can always simplify it as follows: a solution s and a solution space S , an objective function f , and a set of constraints (assume there are m constraints), $c_1(x), c_2(x), \dots, c_m(x)$, which confine the search region, respectively. A particular solution s is usually represented by a vector of variables $x = \{x_1, x_2, \dots, x_n\}$, where n corresponds to the problem dimensions.

The optimality of x is evaluated by an objective function f and its output value y , that is, $y = f(x)$. Our objective can be to either maximize or minimize f . In this paper we assume the latter. Then our goal is to find the minimum solution

$s^* \in S$ where $f(s^*) \leq f(s), \forall s \in S$. Then a minimization problem can be described as follows:

$$\begin{aligned} & \min_{x \in R^n} \\ & \text{subject to } c_i(x) = 0, \quad i \in E \\ & \quad c_i(x) \leq 0, \quad i \in I, \end{aligned} \quad (1)$$

where R , E , and I represent the real number set, the index set for equalities, and the index set for inequalities, respectively. The No-Free-Lunch theorem states that all metaheuristics which search for extreme are exactly the same in performance when averaged over all possible objective function (see, e.g., [6]). In other words, when one works excellent in a certain class of problems, it will be outperformed by the others in other classes.

In recent years, CRO has been proposed and attracted an increasing interest from the community of optimization algorithms, a variety of improved algorithms based on CRO has been suggested (see, e.g., [7–10]) and it has been applied in many fields (see, e.g., [11–15]). Of all these algorithms, adaptive chemical reaction optimization (ACRO) stands out and shows its strong superiority. PSO has been applied in variety of fields and shows its higher convergence speed.

However, it usually converges to local minimum quickly and loses the opportunity to find a better one.

According to the introduction mentioned above, ACRO seems to be a well-performed optimization algorithm. However, similar to CRO, ACRO is still lacking in convergence speed. In order to avoid weaknesses of ACRO and PSO, we proposed a new algorithm which combines the advantages of both (denoted as AACRO).

The rest of the paper is organized as follows. Section briefly outlines the related works in this paper and gives the inspiration of our proposed algorithm. We explain the modifications of ACRO and introduce the basic framework of AACRO in Section. In Section, we propose the proof on convergence and provide the convergence speed analysis. In Section, we describe 23 benchmark problems. In Section, we present the simulations results and compare the results of AACRO and HP-CRO. In particular, Section presents the experimental environment, parameter settings are shown in Section, and Section gives the comparison results. We conclude this paper and give some potential future works in Section.

A good optimization algorithm must have a good global search performance as well as a good local search performance. However, global search and local search performance are always confined to each other in practice. For example, if an optimization algorithm is good at global search then it must be poor at local search and vice versa. In order to achieve the best performance, the two abilities should be well balanced.

2. Related Works

2.1. The Adaptive Chemical Reaction Algorithm. As an improvement of CRO, ACRO is proposed by Yu et al. in 2015 (see, e.g., [9]) and substantially inherits the standard CRO's structure. ACRO reduces the number of parameters defined in canonical from eight (initial population size (*inipopsiz*), initial molecular kinetic energy (*iniKE*), initial central energy buffer (*iniBuffer*), molecular collision rate (*CollRate*), energy loss rate (*LossRate*), decomposition occurrence threshold (*DecThres*), synthesis occurrence threshold (*SynThres*), and perturbation step size (*StepSize*)) to three classes (energy-related, reaction-related, and real-coded-related) and makes the occurrence of elementary reactions adaptive.

The energy-related class includes *iniKE*, *iniBuffer*, and *LossRate*. The modifications of the parameters are as follows:

$$iniKE = (PE_{\omega_l} - PE_{\omega_s}) \times iniPopSize,$$

$$LossRate \sim N(0, r^2), \quad (2)$$

$$iniBuffer = 0,$$

where PE_{ω_l} and PE_{ω_s} are molecules with the largest and the smallest *PE* and the value for *LossRate* of each molecule is approximated by a modified folded normal distribution, which is generated from a normal distribution with a mean value of 0 and standard deviation of r .

The reaction-related class includes *CollRate*, *SynThres*, and *DecThres*. A new parameter *ChangeRate* has been

introduced to replace the original parameters *SynThres* and *DecThres* to control the frequency of decompositions and syntheses. And in order to control the number of molecules, f_{pop} , f_{dec} , and f_{syn} have been introduced to the system; then the population feedback term is modified as follows:

$$\begin{aligned} f_{pop} &= \frac{curPopSize - iniPopSize}{iniPopSize}, \\ f_{dec} &= \frac{1}{2} \times (1 - f_{pop}), \\ f_{syn} &= 1 - f_{dec} = \frac{1}{2} \times (1 + f_{pop}), \end{aligned} \quad (3)$$

where *curPopSize* is the current population size and *iniPopSize* is the initial population size. f_{dec} and f_{syn} determine that the probability the current iteration is a decomposition and a synthesis, respectively.

The real-code-related class includes *StepSize*. In order to solve the continuous optimization problems, the parameter *StepSize* needs to keep changing with iterations. The modification in *StepSize* is twofold.

(1) Initial Value of StepSize

$$StepSize_{init,i} = \frac{(Bound_{upper,i} - Bound_{lower,i})}{2}, \quad (4)$$

where $StepSize_{init,i}$ is the initial *StepSize* for the *i*th element in the solution and $Bound_{upper,i}$ and $Bound_{lower,i}$ are the upper and lower bounds for the *i*th element, respectively.

(2) Evolution of StepSize. The strategy "1/5 success rule" (see, e.g., [16]) has been adopted to modify *StepSize* in the course of searching. This rule was originally stated as follows.

After every n mutations, check how many successes have occurred over the preceding $10n$ mutations. If the number is less than $2n$, multiply the step lengths by the factor 0.85; divide them by 0.85 if more than $2n$ successes occurred.

Besides these parameter modifications, the framework of ACRO is similar to the canonical CRO. ACRO also satisfies two thermodynamics laws and four basic reactions (i.e., on-wall ineffective collision, decomposition, intermolecular ineffective collision, and synthesis).

2.1.1. On-Wall Ineffective Collision Operator. An on-wall ineffective collision represents the situation when a molecule collides with the container (i.e., $\omega \rightarrow \omega'$). This can be done by picking ω' in the neighborhood of ω , so it only leads to a small change of the previous molecule's structure. With the increasement of this elementary reaction, more and more *KE* will be transferred to the buffer and offers the energy needed in decomposition. This process can be regarded as a kind of local search of the solution space.

2.1.2. Intermolecular Ineffective Collision Operator. Intermolecular ineffective collision takes place when molecules collide with each other and then bounce away. The molecules (assume two) (i.e., $\omega_1 + \omega_2 \rightarrow \omega'_1 + \omega'_2$) remain unchanged before and after the process. This elementary reaction is very

```

(1) Input: Objective function  $f$ , constraints
(2) \\Initialization
(3) Set  $iniPopSize$ ,  $iniKE$ ,  $buffer$ ,  $CollRate$ ,  $iter$ ,
     $ChangeRate$  and  $StepSize$ 
(4) Create  $PopSize$  number of molecules
(5) \\Iterations
(6) while (the stopping criteria not met) do
(7)   if  $iter \% n = 0$  and  $iter/n > 10$ 
(8)     if  $totalsuccess (iter - 10n) \geq 2n$ 
(9)        $StepSize = StepSize/0.85$ 
(10)    else
(11)       $StepSize = StepSize \times 0.85$ 
(12)    end if
(13)  end if
(14) Generate  $b \in [0, 1]$ 
(15)   if  $b > CollRate$  then
(16)     Randomly select one molecule
(17)   if Decomposition criterion met then
(18)     Trigger Decomposition
(19)   else
(20)     Trigger On-wall Ineffective Collision
(21)   end if
(22) else
(23)   Randomly select two molecules
(24)   if Synthesis criterion met then
(25)     Trigger Synthesis
(26)   else
(27)     Trigger Inter-molecular Ineffective Collision
(28)   end if
(29) end if
(30) Check for any new minimum solution
(31) end while
(32) \\The final stage
(33) Output the best solution found and its objective function value

```

ALGORITHM 1: ACRO algorithm.

similar to the on-wall ineffective collision operator but no external buffer is involved. The new molecules ω'_1 and ω'_2 are generated from their own neighborhoods. This process can also be regarded as a kind of local research.

2.1.3. Decomposition Operator. Decomposition refers to the situation when a molecule hits the wall and breaks into several parts (for simplicity, we only consider two parts, i.e., $\omega \rightarrow \omega'_1 + \omega'_2$). The idea of decomposition is to allow the system to explore other regions of the solution space after enough local search by the ineffective collisions, so this process can be regarded as a local search over a wider range.

2.1.4. Synthesis Operator. A synthesis happens when multiple (assume two) molecules collide with each other and then fuse together (i.e., $\omega_1 + \omega_2 \rightarrow \omega'$). The resulting molecules have a higher “ability” to explore a new solution region. In other words, the resultant molecule may appear in a region farther away from the existing ones in the solution space. The idea behind synthesis is a kind of global research.

Based on the framework introduced above, we can formulate ACRO algorithm as Algorithm 1.

2.2. Particle Swarm Optimization Algorithm. Similar to CRO, the PSO searches the solution space by using a series of particles, which is randomly distributed in initial search space D . Each particle has its own attributes (i.e., i th particle has three attributes, including its position vector $x_i = (x_{i1}, \dots, x_{id}, \dots, x_{iD})$, velocity $v_i = (v_{i1}, \dots, v_{id}, \dots, v_{iD})$, and its best location found so far $p_i = (p_{i1}, \dots, p_{id}, \dots, p_{iD})$), and the position of each particle is determined by its own flight experience as well as the swarm’s optimal position. Based on the rules of PSO, the update process from iteration k to $k + 1$ becomes the following:

$$\begin{aligned}
 v_i^{k+1} &= wv_i^k + c_1\zeta_1^k (p_i^k - x_i^k) + c_2\zeta_2^k (g_s^k - x_i^k), \\
 x_i^{k+1} &= x_i^k + v_i^{k+1},
 \end{aligned} \tag{5}$$

where v_i^k denotes the velocity of particle i in iteration k ; x_i^k denotes the position of particle i in iteration k ; p_i^k is the optimal solution found by particle i ; g_s^k is the global optimal solution; w is the inertia weight; c_1 is the cognitive weight and c_2 is the social weight; ζ_1^k and ζ_2^k are random numbers uniformly distributed in $[0, 1]$.

2.3. Inspiration of Advanced Adaptive Chemical Reaction Optimization Algorithm. PSO is famous with its high convergence speed. However, high convergence speed ability may lead to inadequate local search and a high probability of falling into local optimum. Moreover, CRO was proposed as a new algorithm; it demonstrates strong local search ability. As an advanced algorithm based on CRO, ACRO simplifies the CRO's structure in a wide range and makes the *StepSize* adaptive, which further improves CRO's performance.

CRO's strong local search performance and PSO's excellent global search performance make the combination of the two algorithms an inevitable trend. The algorithm HP-CRO (see, e.g., [17]) proposed by Nguyen et al. combines both algorithms and achieves a good result.

However, HP-CRO just replaces CRO's decomposition and synthesis operators with PSO's search operator, so it has the same global optimization operator as PSO, which brings about that the accuracy of the optimization depends largely on the parameter settings of the PSO algorithm. In other words, if there exists an incorrect parameter setting, the optimization result may be poor, which greatly weakens the performance of the HP-CRO algorithm. Moreover, without an adaptive scheme, the fixed parameter settings in CRO algorithm greatly limit the accuracy of algorithm optimization.

In order to overcome the shortcomings mentioned above, AACRO is proposed. In next section, the detailed algorithm for AACRO is designed.

3. An Advanced Chemical Reaction Optimization Algorithm Based on Balanced Local Search and Global Search

This section focuses on discussing the infrastructure and basic principles of the AACRO algorithm.

3.1. Basic Modifications. In order to make ACRO and PSO organically combined, we modify 2 parameters of ACRO and introduce a finish operator. The details are as follows.

3.1.1. Changing ACRO's Neighborhood Operator by Adding PSO's Search Operator (p_i^k and g_s^k). As we can see in Section, p_i^k and g_s^k refer to i th particle's optimal solution and the global best solution. PSO's high convergence speed is closely related to these two parameters. Add these to ACRO's search operator will also lead to a high convergence speed. However, if we simply add them together, then the ACRO's search operator will be the same as the PSO's, which will lead to a premature convergence. To solve this, we define a new parameter w_global to control whether it is a global search or a local search. Then the ACRO's neighborhood operator becomes the following:

$$N(\omega) = s_{\Delta} + r_1 \times (c_1 \zeta_1 (PBest - \omega) + c_2 \zeta_2 (GBest - \omega)), \quad (6)$$

where the value of r_1 is mainly dependent on a randomly generated number r_2 distributed in $[0, 1]$. If the value of r_2 is larger than w_global , we set $r_1 = 1$, else $r_1 = 0$.

$s_{\Delta} \sim N(0, StepSize)$ is also a random number generated by Gaussian distribution. $N(\omega)$ is the neighborhood operator. Equation (6) combines global search operator and local search operator with each other; the value of parameter w_global can be set manually and change in schedule, which increases the flexibility of the algorithm.

3.1.2. Modifying Synthesis and Decomposition Criterion. With the change of the neighborhood search operator, AACRO will have a higher convergence speed, if synthesis operator conducts in a relatively fast frequency then the population diversity cannot be maintained. On the other hand, if there are too many decomposition reactions, the total amount of the molecules will shoot up, which turns the algorithm into an unordered search. So if molecule amount is less than one-half of the original molecule amount, the synthesis operator is suspended; the decomposition operator is prohibited if the number of molecules is more than doubled.

3.1.3. Introducing a "Finish" Operator. At the end of the iteration, we introduce a finish operator. If the molecules are more than two, we continuously choose two molecules from the existing molecular population randomly and conduct the synthesis operator. The optimal particle is obtained until the number of molecules is reduced to one. The difference is that the new molecule produced by the operator will be reserved if its *PE* is less than the global minima, otherwise give up. The finish operator is somewhat similar to the crossover operator of genetic algorithm, the only difference is that the finish operator combines Darwin's theory of evolution in the "survival of the fittest" thought (see, e.g., [18]) and just selects the final optimum solution. The steps of the finish operator can be described as shown in Algorithm 2.

3.2. The Framework of AACRO. After some structure changes, we establish the framework of AACRO. Similar to other optimization algorithms, there are also three stages in AACRO: initialization, iteration, and the final stage. Figure 1 shows its flow chart.

In the first stage, all parameters need to be initialized. In this step, the state space and some constraints are defined first; then we produce the molecule swarm by generating *PopSize* numbers of solutions randomly in the solution space; at last, the *PE* and *KE* of each molecule are initialized.

In the iteration stage, a number of iterations are performed. In each iteration, we first determine whether a unimolecular collision or an intermolecular collision would happen by randomly generating a number r in the interval $[0, 1]$ and comparing with *CollRate*. If r is larger than *CollRate*, it will result in an intermolecular collision and two molecules are randomly chosen. If both of them satisfy the synthesis criterion ($KE \leq \beta$), they combine through synthesis, or an intermolecular ineffective collision takes place. Otherwise, unimolecular collision is triggered and one molecule is randomly chosen. If it satisfies the decomposition criterion ($NumHit - MinHit > \alpha$, where α is the duration tolerance without obtaining any new local minimum solution.), the molecule will experience a decomposition, else an on-wall ineffective collision will take place.

```

(1) Get current PopSize
(2) while PopSize  $\geq$  2
(3)   choose two molecules randomly, conduct synthesis reaction
(4)   if  $PE_{w'} \leq MinPE$  then
(5)     popsiz = popsiz - 1,  $MinPE = PE_{w'}$ 
(6)   end if
(7)   update molecules, update PopSize
(8) end while
(9) Output optimal solution

```

ALGORITHM 2: The finish operator.

FIGURE 1: Flow chart of AACRO.

In the final stage, a finish operator is triggered. The existing molecules will continuously take a synthesis reaction until the numbers of the molecules reduce to one. After each synthesis reaction, the new molecule is updated and the old ones are abandoned.

We provide the source code (see, e.g., [19]), and the pseudocode of the details of AACRO is also given in Algorithm 3.

3.3. The Differences between the CRO Versions. We can see from Section that the ACRO version is proposed as an improvement of canonical CRO version and in this section we continue to optimize the structure of ACRO, so the AACRO version is actually also an improvement of the canonical CRO version. Therefore, we conclude several modification experiences and analyze the differences between the canonical CRO, ACRO, and ACCRO version.


```

(1) Input: Objective function  $f$  and the parameter values
(2) \\Initialization
(3) Set  $PopSize$ ,  $w_{global}$ ,  $ChangeRate$ ,  $CollRate$ ,
     $LossRate$  and  $totaliters$ 
(4) Create  $Swarm$ ,  $PE$ ,  $KE$ ,  $StepSize$ ,  $success$  ( $iter$ ),  $n$ 
    and  $MinPE$ 
(5) \\Iterations
(6) while (the stopping criteria not met) do
(7)   if (step size change rule met) then
(8)     if ("1/5 success rule" met) then
(9)        $StepSize = StepSize \times 0.85$ 
(10)    else
(11)       $StepSize = StepSize/0.85$ 
(12)    end if
(13)  end if
(14)  Generate  $b \in [0, 1]$ 
(15)  if ( $b < CollRate$ ) then
(16)    Randomly select one molecule  $M_w$ 
(17)  if (Decomposition criterion met) then
(18)    Trigger Decomposition
(19)  else
(20)    Trigger On-wall Ineffective Collision
(21)  end if
(22)  else
(23)    Randomly select two molecules  $M_{w1}$  and  $M_{w2}$ 
(24)  if (synthesis criterion met) then
(25)    Trigger Synthesis
(26)  else
(27)    Trigger Inter-molecular Ineffective Collision
(28)  end if
(29)  end if
(30)  check for any new minimum solution
(31) end while
(32) Trigger Finish operator
(33) Obtain the global optimal
(34) Output the best solution found and its objective function value

```

ALGORITHM 3: Details of AACRO.

As an initial version, the canonical CRO version builds its basic framework and shows its relatively balanced global search and local search abilities. However, it adopts fixed step size and unreasonable collision criterions, which results in a low optimization efficiency. So the promoted version ACCRO comes out, it greatly streamlines the structure of the canonical CRO version and adopts several adaptive strategies, which makes the canonical CRO adaptive and speeds up its optimization efficiency to some extent.

The AACRO version inherits most of the adaptive strategies used in ACRO version and makes some further improvements. In order to solve the low efficiency at the beginning of the optimization process, we adopt PSO's global search operator as part of ACRO's neighborhood operator and use a new parameter to control whether it is a global search or a local search, which greatly enhances ACRO's optimization efficiency. However, high optimization efficiency may lead to a premature convergence. To prevent this, we modify ACRO's synthesis and decomposition criterions again and make some molecule amount restrictions to ensure a relatively high converge speed.

So we can conclude that the ACRO and AACRO versions are both with adaptive strategies; what is more, the AACRO version has a higher convergence speed and owns a better search ability.

4. Convergence Proof and Convergence Speed Analysis

Similar to CRO, the operation of the AACRO algorithm is a process of repetitive operation of on-wall ineffective collisions, intermolecular ineffective collisions, decomposition, and synthesis operators. Each iteration process is related only to the state of the current population. Therefore, the AACRO algorithm process can also be modeled as a Markov chain (see, e.g., [20]) and the convergence of AACRO can be proved by using the characteristics of the Markov chain. However, it is worth noting that the search domain (i.e., state space S) of continuous constraint problem is often infinite; that is, the algorithm cannot be proved directly by using Markov chain property.

To solve the problem above, we define a minimal step size s_Δ ; if the *StepSize* generated by the neighborhood operator is smaller than s_Δ , *StepSize* is equal to s_Δ . Furthermore, *StepSize* in the ACRO algorithm will slowly decrease with the iteration, which leads to gradually reducing the actual search domain. With this feature, we can simplify the infinite search domain into a finite state space; then the convergence can be proved.

4.1. Algorithm Convergence Proof. Before the proof process, we first provide some basic definitions, assumptions, and corresponding inferences.

Definition 1 (pseudomolecule). A pseudomolecule ω_ϕ is an imaginary molecule with no attributes (i.e., *PE* and *KE*).

The purpose of introducing the pseudomolecule is to keep the numbers of the molecules constant.

Definition 2 (state space). Given a problem A , a state of AACRO can be described as follows:

$$S_A = \omega_1^A \times \omega_2^A \times \cdots \times \omega_{n_{\max}}^A, \quad (7)$$

where ω_i^A denotes the state space of i th molecule and $n_{\max}$ is the maximum population size.

Definition 3 (the best-so-far solution). For a problem A , the best-so-far solution $x_t^{A,bsf}$ is the current optimal solution found up to current iteration t , where $t = 0, 1, 2, \dots$.

Definition 4 (absorbing Markov chain). Let Ω be the set of states, where the probability of transiting from one absorbing state to another is 1. Then a Markov chain $\{S_t\}_{t=0}^{+\infty}$ is absorbing if it satisfies

$$P\{S_{t+1} \notin \Omega \mid S_t \in \Omega\} = 0, \quad t = 0, 1, 2, \dots \quad (8)$$

Theorem 5. The optimizing process of AACRO on solving problem A can be modeled by a Markov chain $\{S_t\}_{t=0}^{+\infty}$.

Proof. We can see that the state space S_{t+1}^A at time $t + 1$ is only dependent on the state S_t^A at time t , namely, we have

$$P\{S_t^A \in \Omega^A \mid S_0^A, S_1^A, \dots, S_t^A\} = P\{S_{t+1}^A \in \Omega^A \mid S_t^A\}, \quad (9)$$

where $P\{\cdot \mid \cdot\}$ is the transition probability and Ω^A is the state space on problem A . Equation (10) is a Markov property, so $\{S_t\}_{t=0}^{+\infty}$ is actually a Markov chain with state space Ω^A . Besides, we can model AACRO as an absorbing Markov chain by an additional term $x_t^{A,bsf}$, the state space of iteration t is $S_t^A = x_t^{A,bsf} \times \omega_1^A \times \omega_2^A \times \cdots \times \omega_{n_{\max}}^A$, and then we have $P\{S_{t+1} \notin \Omega \mid S_t \in \Omega\} = 0$, $t = 0, 1, 2, \dots$, which makes AACRO an absorbing Markov chain. $\square$

Definition 6 (nonattenuation sequence). Let $\{\sigma_t\}_{t=0}^{+\infty}$ be a sequence, where $\forall t \geq 0$, $0 \leq \sigma_t \leq 1$. The sequence is nonattenuation if it satisfies $\prod_{t=0}^{+\infty} (1 - \sigma_t) = 0$.

Lemma 7. Given an absorbing Markov chain $\{S_t\}_{t=0}^{+\infty}$, there exists a nonattenuation sequence $\{\sigma_t\}_{t=0}^{+\infty}$ such that $P\{S_{t+1} \notin \Omega_{\text{opt}} \mid S_t \in \Omega_{\text{opt}}\} \geq \sigma_t$ for $t = 0, 1, 2, \dots$ if and only if the Markov chain is said to be convergent.

Proof. We first prove the sufficiency part. If for $t = 0, 1, 2, \dots$, $P\{S_{t+1} \notin \Omega_{\text{opt}} \mid S_t \in \Omega_{\text{opt}}\} \geq \sigma_t$, it implies $P\{S_{t+1} \notin \Omega_{\text{opt}} \mid S_t \notin \Omega_{\text{opt}}\} \leq 1 - \sigma_t$. Define

$$\tilde{P}(t) = \prod_{i=0}^t P\{S_{i+1} \notin \Omega_{\text{opt}} \mid S_i \notin \Omega_{\text{opt}}\}. \quad (10)$$

Then, we have

$$\begin{aligned} \lim_{t \rightarrow \infty} \tilde{P}(t) &= \prod_{i=0}^{\infty} P\{S_{i+1} \notin \Omega_{\text{opt}} \mid S_i \notin \Omega_{\text{opt}}\} \\ &\leq \prod_{i=0}^{\infty} (1 - \sigma_i) = 0. \end{aligned} \quad (11)$$

Obviously, $\tilde{P}(t) \geq 0$ for all t . From (11) we have $\tilde{P}(t) \leq 0$; thus $\lim_{t \rightarrow \infty} \tilde{P}(t) = 0$. From the property of the absorbing Markov chain, we have

$$P\{S_{t+1} \notin \Omega_{\text{opt}} \mid S_t \in \Omega_{\text{opt}}\} = 0. \quad (12)$$

Then

$$P\{S_{t+1} \in \Omega_{\text{opt}} \mid S_t \in \Omega_{\text{opt}}\} = 1. \quad (13)$$

Therefore

$$\begin{aligned} &P\{S_{t+1} \notin \Omega_{\text{opt}}\} \\ &= P\{S_{t+1} \notin \Omega_{\text{opt}} \mid S_t \in \Omega_{\text{opt}}\} P\{S_t \in \Omega_{\text{opt}}\} \\ &\quad + P\{S_{t+1} \notin \Omega_{\text{opt}} \mid S_t \notin \Omega_{\text{opt}}\} P\{S_t \notin \Omega_{\text{opt}}\} \\ &= P\{S_{t+1} \notin \Omega_{\text{opt}} \mid S_t \notin \Omega_{\text{opt}}\} P\{S_t \notin \Omega_{\text{opt}}\} \\ &= P\{S_0 \notin \Omega_{\text{opt}}\} \prod_{i=0}^t P\{S_{i+1} \notin \Omega_{\text{opt}} \mid S_i \notin \Omega_{\text{opt}}\} \\ &= P\{S_0 \notin \Omega_{\text{opt}}\} \tilde{P}(t). \end{aligned} \quad (14)$$

As $\lim_{t \rightarrow \infty} \tilde{P}(t) = 0$, we have $\lim_{t \rightarrow \infty} P\{S_t \notin \Omega_{\text{opt}}\} = 0$. Then

$$\lim_{t \rightarrow \infty} P\{S_t \in \Omega_{\text{opt}}\} = 1 - \lim_{t \rightarrow \infty} P\{S_t \notin \Omega_{\text{opt}}\} = 1. \quad (15)$$

Therefore, the algorithm will reach the optimal state with probability 1 if iteration time tends to infinity. $\square$

Necessity Part. If a Markov chain is convergent, we can see from Definition that if time tends to infinity, the probability that the state space S_t converges to the optimal space Ω_{opt} is 1, that is, $\lim_{t \rightarrow \infty} P\{S_t \in \Omega_{\text{opt}}\} = 1$, which is equivalent to

$$\lim_{t \rightarrow \infty} P\{S_t \notin \Omega_{\text{opt}}\} = 1 - \lim_{t \rightarrow \infty} P\{S_t \in \Omega_{\text{opt}}\} = 0. \quad (16)$$

According to (11), we have

$$\begin{aligned} \lim_{t \rightarrow \infty} \tilde{P}(t) &= \lim_{t \rightarrow \infty} \frac{P\{S_t \notin \Omega_{\text{opt}}\}}{P\{S_0 \notin \Omega_{\text{opt}}\}} \\ &= \frac{1}{P\{S_0 \notin \Omega_{\text{opt}}\}} \lim_{t \rightarrow \infty} P\{S_t \notin \Omega_{\text{opt}}\} = 0. \end{aligned} \quad (17)$$

Therefore,

$$\prod_{i=0}^{\infty} P\{S_{i+1} \notin \Omega_{\text{opt}} \mid S_i \notin \Omega_{\text{opt}}\} = 0. \quad (18)$$

Let $\sigma_i = 1 - P\{S_{i+1} \notin \Omega_{\text{opt}} \mid S_i \notin \Omega_{\text{opt}}\}$; that is,

$$1 - \sigma_i = P\{S_{i+1} \notin \Omega_{\text{opt}} \mid S_i \notin \Omega_{\text{opt}}\}. \quad (19)$$

Then $\prod_{i=0}^{+\infty} (1 - \sigma_i) = 0$; by Definition, we have $\{\sigma_i\}_{i=0}^{+\infty}$ is a nonattenuation sequence.

As can be seen from the proof of necessity of Lemma, an absorbing Markov chain will reach the optimal state with probability 1 if iteration time tends to infinity. Furthermore, the AACRO algorithm can also be modeled as an absorbing Markov chain according to Theorem. So we can get AACRO can reach the optimal state with probability as long as the time allowed to evolve is sufficiently long.

4.2. Convergence Speed Analysis. As can be seen according to Definitions 11 and 12 in (see, e.g., [21]), the convergence rate at time t is defined as

$$\pi_t = P\{S_t \in \Omega_{\text{opt}}\} \quad t = 0, 1, 2, \dots, \quad (20)$$

where Ω_{opt} represents the optimal state set. The first hitting time in Definition 12 is denoted as

$$\begin{aligned} E[T_f] &= \sum_{t=1}^{\infty} t P\{S_t \in \Omega_{\text{opt}} \cap S_{t-1} \notin \Omega_{\text{opt}}\} \\ &= \sum_{t=1}^{\infty} t (\pi_t - \pi_{t-1}). \end{aligned} \quad (21)$$

From (20) and (21), we can see that the first hitting time and the convergence rate at time t are closely related. Intuitively speaking, the faster the convergence rate, the shorter the first collision time should be.

Although it is difficult to derive accurate convergence rate and first hitting time from the above (20) and (21), it provides a good mathematical basis for analyzing the convergence rate of AACRO.

This paper mainly studies the effect of *StepSize* on convergence rate. First, the effect of *StepSize* on the convergence rate is analyzed. For simplicity, it is assumed that the neighborhood operator is directly equal to the given step size, so that the continuous domain optimization problem becomes a discrete one, that is, the state space is finite.

Since the molecules are randomly generated in the state space, the probability that any molecule converges to a region near to the optimal solution is as follows:

$$\pi_t = P_{\text{opt}} = \text{popsize} \times \left(\frac{S_{\Delta}}{U_b - L_b} \right)^n, \quad (22)$$

where U_b and L_b correspond to the upper bound and lower bound of the state space, respectively, S_{Δ} is the optimal solution space, and n is the problem dimensions.

It can be seen from (22) that if the *StepSize* is equal to the difference between the upper and lower bounds, the algorithm has been completely changed to a random search process, and its efficiency will be greatly reduced. So the maximum value of the general step is half of the difference between the upper and lower bounds. Moreover, if the *StepSize* is large, the probability of reaching the optimal solution from the initial stage space is larger; that is, the convergence rate is larger. However, if the *StepSize* still remained unchanged, the probability of getting a worse solution will increase since each iteration has a quite large *StepSize*; we may have

$$P\{S_{i+1} \notin \Omega_{\text{opt}} \mid S_i \in \Omega_{\text{opt}}\} > 0, \quad (23)$$

which will result in reduction of the current convergence speed.

If we change *StepSize* by “1/5 success rule,” we will actually get a smaller search space (state space), then the state space and the convergence rate are changed as follows:

$$\begin{aligned} (S_A)' &= (\omega_1^A)' \times (\omega_2^A)' \times \dots \times (\omega_{n_{\max}}^A)', \\ \pi_t &= \text{popsize} \times \left(\frac{S_{\Delta}}{(U_b - L_b) / \text{rate}} \right)^n > \pi_t^{\text{initial}}, \end{aligned} \quad (24)$$

where rate denotes the change rate of the *StepSize* and π_t^{initial} represents the initial convergence speed.

We can see from (24) that π_t keeps increasing and maintains in a high speed. What is more, the neighborhood of AACRO combines the PSO's search operator, which further enhances the convergence rate.

It can be seen from the above analysis that the “1/5 success rule” strategy can greatly improve the convergence speed of the algorithm while the algorithm is guaranteed to converge on the global optimal solution.

5. Test Problems

In order to compare the performance between our proposed AACRO algorithm over the HP-CRO algorithm, we use a set of standard benchmark functions used in (see, e.g., [8]). The benchmark functions are shown in Table 1, which contains 23 benchmark functions with different dimension sizes, solution space S , and of course different global minimums $f_{\min}$, and they can be classified into three categories according to their different characteristics.

(1) *Unimodal Functions.* $f_1 - f_7$ belong to unimodal functions and the problem dimension sizes are all 30. However, these functions are relatively easy to solve since there is only one global minimum in each function.

(2) *High-Dimensional Multimodal Functions.* $f_8 - f_{13}$ belong to multimodal functions. The problem dimension sizes are also 30 but there are many local minimums in each function, so functions $f_8 - f_{13}$ are considered as the most difficult problems in these 23 benchmark functions.

TABLE 1: 23 benchmark functions.

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

TABLE I: Continued.

Category	Test function	Name	n	S	$f_{\min}$
III	$f_{14}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})^6} \right]^{-1}$	Shekel's Foxholes function	2	$[-65.536, 65.536]^n$	1
	$f_{15}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	Kowalik's function	4	$[-5, 5]^n$	0.0003075
	$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	Six-hump camel-back function	2	$[-5, 5]^n$	-1.0316285
	$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10$	Brain function	2	$[-5, 10] \times [0, 15]$	0.398
	$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	Goldstein-Price function	2	$[-2, 2]^n$	3
	$f_{19}(x) = -\sum_{i=1}^4 c_i \exp \left[-\sum_{j=1}^4 a_{ij} (x_j - p_{ij})^2 \right]$	Hartman's family	3	$[0, 1]^n$	-3.86
	$f_{20}(x) = -\sum_{i=1}^4 c_i \exp \left[-\sum_{j=1}^6 a_{ij} (x_j - p_{ij})^2 \right]$		6	$[0, 1]^n$	-3.32
	$f_{21}(x) = -\sum_{i=1}^5 \left[(x - a_i)(x - a_i)^T + c_i \right]^{-1}$	Shekel's family	4	$[0, 10]^n$	-10
	$f_{22}(x) = -\sum_{i=1}^7 \left[(x - a_i)(x - a_i)^T + c_i \right]^{-1}$		4	$[0, 10]^n$	-10
	$f_{23}(x) = -\sum_{i=1}^{10} \left[(x - a_i)(x - a_i)^T + c_i \right]^{-1}$		4	$[0, 10]^n$	-10

TABLE 2: Number of NFEs for function f_1-f_{23} .

Name	Function	AACRO	HP-CRO	RCCRO
Category I	$f_1, f_2, f_4, f_5, f_6, f_7$	150000	150000	150000
	f_3	250000	250000	250000
Category II	$f_8, f_{10}, f_{11}, f_{12}, f_{13}$	150000	150000	150000
	f_9	250000	250000	250000
Category III	f_{14}, f_{20}	7500	7500	7500
	f_{15}	250000	250000	250000
	f_{16}	1250	1250	1250
	f_{17}	5000	5000	5000
	f_{19}	4000	4000	4000
	$f_{18}, f_{21}, f_{22}, f_{23}$	10000	10000	10000

(a)

(b)

(c)

(d)

FIGURE 2: Results of G-best 25 run times of HP-CRO1 and AACRO: (a) f_2 , (b) f_5 , (c) f_7 , and (d) f_{10} .

TABLE 3: Parameter settings for different categories.

Order	Parameter	Category I	Category II	Category III	
		f_1-f_7	f_8-f_{13}	$f_{14}-f_{18}$	$f_{19}-f_{23}$
(1)	PopSize	10	100	100	100
(2)	w_{local}	0.2	0.5	0.8	0.8
(3)	w_{global}	0.8	0.5	0.2	0.2
(4)	CollRate	0.2	0.2	0.2	0.2
(5)	LossRate	0.3	0.3	0.3	0.3
(6)	IniKE	1000	1000	1000	1000
(7)	buffer	0	0	0	0
(8)	ChangeRate	0.001	0.001	0.001	0.001

FIGURE 3: Convergence curves of HP-CRO1 and AACRO: (a) f_2 , (b) f_5 , (c) f_7 , and (d) f_{10} .

TABLE 4: The optimization computing results for f_1-f_7 .

	AACRO	HP-CRO1	HP-CRO2	RCCRO1	RCCRO2	RCCRO4
f_1						
Mean	5.70E - 55	6.93E - 50	1.17E - 50	2.35E - 07	2.24E - 07	2.20E - 07
StdDev	2.85E - 55	3.39E - 49	4.18E - 50	4.70E - 08	4.43E - 08	3.83E - 08
Rank	1	3	2	6	5	4
f_2						
Mean	1.42E - 08	4.22E - 06	1.50E - 05	2.18E - 03	2.24E - 03	2.16E - 03
StdDev	8.89E - 09	1.28E - 05	3.50E - 05	3.65E - 04	4.14E - 04	3.32E - 04
Rank	1	2	3	5	6	4
f_3						
Mean	8.77E - 09	7.70E - 94	8.31E - 104	2.50E - 07	2.57E - 07	2.41E - 07
StdDev	4.32E - 08	3.77E - 93	2.98E - 103	9.35E - 08	9.18E - 08	8.60E - 08
Rank	3	2	1	5	6	4
f_4						
Mean	8.32E - 10	4.16E - 04	3.37E - 04	9.28E - 03	9.42E - 03	8.74E - 03
StdDev	5.88E - 09	3.05E - 04	2.99E - 04	3.97E - 03	4.92E - 03	3.63E - 03
Rank	1	3	2	5	6	4
f_5						
Mean	1.13E + 01	1.13E + 01	5.52E + 00	9.37E + 01	7.45E + 01	6.98E + 01
StdDev	1.50E + 01	1.53E + 01	7.13E + 00	9.61E + 01	9.30E + 01	1.03E + 02
Rank	2	3	1	6	5	4
f_6						
Mean	0	0	0	0	0	0
StdDev	0	0	0	0	0	0
Rank	1	1	1	1	1	1
f_7						
Mean	6.42E - 03	7.21E - 03	7.02E - 03	3.79E - 03	3.69E - 03	3.83E - 03
StdDev	1.03E - 02	2.31E - 03	2.00E - 03	1.25E - 03	9.65E - 04	1.31E - 03
Rank	4	6	5	2	1	3
Average rank	1.8	2.8	2.1	3.4	4.2	3.4
Overall rank	1	3	2	4	6	5

(3) *Low-Dimensional Multimodal Functions.* $f_{14}-f_{23}$ belong to low-dimensional multimodal functions. These functions are with the lowest problem dimensions and the search space S , and there exists some local minimums in each function, so they are more difficult than unimodal functions and easier than high-dimensional multimodal functions. The detailed introduction of these benchmark functions can be seen in [22].

6. Simulation Results

6.1. Experimental Environment. Both AACRO and HP-CRO are implemented in Matlab7.8. All simulations are performed on a computer with an Intel Core I5-4590 @ 3.3 GHz CPU and 4 GB of RAM in Windows 7 environment.

6.2. Parameter Setting. In order to achieve the best results, different test functions are set with different parameters and each simulation terminates when a certain number of

function evaluations (NFEs) have been performed. The NFEs limits of AACRO for different test functions are listed in Table 2; the parameter settings are listed in Table 3. The local weight and global weight of PSO are chosen as $c_1 = c_2 = 2$.

6.3. Comparisons. This paper also provides the results of the RCCRO versions of RCCRO1 (RCCRO1, RCCRO2, and RCCRO4) and HP-CRO (HP-CRO1 and HP-CRO2). All versions tested 23 standard test functions. For each function, we run AACRO 25 times and obtain the averaged computed minimum value (mean) and standard deviation (StdDev). We rank the results from the lowest mean to the highest and get the average rank. Finally, we order the average rank and get the overall rank.

As shown in Tables 4–6, the overall rank of AACRO is best and HP-CRO versions rank 2nd, followed by RCCRO versions. It is worth noting that each version has its specialty. In other words, no algorithm can work best on all functions. The AACRO version outperforms on $f_1, f_2, f_4, f_6, f_1, f_8, f_{10}$,

TABLE 5: The optimization computing results for f_8-f_{13} .

	AACRO	HP-CRO1	HP-CRO2	RCCRO1	RCCRO2	RCCRO4
f_8						
Mean	-1.26E + 04	-1.23E + 04	-1.26E + 04	-1.24E + 04	-1.23E + 04	-1.24E + 04
StdDev	3.07E + 01	1.79E + 02	4.34E + 01	7.11E + 01	9.16E + 01	6.94E + 01
Rank	1	5	2	4	6	3
f_9						
Mean	8.60E - 03	3.80E - 03	1.57E - 03	1.85E - 03	1.70E - 03	1.59E - 03
StdDev	8.02E - 02	1.02E - 02	6.07E - 03	4.62E - 04	4.86E - 04	4.94E - 04
Rank	6	5	1	4	3	2
f_{10}						
Mean	1.32E - 14	3.02E - 13	1.12E - 11	2.32E - 03	2.16E - 03	2.37E - 03
StdDev	1.36E - 14	9.44E - 13	5.43E - 11	4.43E - 04	2.93E - 04	3.83E - 04
Rank	1	2	3	5	4	6
f_{11}						
Mean	7.14E - 02	1	1	1.00E + 00	1.000001	1.00E + 00
StdDev	1.41E - 01	1.18E - 14	3.02E - 15	1.48E - 07	2.02E - 07	1.39E - 07
Rank	1	3	2	4	6	5
f_{12}						
Mean	1.22E - 01	3.08E - 01	1.54E - 01	5.82E - 01	3.56E - 01	1.60E - 01
StdDev	8.45E - 01	5.19E - 01	2.33E - 01	8.03E - 01	5.99E - 01	2.76E - 01
Rank	1	4	2	6	5	3
f_{13}						
Mean	3.75E - 32	2.63E - 14	3.81E - 14	6.66E - 06	7.71E - 06	6.46E - 06
StdDev	7.74E - 32	8.96E - 14	1.24E - 13	4.85E - 06	7.95E - 06	4.56E - 06
Rank	1	2	3	5	6	4
Average rank	1.8	3.5	2.1	4.6	5	3.8
Overall rank	1	3	2	5	6	4

f_{11} , f_{12} , f_{13} , f_{14} , f_{15} , f_{16} , f_{18} , f_{19} , f_{21} , f_{22} , and f_{23} . HP-CRO versions are on f_3 , f_5 , f_6 , f_9 , and f_{17} . RCCRO versions are on f_6 , f_7 , and f_{20} .

Table 4 gives the results for unimodal functions. From Table 4, we can see that AACRO outperforms the rest of the algorithms. AACRO behaves best on f_1 , f_2 , and f_4 , and the HP-CRO2 version has a better performance than other algorithms on functions f_3 and f_5 ; the RCCRO2 version ranks first on f_7 . For f_6 , all versions have the same performance, so they all rank first. The standard deviation of AACRO is also less than other versions.

For this table, AACRO ranks first, the 2nd highest overall rank is HP-CRO, and the RCCRO versions have the worst performance. In general, AACRO is efficient in solving unimodal functions.

Table 5 gives the results for high-dimensional multimodal functions (Category II). AACRO also has the best performance compared to others. Besides f_9 , AACRO outperforms the HP-CRO and RCCRO versions. The HP-CRO2 version performs best on f_9 . For RCCRO versions, the performances are all not ideal, except that RCCRO4 gets a 2nd rank on f_9 .

For this table, AACRO also ranks first, followed by HP-CRO versions, and the RCCRO versions ranks last. Therefore,

we can also believe that AACRO can treat high-dimensional multimodal functions well.

Table 6 gives the results for low-dimensional multimodal functions. From the overall rank we can see AACRO has a smaller standard deviation except f_{20} . AACRO outperforms the rest on f_{14} , f_{15} , f_{16} , f_{18} , f_{19} , f_{21} , f_{22} , and f_{23} , and f_{17} gets a 2nd rank. The RCCRO versions perform best on f_{20} . The 1st, 2nd, and 3rd ranks are all RCCRO versions, HP-CRO2 ranks 4th, AACRO ranks 5th, and HP-CRO1 ranks 6th. In general, AACRO is also efficient in solving low-dimensional multimodal functions.

From Tables 4–6, AACRO ranks 1st and HP-CRO versions ranks 2nd, followed by RCCRO versions. The details of AACRO are also presented, shown in Tables 7 and 8.

We can see from Tables 7 and 8 that when dealing with unimodal problems, our proposed ACCRO algorithm takes less average computation time than other problems. It is obvious since this kind of problem has only one global minimum in each of the functions and it is relatively “easy” to get the optimal solution. For the Hartman’s family problems (f_{19} and f_{20}), ACCRO takes the longest average computation time due to a “narrow” solution space S and a relatively complicated exponential function.

TABLE 6: The optimization computing results for $f_{14}-f_{23}$.

	AACRO	HP-CRO1	HP-CRO2	RCCRO1	RCCRO2	RCCRO4
f_{14}						
Mean	9.98E - 01	1.04E + 00	9.98E - 01	2.70E + 00	3.55E + 00	3.49E + 00
StdDev	1.01E - 09	1.95E - 01	9.64E - 07	1.81E + 00	1.74E + 00	2.58E + 00
Rank	1	3	2	4	6	5
f_{15}						
Mean	3.07E - 04	1.31E - 03	3.90E - 04	5.63E - 04	5.39E - 04	5.39E - 04
StdDev	5.17E - 13	3.95E - 03	2.46E - 04	8.56E - 05	1.10E - 04	9.58E - 05
Rank	1	6	2	5	4	3
f_{16}						
Mean	-1.03E + 00	-1.01E + 00	-1.02E + 00	-9.33E - 01	-9.67E - 01	-9.72E - 01
StdDev	1.57E - 15	2.39E - 02	1.27E - 02	1.49E - 01	1.01E - 01	5.81E - 02
Rank	1	3	2	6	5	4
f_{17}						
Mean	3.97E - 01	3.99E - 01	3.98E - 01	3.99E - 01	4.00E - 01	4.00E - 01
StdDev	2.36E - 15	8.06E - 04	3.74E - 04	7.09E - 03	9.10E - 03	7.28E - 03
Rank	2	3	1	4	6	5
f_{18}						
Mean	3.00E + 00	3.00E + 00	3.00E + 00	3.04E + 00	3.03E + 00	3.03E + 00
StdDev	2.51E - 14	1.55E - 04	8.11E - 04	3.56E - 02	2.47E - 02	4.39E - 02
Rank	1	2	3	5	4	6
f_{19}						
Mean	-3.86E + 00	-3.86E + 00	-3.86E + 00	-3.86E + 00	-3.86E + 00	-3.86E + 00
StdDev	2.20E - 14	4.99E - 03	3.61E - 03	2.28E - 03	1.06E - 02	2.29E - 03
Rank	1	5	4	2	6	3
f_{20}						
Mean	-3.26E + 00	-3.25E + 00	-3.27E + 00	-3.32E + 00	-3.32E + 00	-3.32E + 00
StdDev	4.72E - 01	3.21E - 02	2.62E - 02	2.72E - 03	3.72E - 03	3.82E - 03
Rank	5	6	4	1	2	3
f_{21}						
Mean	-1.02E + 01	-9.65E + 00	-9.53E + 00	-8.92E + 00	-8.43E + 00	-9.03E + 00
StdDev	6.28E - 14	2.97E - 01	5.33E - 01	2.15E + 00	2.78E + 00	2.25E + 00
Rank	1	2	3	5	6	4
f_{22}						
Mean	-1.04E + 01	-9.86E + 00	-9.82E + 00	-9.39E + 00	-9.24E + 00	-7.73E + 00
StdDev	6.28E - 14	2.85E - 01	3.45E - 01	2.31E + 00	2.42E + 00	3.08E + 00
Rank	1	2	3	4	5	6
f_{23}						
Mean	-1.05E + 01	-9.99E + 00	-9.95E + 00	-8.84E + 00	-9.09E + 00	-9.27E + 00
StdDev	1.42E - 11	3.76E - 01	5.50E - 01	3.04E + 00	2.58E + 00	2.46E + 00
Rank	1	2	3	6	5	4
Average rank	1.5	3.4	2.7	4.2	4.9	4.3
Overall rank	1	3	2	4	6	5

TABLE 7: Average computation time (f_1-f_7).

Function	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
Times(s)	0.00259	0.00116	0.00140	0.00243	0.00244	0.00262	0.00280	0.00144	0.00418	0.00114	0.00280	0.00347

TABLE 8: Average computation time (f_8-f_{13}).

Function	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}
Times(s)	0.00343	0.00030	0.00036	0.00030	0.00034	0.00034	0.00041	0.00046	0.00038	0.00039	0.00034

For a more detailed comparison of the AACRO proposed in this paper with the HP-CRO version, we give experimental results of some functions, the results of the execution on f_2 , f_5 , f_7 , and f_{10} are shown in Figure 2. The convergence curves on f_2 , f_5 , f_7 , and f_{10} are shown in Figure 3.

It was observed that the performance of AACRO is better than HP-CRO in Figures 2(a) and 2(d). It was worth noting that sometimes HP-CRO can have a better performance in Figure 2(c), and AACRO is better than HP-CRO in most cases in Figure 2(b). Moreover, we can see from Figure 3 that AACRO converges faster than HP-CRO.

7. Concluding and Future Work

In this paper, a new algorithm AACRO based on balanced local search and global search has been proposed. The algorithm combines the ACRO features and also incorporates the optimality operator of the PSO algorithm and adds the weighting factor to control the ratio of local search and global search. The algorithm's structure allows the algorithm to switch seamlessly between global and local searches efficiently, making it easier to find optimal values.

What is more, we give the convergence proof and convergence speed analysis. We draw a conclusion that the AACRO algorithm will keep converging with a high speed.

At last, this algorithm is simulated and compared with HP-CRO and RCCRO two versions of the algorithm. The results show that the algorithm can solve the optimization problem efficiently.

Our future work will be focused on the investigations of AACRO's parameters and figuring out the impact of each parameter, and the structure of the algorithm still needs to be streamlined. We expect to combine improved algorithms with engineering practice, which is also another key issue in the near future.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

- [1] D. E. Goldberg, "Genetic algorithms in search, optimization, and machine learning," *Choice Reviews Online*, vol. 27, no. 02, pp. 27-0936-27-0936, 1989.
- [2] S. Kirkpatrick, J. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *American Association for the Advancement of Science: Science*, vol. 220, no. 4598, pp. 671-680, 1983.
- [3] M. Dorigo, "Optimization, learning and natural algorithms," Tech. Rep., Italy, Politecnico di Milano, 1992.
- [4] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760-766, Springer US, Boston, MA, USA, 2011.
- [5] A. Y. S. Lam and V. O. K. Li, "Chemical reaction optimization: a tutorial," *Memetic Computing*, vol. 4, no. 1, pp. 3-17, 2012.
- [6] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67-82, 1997.
- [7] 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.
- [8] 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.
- [9] J. J. Q. Yu, A. Y. S. Lam, and V. O. K. Li, "Adaptive Chemical Reaction Optimization for global numerical optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2015*, pp. 3192-3199, jpn, May 2015.
- [10] B. Alatas, "ACROA: artificial chemical reaction optimization algorithm for global optimization," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13170-13180, 2011.
- [11] 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.
- [12] T. K. Truong, K. Li, and Y. Xu, "Chemical reaction optimization with greedy strategy for the 0-1 knapsack problem," *Applied Soft Computing*, vol. 13, no. 4, pp. 1774-1780, 2013.
- [13] 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.
- [14] W. Y. Szeto, Y. Liu, and S. C. Ho, "Chemical reaction optimization for solving a static bike repositioning problem," *Transportation Research Part D: Transport and Environment*, vol. 47, pp. 104-135, 2016.
- [15] Y. Xu, K. Li, L. He, L. Zhang, and K. Li, "A hybrid chemical reaction optimization scheme for task scheduling on heterogeneous computing systems," *IEEE Transaction on Parallel and Distributed System*, vol. 26, no. 12, pp. 3208-3222, 2015.
- [16] B. Doerr and C. Doerr, "Optimal parameter choices through self-adjustment: Applying the 1/5-th rule in discrete settings," in *Proceedings of the 16th Genetic and Evolutionary Computation Conference, GECCO 2015*, pp. 1335-1342, esp, July 2015.
- [17] 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.
- [18] R. J. Richards, "Darwin's theory of natural selection and its moral purpose," *Debates in Nineteenth-Century European Philosophy: Essential Readings and Contemporary Responses*, pp. 211-225, 2016.
- [19] <https://github.com/liangchen2017/AACRO>.
- [20] Z. Hu, S. Xiong, Q. Su, and Z. Fang, "Finite Markov chain analysis of classical differential evolution algorithm," *Journal of Computational & Applied Mathematics*, vol. 268, pp. 121-134, 2014.
- [21] A. Y. S. Lam, V. O. K. Li, and J. Xu, "On the convergence of chemical reaction optimization for combinatorial optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 605-620, 2013.
- [22] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82-102, 1999.

Submit your manuscripts at
www.hindawi.com