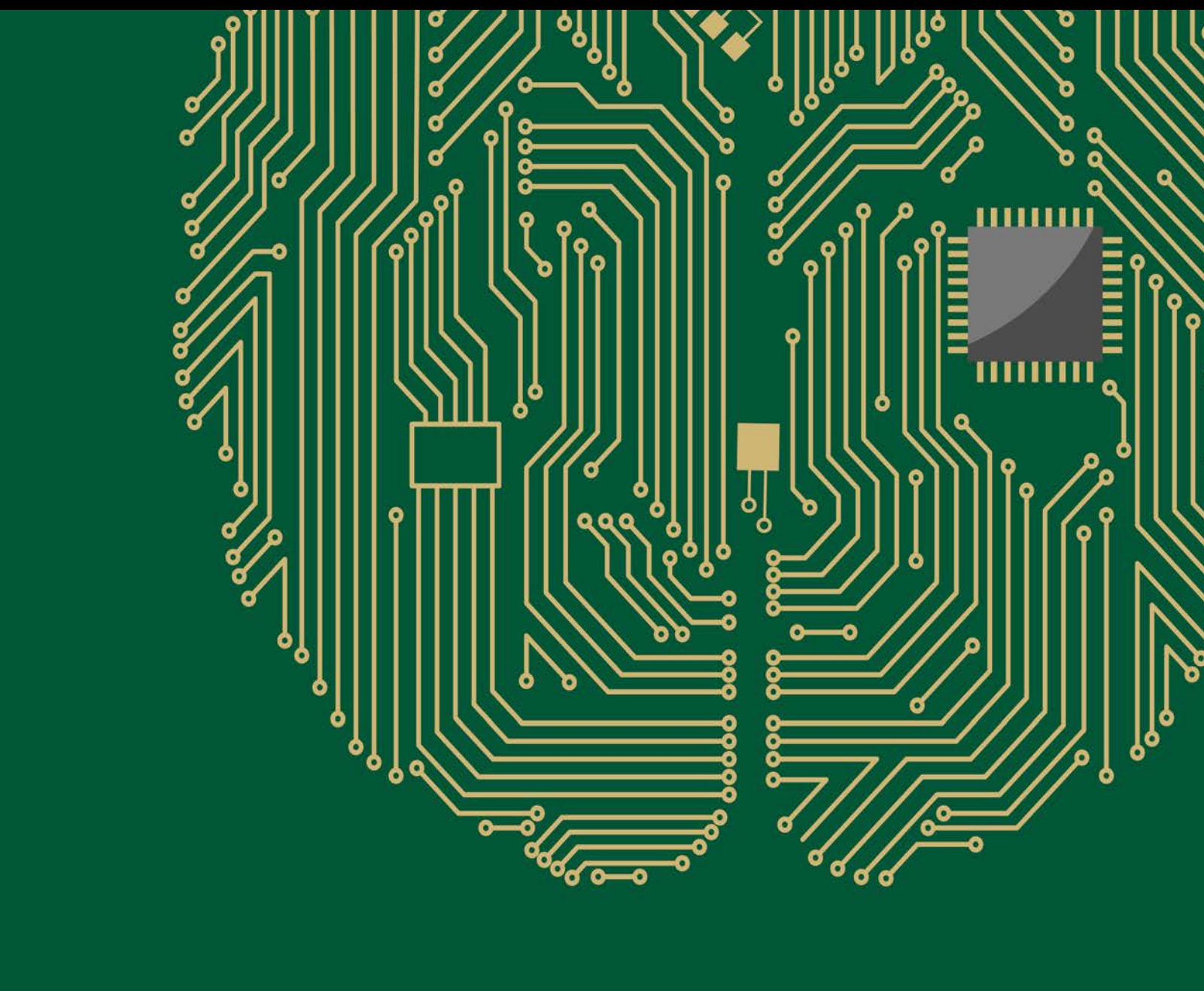
Computational Intelligence and Neuroscience

Biologically Inspired Methods for Imaging, Cognition, Vision, and Intelligence

Guest Editors: Yufeng Zheng, Erik Blasch, and Adel S. Elmaghraby



Biologically Inspired Methods for Imaging, Cognition, Vision, and Intelligence

Computational Intelligence and Neuroscience

Biologically Inspired Methods for Imaging, Cognition, Vision, and Intelligence

Guest Editors: Yufeng Zheng, Erik Blasch, and Adel S. Elmaghraby

Editorial Board

Contents

Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 2402067, 1 page
http://dx.doi.org/10.1155/2016/2402067

Editorial

Biologically Inspired Methods for Imaging, Cognition, Vision, and Intelligence

Yufeng Zheng,[1] Erik Blasch,[2] and Adel S. Elmaghraby[3]

[1]*Department of Advanced Technologies, Alcorn State University, Lorman, MS 39096, USA*
[2]*US Air Force Research Laboratory (AFRL), Rome, NY 13441, USA*
[3]*Department of Computer Engineering and Computer Science, University of Louisville, Louisville, KY 40292, USA*

Correspondence should be addressed to Yufeng Zheng; yzheng@alcorn.edu

Received 1 November 2015; Accepted 2 November 2015

State-of-the-art image analysis and pattern recognition techniques have been successfully applied to a wide variety of industrial fields such as medical imaging, remote sensing, and biometrics, as well as methods for machine learning, computer vision, and visualization. However, the current methods of image analysis, computer vision, and artificial intelligence cannot achieve the performance of human visual and cognitive abilities although advances have progressed in hardware including optics, electronics, and computers. As Scott [1] stated, "the evolution of biological vision is gradual (incremental) and top down with basic modules and primitive architectural structures preserved." The future of computer vision will be biologically inspired [2] that has much in common with human vision as a unified science of vision [1]. Thus, computational photography, cognition, and intelligence have recently become the sources of interactive scientific research.

This special issue focuses on biologically inspired or nature-driven approaches to computer vision. There were a total of 17 manuscripts received, five of which were accepted for publication. Two papers address computer vision problems: estimation of homographs between two different perspectives and computation of visual saliency. The other three papers present optimization algorithms for pattern recognition using an artificial bee colony approach, a differential evolution method, and a feed-forward neural network.

The success of computer vision and intelligence may start from the data representation of the real world (signal sensing, imaging), through analysis and cognition (learning, modelling, and classification), to achieve a certain level of performance (model revision, autolearning, and improvement). This special issue sheds light on the importance of bioinspired methods. Furthermore, editors believe that deeper research and development on bioinspired methods will greatly benefit the computational intelligence community.

Yufeng Zheng
Erik Blasch
Adel S. Elmaghraby

References

[1] P. D. Scott, "Applied machine vision," in *Science of Vision*, K. N. Leibovic, Ed., pp. 439–465, Springer, New York, NY, USA, 1990.

[2] I. Overington, *Computer Vision: A Unified, Biologically-Inspired Approach*, Elsevier Science, New York, NY, USA, 1992.

Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 9820294, 13 pages
http://dx.doi.org/10.1155/2016/9820294

Research Article

A New Modified Artificial Bee Colony Algorithm with Exponential Function Adaptive Steps

Wei Mao,[1] Heng-you Lan,[1,2] and Hao-ru Li[3]

[1]*Department of Mathematics, Sichuan University of Science & Engineering, Zigong, Sichuan 643000, China*
[2]*Key Laboratory of Higher Education of Sichuan Province for Enterprise Informationalization and Internet of Things, Zigong, Sichuan 643000, China*
[3]*School of Automation and Electronic Information, Sichuan University of Science & Engineering, Zigong, Sichuan 643000, China*

Correspondence should be addressed to Heng-you Lan; hengyoulan@163.com

Received 26 July 2015; Revised 10 October 2015; Accepted 19 October 2015

Academic Editor: Yufeng Zheng

As one of the most recent popular swarm intelligence techniques, artificial bee colony algorithm is poor at exploitation and has some defects such as slow search speed, poor population diversity, the stagnation in the working process, and being trapped into the local optimal solution. The purpose of this paper is to develop a new modified artificial bee colony algorithm in view of the initial population structure, subpopulation groups, step updating, and population elimination. Further, depending on opposition-based learning theory and the new modified algorithms, an improved S-type grouping method is proposed and the original way of roulette wheel selection is substituted through sensitivity-pheromone way. Then, an adaptive step with exponential functions is designed for replacing the original random step. Finally, based on the new test function versions CEC13, six benchmark functions with the dimensions $D = 20$ and $D = 40$ are chosen and applied in the experiments for analyzing and comparing the iteration speed and accuracy of the new modified algorithms. The experimental results show that the new modified algorithm has faster and more stable searching and can quickly increase poor population diversity and bring out the global optimal solutions.

1. Introduction

It is well known that algorithms for solving various characteristics optimization problems can be classified into different groups, such as population-based algorithms, stochastic algorithms, deterministic algorithms, and iterative algorithms. An algorithm is called population-based [1] if one works with a group of solutions and tries to improve them. Two important classes of population-based optimization algorithms are exactly evolutionary algorithms and swarm intelligence-based algorithms [2]. Swarm intelligence is an innovative artificial intelligence technique with collective behavior of self-organized systems [3]. Since many swarm intelligence algorithms, such as genetic algorithm (GA) [4], particle swarm optimization (PSO) [5], ant colony optimization (ACO) [6], and biogeography-based optimization [7], have simplicity, ease of implementation, outstanding performance, and other advantages [8], they have shown great success in solving some nonconvex, discontinuous, or nondifferentiable optimization problems. However, these intelligence algorithms are sensitive to value and precision. Thus, inspired by the behavior of honey bees, Karaboga [9] introduced basic artificial bee colony algorithm (ABC) in 2005 and constituted one of the most prominent approaches in the field of bee-inspired algorithms. Further, in consideration of the solution reaching speed, the success rate, and the performance rate, El-Abd [10] provided a complete performance assessment of ABC and compared it with the widely known differential evolution (DE), GA, heuristic algorithms, PSO, and other foraging algorithms (e.g., bacterial algorithm, ACO, and bacterial foraging optimization) by using the well-known benchmark functions in [11].

It was claimed that ABC is the most successful algorithm for multimodal and hybrid functions [12]. This is because ABC has no demand to the objective function, constraint, and external information and is only based on fitness function in the search process [13]. Further, ABC has the following advantages. (i) The mechanism of multiple roles:

using different methods, bees adjust quality of the solutions spontaneously, so as to adapt to the next search process [14]. (ii) The cooperative working mechanism: according to the information from other bees, bees decide whether to find the optimal solution with larger probability [15]. (iii) The strong robustness: the search rules are not certain but are probabilistic and have excellent robustness and a wide range of applicability [16]. (iv) The stability: even if the individual fails, the entire swarm can still complete the task [17]. (v) Less control parameters, simple operation, and ease of implementation [18]: indeed, ABC has been shown to be very competitive with respect to other state-of-the-art foraging and evolutionary algorithms.

However, there exists still an insufficiency to ABC because ABC does well in exploration but is poor at exploitation. As for the improvement and development of ABC, Karaboga and Gorkemli [19] proposed a new update rule for onlooker bees in the hive to improve the local search and convergence characteristics of ABC. Inspired by PSO, Imanian et al. [20] changed the update rule of basic ABC to increase the convergence speed for solving high dimensional and continuous optimization problems. Wang et al. [21] proposed multistrategy ensemble artificial bee colony algorithm (MEABC) to improve the local and global search capability of basic ABC and tested the performance of MEABC by using basic, shifted, and rotated benchmark functions. Gao et al. [22] developed new search equations to adjust exploration and exploitation capability of ABC. In a different approach for ABC, Das et al. [23] proposed a learning routine based on fitness and proximity stimuli and tested the method with standard benchmark functions. Zang et al. [24] designed a logarithmic function adaptive step instead of the original random step.

Moreover, in dealing with some complex problems by applying ABC, there are some defects such as slow search speed, poor population diversity, stagnation in the working process, and trapping into the local optimal solution [13]. Recently, ABC has been extended and improved by many researches. But since ABC is relatively new, the researches in the literatures lack systematicness and are scattered. See, for example, [1–3, 12–30] and the references therein. In 2012, Li et al. [31] pointed out that "ABC has no mechanism to use the global information in the search space, so it easily results in a waste of computing power and gets trapped in local optima". Further, the authors proposed a novel algorithm (named as DEABC, i.e., differential evolution artificial bee colony algorithm), which synthesizes DE and ABC and enhances individuals by sharing information between DE population and bee colony. For related works, one can see [1, 30] and the references therein.

Motivated and inspired by the above works, a new modified artificial bee colony (MABC) algorithm shall be constructed based on adaptive step with exponential functions. By using opposition-based learning theory and an improved *S*-type grouping method, the initial population for MABC will be given, and the original way of roulette wheel selection shall be substituted by sensitivity-pheromone way. Specifically, an adaptive step with exponential functions will be designed to replace the original random step. In order to verify the validity of the improved algorithm, MABC, it is compared with DEABC [30], novel artificial bee colony algorithm (NABC) [8], and ABC; the experiment results tested with six well-known benchmark optimization functions, which are chosen from new test function versions CEC13, show that MABC is superior to ABC, NABC, and DEABC.

The rest of this paper is organized as follows. Section 2 gives a brief introduction to ABC. The new MABC is presented and analyzed in Section 3. Section 4 presents and discusses the experimental results of six benchmark functions with the dimensions $D = 20$ and $D = 40$, respectively. Finally, the conclusion is drawn in Section 5.

2. Brief Review to ABC

In this section, a brief review on ABC is going to be given.

2.1. Thoughts of the Algorithm. A honey bee colony can successfully discover the highest quality food sources in nature. Hence, the idea of ABC comes from intelligent foraging behavior of honey bees to finding good solutions for solving optimization problems. In a general way, according to the ways of searching food, the colony of bees is divided into three kinds: employed bees, onlooker bees, and scout bees. The employed bees are responsible for exploiting the nectar sources. They explore the beforehand food source position and give the quality information of the food to the onlooker bees. The onlooker bees wait in the hive and decide to exploit a food source based on the information shared by the employed bees. In order to find a new nectar source, the scout bees randomly search environment either depending on an internal motivation or based on possible external clues [32]. The position of a nectar source implies a possible solution of the optimization problems, and the profitability of a nectar source corresponds to the quality (fitness) of the possible solution. Each nectar source is exploited by only one employed bee. In other words, the number of nectar sources equals the number of employed bees or onlooker bees [25]. In this process, the employed bees maintain good solution, the onlooker bees improve convergence speed, and the scout bees enhance the ability to remove local optimum [26, 31].

2.2. ABC Iteration Steps. From [24, 33], it follows that main iteration steps of ABC can be listed as follows.

Step 1 (initialization). Randomly generate N solutions (i.e., food sources) $\{x_1, x_2, \ldots, x_N\}$ as initial population in a D dimension searching space, where N is the number of food sources, which equals the half of the colony size, $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ is a D-dimensional solution vector and the ith food source in the initial population for $i = 1, 2, \ldots, N$, and D also denotes the number of optimization parameters.

Step 2 (renewing population). In the stage of collecting honey, each employed bee produces a new nectar source within the neighborhood of the food source. After comparing the new nectar source with the old ones, the high probability

will be memorized. Next, in the stage of follow, every onlooker bee evaluates the profitability of nectar sources taken from all employed bees and then chooses a food source at a certain probability. As in the case of the employed bees, she produces a modification on the source position in her memory and keeps the better nectar source. The regeneration of nectar sources in these two stages is based on the following formula:

$$v_{ij} = x_{ij} + \text{rand} \cdot \left(x_{ij} - x_{kj}\right), \tag{1}$$

where $i, k = 1, 2, \ldots, N$, $j = 1, 2, \ldots, D$, and rand $\in [0, 1]$ is a random number, which controls the generation range of neighborhood of x_{ij}. With the search being close to optimal solution, the range of neighborhood will become smaller and smaller.

Step 3 (nectar source selection). In the stage of follow, the onlooker bees choose food source through comparing the probability which is computed by the fitness value. The nectar sources of high probability are selected in large probability. And the probability of being selected for the food sources is calculated as follows:

$$P_i = \frac{\text{Fit}_i}{\sum_{i=1}^{N} \text{Fit}_i}, \tag{2}$$

where Fit_i ($i = 1, 2, \ldots, N$) is the fitness (profitability) value of the ith solution, which is obtained by the following equation:

$$\text{Fit}_i = \begin{cases} \dfrac{1}{1 + f_i}, & \text{if } f_i \geq 0, \\ 1 + \left|f_i\right|, & \text{if } f_i < 0, \end{cases} \tag{3}$$

where f_i is the objective function value for $i = 1, 2, \ldots, N$, which is specific for the optimization problem. If the new food source position has a quality equal to or better than the old one, then the old food source position is replaced by the new one. Otherwise, the old one is retained, which is the same as the employed bees stage.

Step 4 (population elimination). If a solution has not been improved significantly through a predetermined number of trials, called "max iteration," then the solution is regarded as falling into a local optimal solution and their original position will be abandoned. Thus, the corresponding employed bees will become scout bees and a new solution instead of the eliminated solution is randomly generated, which can be expressed as follows:

$$x_{ij} = \text{rand} \cdot \left(x_{\max j} - x_{\min j}\right) + x_{\min j}, \tag{4}$$

where i, j are the same as in (1) and $x_{\max j}$ and $x_{\min j}$ denote the jth individual maximum and the ith individual minimum values, respectively.

Based on the above iteration steps, the process of ABC [12] can be shown in Algorithm 1.

3. New MABC

From the above introductions, it follows that ABC has many advantages, such as simplicity, ease of implementation, and outstanding performance. However, the algorithm steps indicate that ABC has the following obvious flaws:

(i) The randomly generated initial population will lead the solution to be random distribution in the solution space. Hence, the searching ability of ABC will be affected directly.

(ii) All individuals begin to search directly in the whole solution space, which will reduce searching efficiency.

(iii) The population is regenerated by random step length. When the algorithm performs in the neighborhood of optimal solutions, the searching range will be severely restricted. Thus, the convergence rate and optimization precision of ABC will be influenced.

In order to overcome the existing deficient problem of ABC and related algorithms, the purpose of this paper is to introduce and study a new MABC and to improve the initial population structure, the population grouping, and the population regeneration of ABC.

3.1. Opposition-Based Learning of Population Initialization. ABC is relatively sensitive to the initial population construction. Therefore, the search range of ABC will be restricted and the global search ability of ABC will also be influenced when the initial population distribution in the search space or the local area is random. However, the initial population constructed by using opposition-based learning method has the diversity as much as possible, is better representative, and meets the accuracy of experimental requirements (see, e.g., [6, 13, 34, 35] and the references therein).

Based on the opposition-based learning method, the initial population for MABC will be constructed in this paper. Firstly, an initial solution is given, and next the corresponding reverse initial solution to the initial solution is produced. Then, the two kinds of solutions are sorted. Finally, the better fitness solution is chosen as the initial population. This method will help to enhance the efficiency of MABC and to improve the quality of solutions [34]. The processes can be listed as shown in Algorithm 2.

3.2. S-Type Subpopulation Grouping. Intelligent optimization algorithms commonly use general grouping method as follows. Firstly, the population fitness function value of each individual is calculated. Secondly, the fitness function value is arranged in descending order. Then, the entire ordered population is divided into M subpopulations, and the process of the grouping [24] is as shown in Table 1. However, this grouping method still causes some problems: the relative advantage of individuals is divided into the same species and their disadvantage is concentrated in other uncertain groups. This phenomenon will make very large differences between subpopulations, and this grouping method is not conducive to the expansion of population diversity.

Input: N, max iteration
Output: Best solution
(1) Initialize the food sources using the following equation
$x_{ij} = \text{rand} \cdot (ub_j - lb_j) + lb_j$ for $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, D$,
where ub_j and lb_j are upper and lower boundaries of parameter j, respectively
(2) Evaluate the food sources using (3)
(3) Iter = 1, where Iter denotes iteration number
(4) While Iter ≤ max iteration do
(5) Send employed bees to produce new solutions as in (1)
(6) Evaluate the new solutions and apply greedy selection
(7) Calculate the probability using (2)
(8) Send onlooker bees to produce new greedy selection
(9) Evaluate the new solutions and apply greedy selection
(10) Send one scout into the search area for discovering a new food source
(11) Memorize the best solution found so far
(12) Iter = Iter + 1
(13) end while
(14) Return the best solution

Algorithm 1: Flowchart of ABC.

Initialize the population size N.
{The random initialization phase}
for $i = 1$ to N do
for $j = 1$ to D do

$$x_{ij} = \text{rand} \cdot (x_{\max j} - x_{\min j}) + x_{\min j}, \qquad (*)$$

end for
end for
{The opposition-based learning phase}
for $i = 1$ to N do
for $j = 1$ to D do

$$ox_{ij} = x_{\max j} + x_{\min j} - x_{ij}, \qquad (**)$$

end for
end for
where i, j, $x_{\max j}$ and $x_{\min j}$ are the same as in (4). Choosing the best adaptive value N of individual from $\{x_{ij} \cup ox_{ij}\}$ as initial population.

Algorithm 2

To reduce the difference between each group (see [24]), a new grouping method is presented, which is called S-type subpopulation grouping. Similarly, the fitness function value is arranged in descending order. But, relative to the general grouping method, the order of the value in the new groups is different, and the process of new grouping is listed in Table 2.

Indeed, first of all, the fitness function values of each individual of population are calculated, and the fitness function values in descending order are ranked. Next, the entire population is divided into M subpopulations $S_1, S_2, \ldots, S_M$. Let Y_i be the ith individual for $i = 1, 2, \ldots, N$. If the solutions due to S-type subpopulation grouping are odd rounds and there are a large number subpopulations, then there will be more differences between each group. Thus, the final round can be abandoned through S-type subpopulation grouping. Now one can see that S-type subpopulation grouping can overcome the disequilibrium between groups, making the subpopulation have a little difference, which is good to the expansion of population diversity.

Furthermore, a new population will be recomposed by all subpopulations when each subpopulation group is less than max iteration N of intragroup. And then the fitness function values of each population are recalculated and are arranged according to the descending order. Only one of the individuals of equivalent fitness function values is kept. Let the number of remaining individuals at last be K. Then the population can be regenerated as follows:

$$\begin{aligned} &\text{the top } M \text{ individuals are retained,} && \text{if } K \geq M, \\ &M\text{-}K \text{ individuals will randomly be initialized and filled into the population,} && \text{if } K < M. \end{aligned} \qquad (5)$$

Table 1: General grouping process.

Solutions	Subpopulations			
	S_1	S_2	$\cdots$	S_M
The first round	Y_1	Y_2	$\cdots$	Y_M
The second round	Y_{M+1}	Y_{M+2}	$\cdots$	Y_{2M}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 2: S-type subpopulation grouping.

Solutions	Subpopulations			
	S_1	S_2	$\cdots$	S_M
The first round	Y_1	Y_2	$\cdots$	Y_M
The second round	Y_{2M}	Y_{2M-1}	$\cdots$	Y_{M+1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
The final round	Y_{2nM}	Y_{2nM-1}	$\cdots$	$Y_{(2n-1)M+1}$
	$\vdots$	Y_{2nM+K}	$\cdots$	Y_{2nM+1}

3.3. Sensitivity-Pheromone Option. Since the onlooker bees chosen by roulette way are too greedy and the population diversity is deduced too quickly, it is easily plunged into local optimum by using ABC for solving function optimization problems. In applications, the roulette way is practically affected by the population diversity.

Thus, it is an important idea for sensitivity to replace the roulette way. Sensitivity is put forward by free search algorithm [35] with pheromone (related to optimization of adaptive value) to selected area, which can be expressed as follows:

(i) Calculate the N values of fitness to the individual $f(x)$.

(ii) For each $i = 1, 2, \ldots, N$, calculate the ith pheromone

$$p(i) = \frac{a \cdot f(i)}{f_{\max}} + (1 - a), \quad a = \frac{f_{\max}}{\sum_{i=1}^{N} f_i}, \tag{6}$$

where $f_{\max}$ stands for the maximum values of $f(i)$ for $i = 1, 2, \ldots, N$.

(iii) Generate the jth individual sensitivity randomly: $S(j) \sim U(0, 1)$.

(iv) Find out sensitivity area i matching individual of the first j and randomly find i to meet $p(i) \geq S(i)$.

By the above model, it is easy to see the following: (i) since the sensitivity-pheromone is random, any area can be searched in theory, and the local optimum can largely be avoided by using the model. (ii) The fourth step of the model is to make MABC have a clear direction, and convergence or divergence of the objective function in the search space can be determined quickly. This method is similar to the roulette way of nectar source for the onlooker bees, and so sensitivity and pheromone option instead of roulette way can be applied.

3.4. Updating Step Length with Exponential Function. According to solution search equation (1) of ABC, the new candidate solution is generated through moving the old one towards another selected random solution of the population. This method can increase the probability of locating global optimal solution, but it cannot guarantee obtaining a better solution. Hence, the convergence speed is slow. If each new candidate solution is better than the old one, then convergence speed will become faster. However, if only the better solution is considered, then the algorithm could get trapped into local optimum [32]. In ABC, the updating step length denoted by σ (i.e., rand in (1)) plays a very important role and controls the ability to search (local or global) and the convergence rate. On the other hand, if the value of σ is too large, then the algorithm is easy to lose the global optimal solution. And, on the contrary, the algorithm is easy to show the phenomenon of stagnation and fall into local optimum [33]. Therefore, it is very necessary to design a new suitable step length associated with the performance of the algorithm.

The following exponential function is used in this paper to update step length:

$$\sigma = (\text{rand} - 0.5) \cdot 2^{\text{rand}}, \tag{7}$$

and the improved population moving step length is

$$\Delta = \sigma \cdot \left(x_{ij} - x_{kj}\right) = (\text{rand} - 0.5) \cdot 2^{\text{rand}} \cdot \left(x_{ij} - x_{kj}\right), \tag{8}$$

for $i, k = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, D$, where rand is the same as in (1). Thus, the location of the updated solution is

$$\begin{aligned} v_{ij} &= x_{ij} + \sigma \cdot \left(x_{ij} - x_{kj}\right) \\ &= x_{ij} + (\text{rand} - 0.5) \cdot 2^{\text{rand}} \cdot \left(x_{ij} - x_{kj}\right) \end{aligned} \tag{9}$$

and the location of the random generated solutions is

$$x_{ij} = x_{\min j} + (\text{rand} - 0.5) \cdot 2^{\text{rand}} \cdot \left(x_{\max j} - x_{\min j}\right), \tag{10}$$

where $x_{\max j}$ and $x_{\min i}$ are the same as in $(*)$.

Now one can know that with the increase of evolution algebra, the updating step will be increased and become larger and larger. In the early stage of the evolution, the improved evolution algebra can be conducted a comprehensive local search, and make the algorithm better position to the target area. In the middle and later periods of the evolution, the improved evolution algebra will be carried out global search and can speed up the convergence rate so as to reach to optimal solution.

3.5. Flowchart of MABC. To clearly know MABC, the detailed flowchart of MABC is shown in Algorithm 3, where the max iteration as the termination criteria is adopted the same as in Algorithm 1, and CS denotes the number of colony size, which is equal to the sum of the numbers of employed bees and onlooker bees.

(1) Initialization phase
(1.1) Food Number = CS/2, that is, the number of food sources equals to half of the colony size
(1.2) Based on the opposition-based learning of population initialization as in Section 3.1
(1.3) Calculate their fitness values
(1.4) Based on *S*-type sub-population grouping described in Section 3.2
(1.5) trial = zeros (1, Food Number). Initialize trial counters to be zeros
(1.6) Initialize elitists to be food sources
(1.7) Memorize the best food source
Iter = 0;
while Iter ≤ max iteration
  Select a better elitist by sensitivity-pheromone option in Section 3.3
  (2) Employed bees phase
    for i = 1 to Food Number
      (2.1) Update step length of each food source by (7) and generate a new solution by (10)
      (2.2) Evaluate the new solution
      (2.3) Iter = Iter + 1
      (2.4) A sensitivity-pheromone selection is applied between the current solution i and its mutant.
        If the mutant solution is better than the current solution i, replace the solution
        with the mutant and reset the trial counter of solution i. Otherwise, increase
        its trial counter.
    end
  Calculate probabilities using (6)
  (3) Onlooker bees phase
    (3.1) Select a food source by sensitivity-pheromone method in Section 3.3
    (3.2) For each onlooker bee, update step length of each food source by (7) and generate a new solution by (10)
    (3.3) Iter = Iter + 1
    (3.4) A sensitivity-pheromone selection is applied between the current solution and its mutant.
      If the mutant solution is better than the current solution, replace the solution
      with the mutant and reset the trial counter of solution.
      Otherwise, increase its trial counter.
  The optimal solution is memorized
  (4) Scout bees phase
    (4.1) Determine the food sources whose trial counter exceeds the "limit" value,
      and only one scout bee is allowed to occur in each iteration
    (4.2) Use a scout bee to carry out randomly search
  if Food Number ≥ max iteration
end while
Return *the optimal solution*

Algorithm 3: Flowchart of MABC.

4. Simulation and Analysis

In this section, the performance and accuracy of MABC, ABC, NABC, and DEABC will be analyzed and compared through the figures, the total run time, mean value, and standard deviation of six benchmark functions from the new test function versions CEC13, respectively.

4.1. Convergence Performance Analysis. In order to verify the validity of MABC, six benchmark functions listed in Table 3 (see [11, 27]) are used in the experiments. According to their characteristics, these six functions are divided into two groups. The Sphere, SumSquares, and Schwefel 2.22 are unimodal functions, and Ackley, Griewank, and Rastrigin are multimodal functions. For each benchmark function, two dimension sizes, 20 and 40, of solution space are tested. The colony size is 80 and the max iteration is 1500; limit is 100. Each of the experiments is repeated 10 times.

In order to further compare the performance of four proposed algorithms, convergence performance of the six functions for ABC, DEABC, NABC, and MABC is plotted to dimension sizes 20 and 40, respectively. See Figures 1–6.

From the convergence performances of SumSquares, Griewank, and Rastrigin functions, one can see that when $D = 20$, ABC converges with the slowest speed and locates a local optimal solution. DEABC has the fastest speed at the beginning and goes into the global optimal solution. NABC also goes into the global optimal solution, but the speed is slower than those of DEABC and MABC. Further, MABC has the fastest speed at the end and is the first one to attain the true global number. When $D = 40$, ABC and DEABC fall into the global solution. NABC has the fastest speed at the beginning and attains the global optimal solution at the end, but the true global number of NABC is smaller than that of MABC. MABC goes into the global optimal solution at the end, and the speed is faster than that of NABC.

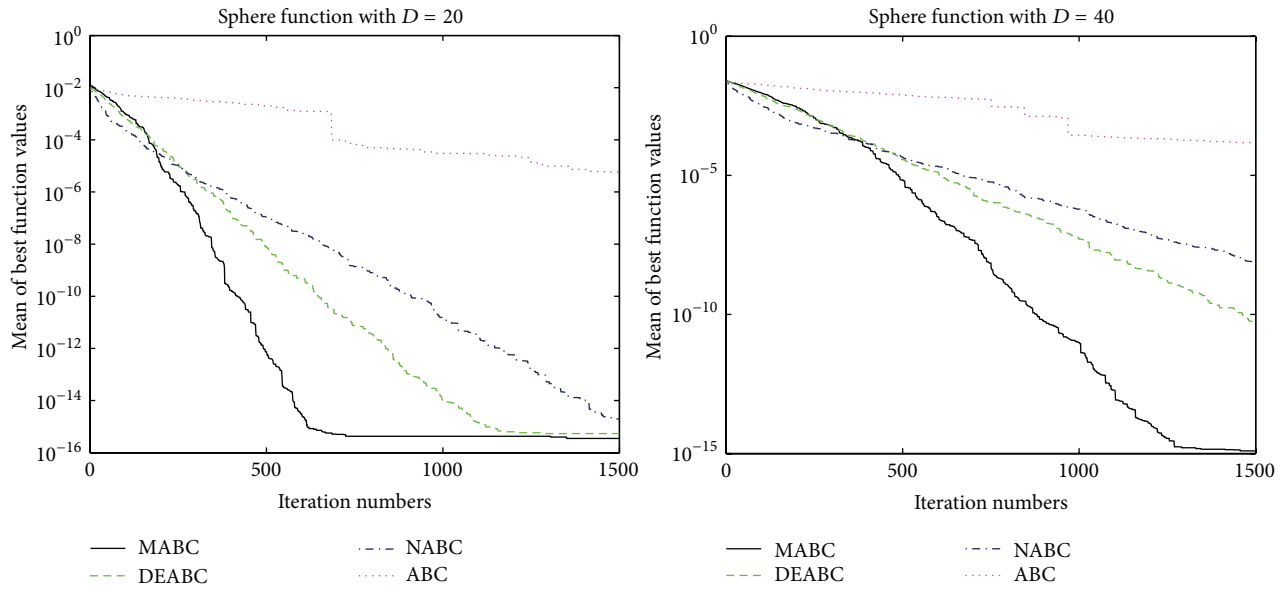


Figure 1: Convergence performance of ABC, DEABC, NABC, and MABC.

Table 3: Numerical benchmark functions.

Name	Formula	Search domain	Minimum
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0
SumSquares	$f_2(x) = \sum_{i=1}^{D} i x_i^2$	$[-10, 10]^D$	0
Schwefel 2.22	$f_3(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-100, 100]^D$	0
Griewank	$f_4(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^D$	0
Rastrigin	$f_5(x) = \sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	$[-5.12, 5.12]^D$	0
Ackley	$f_6(x) = -20e^{-0.2\sqrt{(\sum_{i=1}^{D} x_i^2)/D}} - e^{(\sum_{i=1}^{D} \cos(2\pi x_i))/D} + 20 + e$	$[-32, 32]^D$	0

The convergence performance of Sphere function exhibits that various behavior for both dimension sizes. When $D = 20$, ABC converges with the slowest speed, and NABC and DEABC have fastest speed in the first place. Later on, MABC is the first to arrive to the global optimal solution, the second one is DEABC, and the third one is NABC. And ABC locates a local optimal solution. When $D = 40$, ABC also goes into the local optimal. NABC, DEABC, and MABC have the similar speed originally; hereafter MABC has quicker speed than NABC and DEABC. Finally, these three algorithms come to the global optimal solution, but the best one is MABC, the second one is DEABC, and the last one is NABC.

The convergence performances of Schwefel 2.22 and Ackley functions demonstrate that, for both dimension sizes of 20 and 40, ABC, NABC, and DEABC all fall into the global optimal solution, but DEABC converges with the fastest speed originally. Later on, MABC has the fastest speed and locates the global optimal solution at the end.

From Figures 1–6, it can be comprehensively deduced that ABC converges with the slowest speed and tends to a local optimal solution. DEABC and NABC have faster speed than ABC but sometimes enter into local optimal solution. MABC has fastest speed at the end and achieves the really global optimal solution.

4.2. Comparison of Mean Value and Standard Deviation. The results of comparing the mean value and standard deviation are shown in Table 4, where the first column shows the benchmark functions, the "D" column represents the dimension of function, the "Mean" column contains the average best values of benchmark functions, and the "SD" column shows the standard deviation of the best value.

For six benchmark functions, one can know that when $D = 20$, the mean values of ABC, NABC, DEABC, and MABC are better than those of them when $D = 40$. For SumSquares, Griewank, and Rastrigin functions, it is

Table 4: Comparison of the mean value and standard deviation.

Function	D	ABC		NABC	
		Mean	SD	Mean	SD
Sphere	20	$9.12643e-4$	$6.80695e-4$	$8.51338e-16$	$2.29525e-16$
	40	0.445472	0.234793	$3.7323e-11$	$1.50446e-12$
SumSquares	20	$1.3749e-5$	$2.33632e-5$	$8.75587e-16$	$2.30537e-16$
	40	0.994557	1.70504	$4.65058e-10$	$2.77676e-10$
Schwefel 2.22	20	$9.42221e-4$	$7.46032e-4$	$2.22277e-8$	$1.00025e-8$
	40	0.0461286	0.0287431	$2.15524e-4$	$6.03173e-5$
Griewank	20	$2.5264e-6$	$1.43918e-6$	$4.44089e-16$	$1.92296e-16$
	40	0.0125412	0.00802599	$1.07636e-12$	$5.37341e-13$
Rastrigin	20	0.0331619	0.0574246	$7.75912e-14$	$3.28186e-14$
	40	15.5119	20.1043	$6.00988e-9$	$2.57688e-9$
Ackley	20	$4.22652e-3$	$3.9353e-3$	$7.97212e-10$	$4.51507e-10$
	40	0.603485	0.0057116	$3.77068e-6$	$7.14915e-7$
Function	D	DEABC		MABC	
		Mean	SD	Mean	SD
Sphere	20	$5.33457e-16$	$6.65399e-18$	$4.21651e-16$	$8.89001e-18$
	40	$5.54516e-13$	$3.68742e-13$	$1.49866e-15$	$2.47044e-16$
SumSquares	20	$6.03584e-16$	$1.3174e-16$	$4.16975e-16$	$1.01262e-16$
	40	$5.20275e-10$	$6.16741e-7$	$3.29277e-15$	$2.28728e-15$
Schwefel 2.22	20	$3.61252e-6$	$1.24222e-6$	$8.79872e-16$	$1.09679e-16$
	40	$1.08779e-4$	$1.65357e-4$	$4.12959e-10$	$1.80491e-10$
Griewank	20	$3.70074e-16$	$2.31111e-16$	$1.11022e-16$	0
	40	$6.54372e-11$	$8.03242e-11$	$8.14164e-16$	$2.34132e-16$
Rastrigin	20	$2.84217e-14$	$2.84217e-14$	$9.4739e-15$	$1.64093e-14$
	40	$4.40958e-7$	$3.26047e-7$	$1.89478e-13$	$3.28186e-14$
Ackley	20	$1.21723e-6$	$1.197e-6$	$3.16784e-14$	$2.05116e-15$
	40	$7.67309e-7$	$5.03943e-7$	$5.01809e-9$	$1.85715e-9$

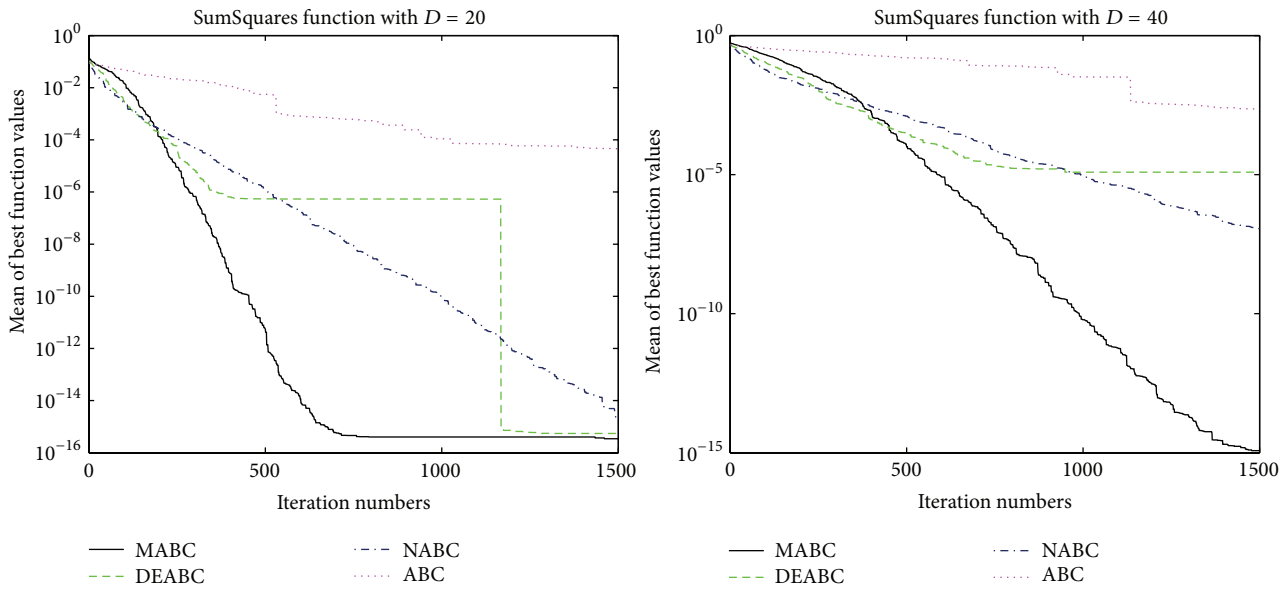


Figure 2: Convergence performance of ABC, DEABC, NABC, and MABC.

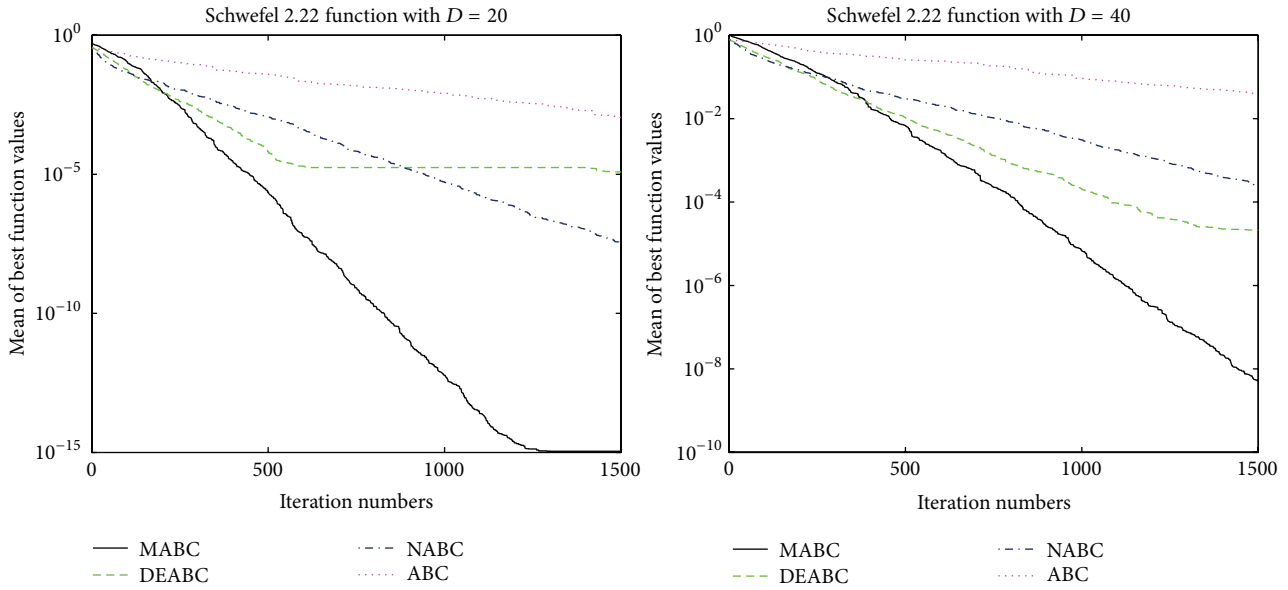


Figure 3: Convergence performance of ABC, DEABC, NABC, and MABC.

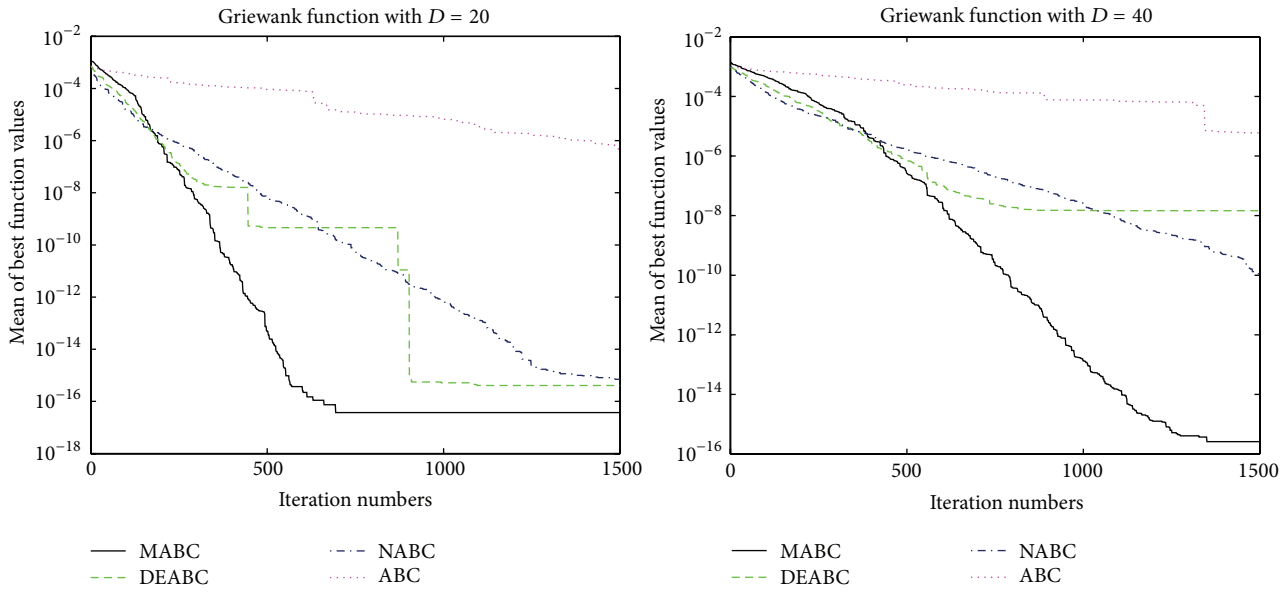


Figure 4: Convergence performance of ABC, DEABC, NABC, and MABC.

observed that when $D = 40$, the mean value of NABC has better mean than that of DEABC. On the other hand, in terms of standard deviation, ABC and NABC are less than DEABC for Sphere and Ackley functions. In addition, MABC has better average best values than DEABC, and DEABC has better average best values than ABC and NABC. The least standard deviation of MABC to six test functions with the dimensions $D = 20$ and $D = 40$ is $8.89001e - 18$, and the standard deviations of MABC are always less than DEABC; meanwhile the standard deviation of DEABC is less than ABC and NABC.

Hence, by the results of six test functions with the dimensions $D = 20$ and $D = 40$, MABC have better results than ABC, NABC, and DEABC. For all test functions, the mean and standard deviation of MABC results are several orders of magnitude better than ABC, NABC, and DEABC, particularly in case of SumSquares, Schwefel 2.22, and Ackley functions. Therefore, MABC has more accurate solution and better stability than ABC, NABC, and DEABC.

4.3. Comparison of Total Run Time. This subsection will show effectively that the searching speed of MABC is fastest through the phenomenon for total run time of solutions (see Figure 7). To all the functions, when $D = 20$ or 40, NABC costs the more time, ABC and DEABC follow closely, MABC

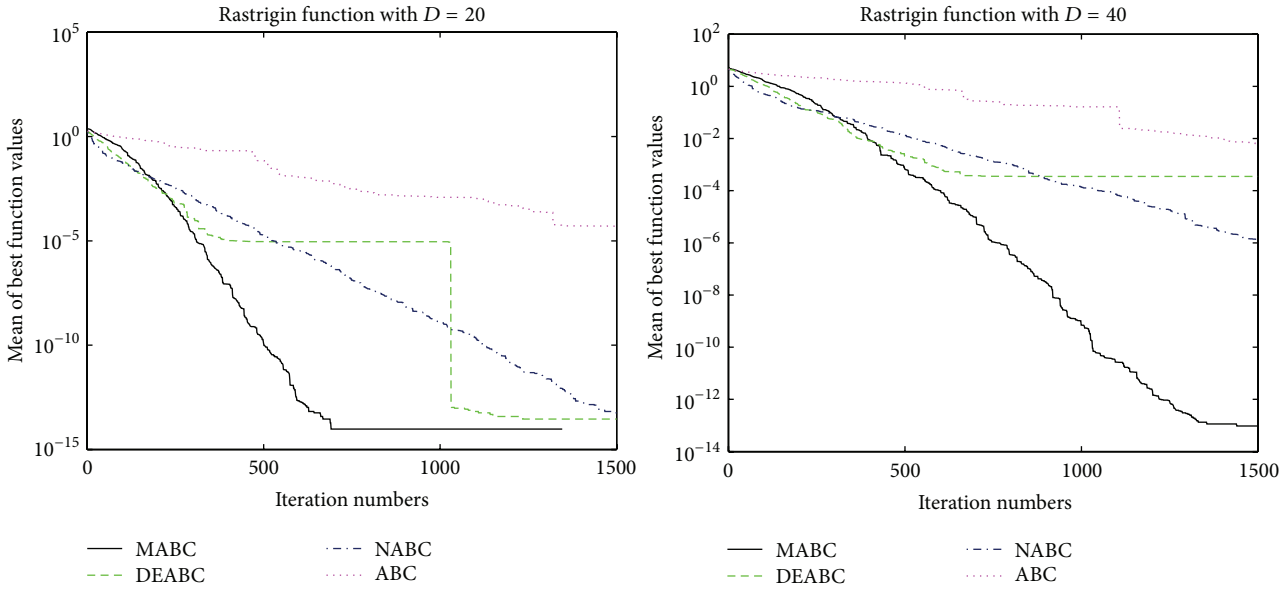


Figure 5: Convergence performance of ABC, DEABC, NABC, and MABC.

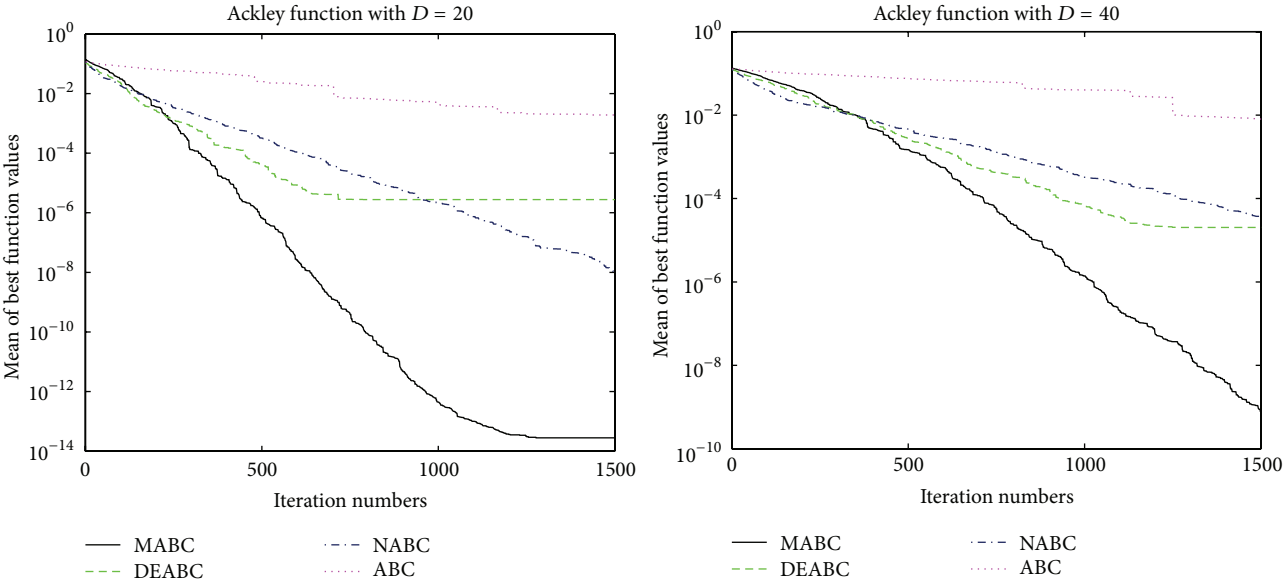


Figure 6: Convergence performance of ABC, DEABC, NABC, and MABC.

spends time the least, and the time of search solution of MABC is independent of dimension sizes. And the Sphere function costs least computational time, while the Rastrigin function takes the maximum one, so it shows that the finding solution time is related to the function types. Further, the time of search solution of MABC is less than that of any one of ABC, NABC, and DEABC.

Moreover, based on the six benchmark functions with the dimension $D = 20$ or 40, Table 5 displays all of the reduction rates, and the reduction rate of total iteration time is at least 73.52% and up to 85.88% for MABC to DEABC, MABC to NABC, and MABC to ABC.

5. Concluding Remarks

In this paper, to improve classical artificial bee colony algorithms, a new modified artificial bee colony (MABC) algorithm was developed, which is poor at exploitation and has some defects such as slow search speed, poor population diversity, the stagnation in the working process, and being trapped into the local optimal solution.

In order to make the initial population diversity distribute evenly and to take full use of the search space information, the initial population was constructed by using the opposition-based learning theory. Next, a new S-type method of grouping

Table 5: Reduction rate of total iteration time.

Function	D	MABC to DEABC	MABC to NABC	MABC to ABC
Sphere	20	85.38%	85.88%	85.43%
	40	85.11%	85.83%	85.19%
SumSquares	20	81.08%	81.41%	81.17%
	40	80.59%	81.38%	80.77%
Schwefel 2.22	20	84.26%	84.64%	84.42%
	40	83.53%	84.50%	83.74%
Griewank	20	77.81%	78.22%	77.83%
	40	75.39%	76.19%	73.52%
Rastrigin	20	83.80%	83.96%	83.84%
	40	82.23%	83.28%	82.35%
Ackley	20	82.26%	82.80%	82.48%
	40	81.20%	81.93%	81.07%

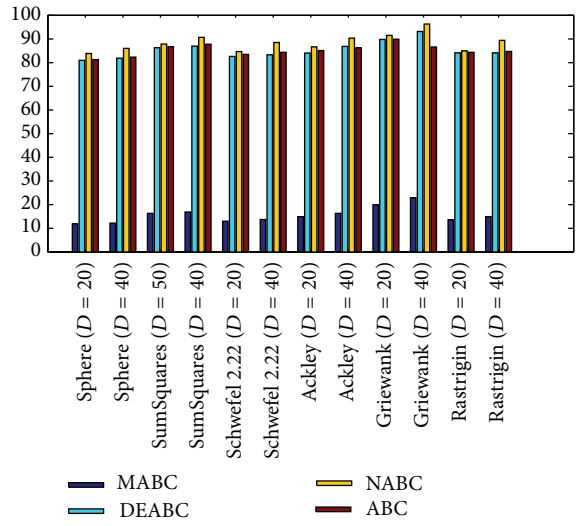


Figure 7: Total run time comparison of ABC, NABC, DEABC, and MABC.

was applied to reduce the difference between each group and replaced a greedy method (i.e., the roulette wheel way) by free search algorithm-sensitivity with pheromone. Then, a suitable exponential function updating length step was designed for replacing the original random step and carrying out faster and more stable searching of the global optimal solution.

Finally, from the new test function versions CEC13, six benchmark functions with the dimensions $D = 20$ and $D = 40$ were chosen and applied in the experiments for analyzing and comparing the iteration speed and accuracy of MABC to differential evolution artificial bee colony algorithm (DEABC), novel artificial bee colony algorithm (NABC), and basic artificial bee colony algorithm (ABC). With respect to the six benchmark functions with the dimensions $D = 20$ and $D = 40$, respectively, one can see from the experimental results in Tables 4 and 5 that the smallest standard deviation of the new modified algorithm is $8.89001e-18$, and the reduction rate of total iteration time for the new modified algorithm relative to the other three proposed artificial bee colony algorithms is at least 73.52% and up to 85.88%. Moreover, the experimental results presented in Figures 1–7 show that MABC has faster and more stable searching and can increase poor population diversity and avoid falling into the local optimal solutions.

MABC is superior to ABC, NABC, and DEABC, and the feasibility and effectiveness of MABC are very easy to be implemented. However, some other state-of-the-art algorithms may be selected as comparable algorithms, and more complex test functions and more diverse set of benchmark functions may be implemented to validate the benefits of MABC. Thus, the following works as *open questions* will be worth further studying:

(i) Some applications of MABC in the areas of pattern classification, fuzzy control, nonlinear system modeling, parameter tuning of proportional-integral-derivative controller, and so on.

(ii) Selecting more state-of-the-art algorithms as comparable algorithms.

(iii) Simulating more diverse set of benchmark functions for corresponding swarm intelligence algorithms.

(iv) Implementing more complex functions for validating the benefits of MABC and other corresponding improving swarm intelligence algorithms.

Conflict of Interests

The authors declare that they have no competing interests regarding the publication of this paper.

Acknowledgments

This work was partially supported by Sichuan Province Cultivation Fund Project of Academic and Technical Leaders and the Open Research Fund of Key Laboratory of Higher

Education of Sichuan Province for Enterprise Informationalization and Internet of Things (2013WZJ01) and the Scientific Research Project of Sichuan University of Science & Engineering (2015RC07) and cofinanced by National Natural Science Foundation of China (61573010 and 11501391).

References

[1] W. L. Xiang, S. F. Ma, and M. Q. An, "Habcde: a hybrid evolutionary algorithm based on artificial bee colony algorithm and differential evolution," *Applied Mathematics and Computation*, vol. 238, pp. 370–386, 2014.

[2] H.-C. Tsai, "Integrating the artificial bee colony and bees algorithm to face constrained optimization problems," *Information Sciences*, vol. 258, pp. 80–93, 2014.

[3] M. F. Tasgetiren, Q.-K. Pan, P. N. Suganthan, and A. Oner, "A discrete artificial bee colony algorithm for the no-idle permutation flowshop scheduling problem with the total tardiness criterion," *Applied Mathematical Modelling*, vol. 37, no. 10-11, pp. 6758–6779, 2013.

[4] K. S. Tang, K. F. Man, S. Kwong, and Q. He, "Genetic algorithms and their applications," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 22–37, 1996.

[5] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., pp. 760–766, Springer US, 2010.

[6] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[7] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[8] S. Zhang and S. Liu, "A novel artificial bee colony algorithm for function optimization," *Mathematical Problems in Engineering*, vol. 2015, Article ID 129271, 10 pages, 2015.

[9] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical Report tr06, Computer Engineering Department, Engineering Faculty, Erciyes University, Kayseri, Turkey, 2005.

[10] M. El-Abd, "Performance assessment of foraging algorithms vs. evolutionary algorithms," *Information Sciences*, vol. 182, no. 1, pp. 243–263, 2012.

[11] T. Y. Lim, M. A. Al-Betar, and A. T. Khader, "Adaptive pair bonds in genetic algorithm: an application to real-parameter optimization," *Applied Mathematics and Computation*, vol. 252, pp. 503–519, 2015.

[12] Y. Xiang, Y. Peng, Y. Zhong, Z. Chen, X. Lu, and X. Zhong, "A particle swarm inspired multi-elitist artificial bee colony algorithm for real-parameter optimization," *Computational Optimization and Applications*, vol. 57, no. 2, pp. 493–516, 2014.

[13] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, vol. 42, no. 1, pp. 21–57, 2014.

[14] J. C. Bansal, H. Sharma, and S. S. Jadon, "Artificial bee colony algorithm: a survey," *International Journal of Advanced Intelligence Paradigms*, vol. 5, no. 1-2, pp. 123–159, 2013.

[15] A. L. Bolaji, A. T. Khader, M. A. Al-Betar, and M. A. Awadallah, "Artificial bee colony algorithm, its variants and applications: a survey," *Journal of Theoretical and Applied Information Technology*, vol. 47, no. 2, pp. 434–459, 2013.

[16] H. Habbi, Y. Boudouaoui, D. Karaboga, and C. Ozturk, "Self-generated fuzzy systems design using artificial bee colony optimization," *Information Sciences*, vol. 295, pp. 145–159, 2015.

[17] L. N. Vitorino, S. F. Ribeiro, and C. J. A. Bastos-Filho, "A mechanism based on artificial bee colony to generate diversity in particle swarm optimization," *Neurocomputing*, vol. 148, pp. 39–45, 2015.

[18] M. S. Kiran and O. Findik, "A directed artificial bee colony algorithm," *Applied Soft Computing Journal*, vol. 26, pp. 454–462, 2015.

[19] D. Karaboga and B. Gorkemli, "A quick artificial bee colony (qABC) algorithm and its performance on optimization problems," *Applied Soft Computing Journal*, vol. 23, pp. 227–238, 2014.

[20] N. Imanian, M. E. Shiri, and P. Moradi, "Velocity based artificial bee colony algorithm for high dimensional continuous optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 36, pp. 148–163, 2014.

[21] H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, and J.-s. Pan, "Multi-strategy ensemble artificial bee colony algorithm," *Information Sciences*, vol. 279, pp. 587–603, 2014.

[22] W.-F. Gao, S.-Y. Liu, and L.-L. Huang, "Enhancing artificial bee colony algorithm using more information-based search equations," *Information Sciences*, vol. 270, pp. 112–133, 2014.

[23] S. Das, S. Biswas, and S. Kundu, "Synergizing fitness learning with proximity-based food source selection in artificial bee colony algorithm for numerical optimization," *Applied Soft Computing*, vol. 13, no. 12, pp. 4676–4694, 2013.

[24] M. X. Zang, X. Ma, and Y. M. Duan, "Improved artificial bee colony algorithm," *Journal of Xidian University*, vol. 42, no. 2, pp. 65–70, 2015.

[25] C. Zhang, J. Zheng, and Y. Zhou, "Two modified artificial bee colony algorithms inspired by grenade explosion method," *Neurocomputing*, vol. 151, no. 3, pp. 1198–1207, 2015.

[26] M. Mernik, S.-H. Liu, D. Karaboga, and M. Črepinšek, "On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation," *Information Sciences*, vol. 291, pp. 115–127, 2015.

[27] M. S. Kiran, H. Hakli, M. Gunduz, and H. Uguz, "Artificial bee colony algorithm with variable search strategy for continuous optimization," *Information Sciences*, vol. 300, pp. 140–157, 2015.

[28] T. G. Chen and R. B. Xiao, "Modeling design iteration in product design and development and its solution by a novel artificial bee colony algorithm," *Computational Intelligence and Neuroscience*, vol. 2014, Article ID 240828, 13 pages, 2014.

[29] B. Li, "Research on WNN modeling for gold price forecasting based on improved artificial bee colony algorithm," *Computational Intelligence and Neuroscience*, vol. 2014, Article ID 270658, 10 pages, 2014.

[30] S. Wei and N. Z. Lu, "An improvement for artificial bee algorithm," *Modern Computer*, vol. 6, no. 17, pp. 25–29, 2014.

[31] L. Li, F. M. Yao, L. J. Tan et al., "A novel DE-ABC-based hybrid algorithm for global optimization," in *Proceedings of the 7th International Conference on Intelligent Computing (ICIC '11), Zhengzhou, China, August 2011*, D. S. Huang, Y. Gan, P. Premaratne, and K. Han, Eds., Lecture Notes in Computer Science, pp. 558–565, Springer, Berlin, Germany, 2012.

[32] S.-M. Chen, A. Sarosh, and Y.-F. Dong, "Simulated annealing based artificial bee colony algorithm for global numerical

optimization," *Applied Mathematics and Computation*, vol. 219, no. 8, pp. 3575–3589, 2012.

[33] S. C. Satapathy and A. Naik, "Modified teaching-learning-based optimization algorithm for global numerical optimization—a comparative study," *Swarm and Evolutionary Computation*, vol. 16, pp. 28–37, 2014.

[34] Q. Xu, L. Wang, N. Wang, X. Hei, and L. Zhao, "A review of opposition-based learning from 2005 to 2012," *Engineering Applications of Artificial Intelligence*, vol. 29, pp. 1–12, 2014.

[35] K. Penev, "Free search—comparative analysis 100," *International Journal of Metaheuristics*, vol. 3, no. 2, pp. 118–132, 2014.

Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 3629174, 12 pages
http://dx.doi.org/10.1155/2016/3629174

Research Article

A Multiobjective Approach to Homography Estimation

Valentín Osuna-Enciso,[1] Erik Cuevas,[2] Diego Oliva,[2,3] Virgilio Zúñiga,[1] Marco Pérez-Cisneros,[1] and Daniel Zaldívar[2]

[1]*Sciences Division, Centro Universitario de Tonalá of Universidad de Guadalajara, 45400 Guadalajara, JAL, Mexico*
[2]*Electronic Division, Centro Universitario de Ciencias Exactas e Ingenierías of Universidad de Guadalajara, 44430 Guadalajara, JAL, Mexico*
[3]*Computer Sciences Department, Tecnológico de Monterrey Campus Guadalajara, 45201 Guadalajara, JAL, Mexico*

Correspondence should be addressed to Valentín Osuna-Enciso; valentin.osuna@cutonala.udg.mx

Received 4 June 2015; Accepted 4 October 2015

Academic Editor: Yufeng Zheng

In several machine vision problems, a relevant issue is the estimation of homographies between two different perspectives that hold an extensive set of abnormal data. A method to find such estimation is the random sampling consensus (RANSAC); in this, the goal is to maximize the number of matching points given a permissible error (Pe), according to a candidate model. However, those objectives are in conflict: a low Pe value increases the accuracy of the model but degrades its generalization ability that refers to the number of matching points that tolerate noisy data, whereas a high Pe value improves the noise tolerance of the model but adversely drives the process to false detections. This work considers the estimation process as a multiobjective optimization problem that seeks to maximize the number of matching points whereas Pe is simultaneously minimized. In order to solve the multiobjective formulation, two different evolutionary algorithms have been explored: the Nondominated Sorting Genetic Algorithm II (NSGA-II) and the Nondominated Sorting Differential Evolution (NSDE). Results considering acknowledged quality measures among original and transformed images over a well-known image benchmark show superior performance of the proposal than Random Sample Consensus algorithm.

1. Introduction

A homography is a transformation that maps points of interest by considering movements as translation, rotation, skewing, scaling, and projection among image planes, all of them contained into a single, invertible matrix. In general terms, those displacements could be considered to be belonging to three cases: (1) an object moving in front of a static camera, (2) a static scene captured by a moving camera, and (3) multiple cameras from different viewpoints. In either case, those approximations simplify the utilization of image sequences to construct panoramic views [1–3], to increment resolution in low quality imagery [4–6], to remove camera movements when studying the motion of an object into a video [7], and to control the position of robots [8–10], among other uses [11–13].

Taking a set of experimental data as a base, in a modeling problem there exist two data types: those that can be adjusted to a model with a certain probability (also known as inliers) and those that are not related to the model (e.g., outliers). There are several algorithms specialized in solving this classification problem; one of such techniques is the Random Sample Consensus (RANSAC) [14].

In the algorithm, minimum subsets of experimental data are randomly taken, and a model is proposed and evaluated according to a permissible error (Pe), in order to determine how well the model adjusts to the data [15]. This process is repeated until a number of iterations are completed, and the model with the maximum number of inliers is taken.

Even considering that RANSAC is a robust and simple algorithm, it has some drawbacks [16–18], two of which are the high dependency between the number of matching points (model quality) and the permissible error. In this work, it is considered that those disadvantages belong to a multiobjective optimization problem. On the one hand, due to the random nature of RANSAC, achieving improvements

in the quantity of inliers implies more iterations in order to discard unadjusted data to the proposed model. On the other hand, the number of matching points conflictingly depends on the permissible error (Pe). A low Pe value increases the accuracy of the model but degrades its generalization ability to tolerate noisy data (number of matching points). By contrast, a high Pe value improves the noise tolerance of the model but adversely drives the process to false detections. The main error source in the model estimation procedure arises from defining the Pe value with no consideration of the relationship between the dataset and the model.

In order to make the RANSAC algorithm more efficient, some improvements have been suggested; for instance, in the algorithm called MLESAC [19] it is considered that the inliers into the images will follow a Gaussian distribution whereas the outliers are considered as uniformly positioned; according to that, the voting process is achieved through maximizing the likelihood and the original RANSAC. The SIMFIT method [20] proposed the forecasting of the permissible error, through an iterative reestimation of that value, until the model is adjusted to the experimental data. Some other variants to the original RANSAC are the projection-pursuit method, the Two-Step Scale Estimator, and the CC-RANSAC [15, 21, 22], all of them focused on maximizing the number of inliers by making more searches into the data and therefore making the complete process more expensive, computationally speaking. In such sense, an algorithm that tries to reduce the computational cost is the one proposed in [17], where the maximization of the inliers is achieved by using a metaheuristic technique, called Harmony Search.

Nevertheless the mentioned improvements, the search strategy used in the mentioned articles (with exception of [17]), are more close to random walking, and therefore those approaches are computationally expensive. Moreover, in all the cases only one objective function is considered, usually related to the number of matching points, while the permissible error is left behind. In accordance with that, and in order to overcome the typical RANSAC problems, we propose to visualize the RANSAC operation as a multiobjective problem solved by an evolutionary algorithm. Under such point of view, at each iteration, new candidate solutions are built by using evolutionary operators taking into account the quality of the previously generated models, rather than purely random, reducing significantly the number of iterations. Likewise, new objective functions can be added to incorporate other elements that allow an accurate evaluation of the quality of a candidate model.

When an optimization problem involves more than one objective function, the procedure of finding one or more optimum solutions is known as multiobjective optimization (MO) [23]. Under MO, different solutions produce conflicting scenarios among the objectives [24]. Contrary to single objective optimization, in MO it is usually difficult to find one optimal solution. Instead, algorithms for optimizing multiobjective problems try to find a family of points known as the Pareto optimal set [25]. These points verify that there is no different feasible solution which strictly improves one component of the objective function vector without worsening at least one of the remaining ones. Evolutionary algorithms (EAs) are considered the most adequate methods for solving complex MO problems, due mainly they are many times capable of maintaining a good diversity [26], can extend to multiple populations [27], as well as can work with a variety of problems such as discrete ones [28]. Several variants of nondominated sorting as well as new methods have been proposed in recent years in order to solve problems related to feature selection [29], community detection [30], among other issues [24, 31]; however, the Nondominated Sorting Genetic Algorithm II (NSGA-II) [32] and the Nondominated Sorting Differential Evolution (NSDE) [31] are some of the most representative.

In this paper, the estimation process is considered as a multiobjective problem where the number of matching points and the permissible error (Pe) are simultaneously optimized. In order to solve the multiobjective formulation, two different evolutionary algorithms have been explored: the Nondominated Sorting Genetic Algorithm II (NSGA-II) and the Nondominated Sorting Differential Evolution (NSDE). Results considering acknowledged quality measures among original and transformed images over a well-known image benchmark show superior performance of the proposal than Random Sample Consensus algorithm on the problem being assessed, giving good results even with high outliers levels.

The remainder of the paper is organized as follows: Section 2 explains the problem of image homography considering multiple views. Section 3 introduces the fundamentals of the RANSAC method. Section 4 briefly explains the evolutionary approaches that are used in this paper in order to solve the multiobjective problem while Section 5 presents the proposed method. Section 6 exhibits the experimental set and its performance. Finally, Section 7 establishes some final conclusions.

2. Homography between Images

For the case where two images are taken of the same scene from different perspectives, a problem consists in finding a transformation that permits the matching among the pixels belonging to both images. This denominated the image matching problem. The search of a geometric transformation is achieved by utilizing corresponding points from image pairs [33, 34], which enable forming feature vectors, also called image descriptors. Even when considering that such descriptors are not completely reliable, so they can produce erroneous results for the image matching, in this paper they are used to find the geometric relations between images by using the homography, which is explained in the next paragraphs.

Consider a set of points

$$\mathbf{U} = \left\{\left(\mathbf{x}_1, \mathbf{x}_1'\right), \left(\mathbf{x}_2, \mathbf{x}_2'\right), \ldots, \left(\mathbf{x}_M, \mathbf{x}_M'\right)\right\}, \tag{1}$$

such that $\mathbf{x}_i = (x_i, y_i, 1)^T$ and $\mathbf{x}_i' = (x_i', y_i', 1)^T$ are the positions with respect to a given image pair.

By means of a Q plane, a homography $\mathbf{H}$ establishes a geometric relation between two images taken under different perspectives, as can be seen in Figure 1; this allows for a projection of the points from the plane to a pair of images,

(1) for $i = 1$ through G
(2) Randomly select a subsample from $\mathbf{S_i} \subset \mathbf{G} \subseteq \mathbf{U}$, and assemble a sample $\mathbf{S}_i$
(3) According to $\mathbf{S}_i$, compose the candidate hypothesis h_i
(4) Calculate the degree of agreement $A_i(\mathbf{U}, h_i)$

Pseudocode 1

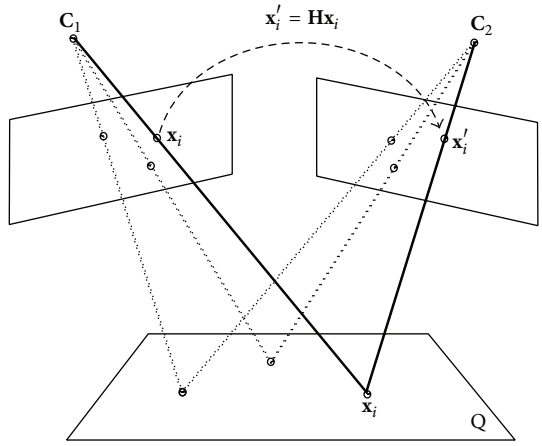


Figure 1: Homography from a plane between two views.

through $\mathbf{x}_i' = \mathbf{H}\mathbf{x}_i$ or $\mathbf{x}_i = \mathbf{H}^{-1}\mathbf{x}_i'$. Conducive to find the homography between an image pair, a set with four point matches is only required, to construct a linear system which must be solved [35]. Concerning evaluation of the quality of the candidate homography, it is necessary to calculate the distance among the point positions of the first image with respect to the second image; that distance is labeled as the Mismatch Error and is defined by

$$EH_i^2 = \left[d\left(\mathbf{x}_i', \mathbf{H}\mathbf{x}_i\right)\right]^2 + \left[d\left(\mathbf{x}_i, \mathbf{H}^{-1}\mathbf{x}_i'\right)\right]^2 \tag{2}$$

as long as $\eta = d(\mathbf{x}_i, \mathbf{H}^{-1}\mathbf{x}_i')$ and $\eta' = d(\mathbf{x}_i', \mathbf{H}\mathbf{x}_i)$ are the respective errors from each image.

Consider the example shown in Figure 2, where five correspondences $\mathbf{U} = \{(\mathbf{x}_1, \mathbf{x}_1'), \ldots, (\mathbf{x}_5, \mathbf{x}_5')\}$ are depicted; for the case of the points $(\mathbf{x}_3, \mathbf{x}_3')$, the error (η, η') is considerable, and therefore the quality of the candidate homography will be ranked with a low value.

3. Random Sampling Consensus (RANSAC) Algorithm

To find correspondences from images through a geometric transformation (homography) and therefore to increase the number of correct matches (inliers), the use of a robust approach, such as RANSAC, is necessary. Contrary to the inliers, outliers are conflicting points related to the candidate homography.

The idea behind the algorithm consists in discovering the best hypothesis h^B from a set of hypotheses H generated by the source data, usually corrupted with noise. The construction of candidate hypothesis h_i is achieved by means of a sample $\mathbf{S}_i$, with a minimum size s, to model estimation. As in this paper $s = 4$, then several $\mathbf{S}_i$ could be taken from the complete source data $\mathbf{U}$, and, therefore, an exhaustive search would be computationally expensive.

In Pseudocode 1 the basic pseudocode of the RANSAC algorithms family is shown.

A subsequent step consists in finding the best candidate hypothesis h^B from all the constructed and evaluated hypotheses, according to

$$h^B = \underset{i=1,\ldots,G}{\arg\max}\, A_i\left(\mathbf{U}, h_i\right). \tag{3}$$

The degree of agreement is directly related to the number of inliers, and it is calculated by

$$A_i\left(\mathbf{U}, h_i\right) = \sum_{j=1}^{M} \theta\left(e_j^2\left(h_i\right)\right), \quad j = 1, \ldots, M,$$
$$\theta\left(e_j^2\left(h_i\right)\right) = \begin{cases} 0, & e_j^2\left(h_i\right) > \text{Pe}, \\ 1, & e_j^2\left(h_i\right) \leq \text{Pe}, \end{cases} \tag{4}$$

where Pe is a permissible error, M is the number of elements contained in the source data $\mathbf{U}$, and $e_j^2(h_i) = EH_i^2$ is the quadratic error produced by the jth data considering the hypothesis h_i; in other words, it represents the error produced by the ith correspondence.

In original RANSAC algorithm, the best hypothesis is the one with the maximum number of inliers. The point j which produces an error $e_j^2(h_i)$ lesser than a permissible error Pe is considered as an inlier of candidate hypothesis h_i; otherwise, it is considered as an outlier.

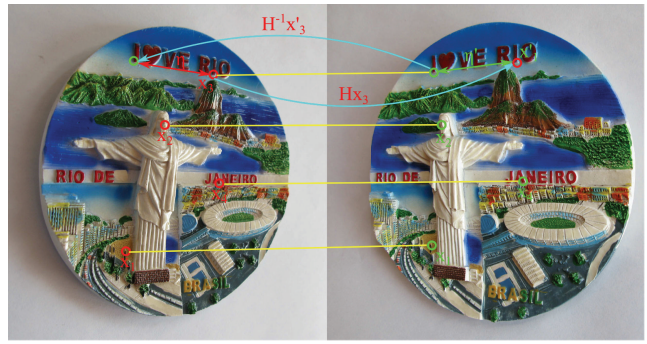


Figure 2: Example of evaluation process for a particular homography **H**.

The RANSAC technique has to search the entire source data **U** at least once in the worst case; by considering such situation, the algorithm is similar to random walking. Several strategies could improve that kind of search, like evolutionary algorithms (EAs) [36]. These techniques are capable of exploitation and exploration of the search space judiciously, by considering that new candidate solutions will contain information regarding the best spots from search space, visited through each generation.

This work proposes working the estimation process as a multiobjective problem, by simultaneously optimizing both the number of matching points and the permissible error (Pe). In order to solve the multiobjective formulation, two different evolutionary algorithms have been explored: the Nondominated Sorting Genetic Algorithm II (NSGA-II) and the Nondominated Sorting Differential Evolution (NSDE). With the formulation, the proposed method adopts a different sampling strategy than RANSAC to generate putative solutions. Under the new mechanism, at every iteration new candidate solutions are generated based on the quality of previously found solutions, avoiding random walks in the searching process, as in the case of RANSAC.

4. Multiobjective Evolutionary Algorithms

A MO problem can be stated as minimizing or maximizing the function [37]

$$\begin{aligned} \mathbf{f}(\mathbf{x}) &= \left[f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\right], \\ \text{Subject to} \quad & g_j(\mathbf{x}) \geq 0, \quad j = 1, 2, \ldots, J, \\ & h_k(\mathbf{x}) = 0, \quad k = 1, 2, \ldots, K, \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, \quad i = 1, 2, \ldots, n, \end{aligned} \tag{5}$$

where a solution $\mathbf{x}$ is a vector of n decision variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. The last set of constraints is called variable bounds, restricting each decision variable x_i to take a value within a lower $x_i^{(L)}$ and upper $x_i^{(U)}$ bound and whose limits constitute a decision space D. There are J inequalities and K equality constraints, both associated with the problem. In order to cover both minimization and maximization of objective functions, the operator $\lhd$ is used between two solutions $\mathbf{u}$ and $\mathbf{v}$. Therefore, $\mathbf{u} \lhd \mathbf{v}$ denotes that solution $\mathbf{u}$ is better than solution $\mathbf{v}$ whereas $\mathbf{u} \trianglelefteq \mathbf{v}$ implies that solution $\mathbf{u}$ is better than or equal to solution $\mathbf{v}$.

Different from single objective optimization, in the case of multiobjective optimization, it is usually difficult to find one optimal solution. Instead, algorithms for optimizing multiobjective problems attempt to find a group of points known as the Pareto optimal set [38]. These points verify that there is no other feasible solution which strictly improves one component of the objective function vector without worsening at least one of the remaining ones. A more formal definition of Pareto optimality or Pareto efficiency is the following.

Definition 1. If, given a solution $\mathbf{u}$, there exists another solution $\mathbf{v}$ such that $\forall p = 1, \ldots, M$ $f_p(\mathbf{u}) \trianglelefteq f_p(\mathbf{v})$ and $\exists p \in \{1, \ldots, M\}$ such that $f_p(\mathbf{u}) \lhd f_p(\mathbf{v})$, then one will say that solution $\mathbf{u}$ *dominates* solution $\mathbf{v}$ (denoted by $\mathbf{u} \prec \mathbf{v}$), and, obviously, solution $\mathbf{v}$ will never be sensibly selected as the solution to the problem. If $f_p(\mathbf{u}) \trianglelefteq f_p(\mathbf{v})$, $\forall p$, one will say that solution $\mathbf{u}$ weakly dominates solution $\mathbf{v}$ and will be denoted by $\mathbf{u} \preceq \mathbf{v}$.

Definition 2. A solution $\mathbf{u} \in D$ is considered to be part of the Pareto optimal set F if and only if $\nexists \mathbf{v} \in D$ such that $\mathbf{v} \prec \mathbf{u}$.

Evolutionary algorithms (EAs) are considered the most adequate methods for solving complex MO problems and some have been proposed to face such problems, where the Nondominated Sorting Genetic Algorithm II (NSGA-II) and the Nondominated Sorting Differential Evolution (NSDE) are some of the most representative.

4.1. Nondominated Sorting Genetic Algorithm II (NSGA-II). NSGA-II, introduced by Deb et al. [32], is one of the most applicable and employed algorithms based on GA to solve multiobjective optimization problems. NSGA-II starts randomly generating an initial ($t = 0$) parent population P^t of size N. During several consecutive generations ($t = 1, \ldots,$ maxIterations), the M objective values of P^t are evaluated. Then, the population is ranked based on the nondomination sorting procedure to create Pareto optimal fronts F. Each individual of the population under evaluation obtains a rank equal to its nondomination level (1 is the best level, 2 is the next-best level, and so on), where the first

front contains individuals with the best rank, the second front corresponds to the individuals with the second rank, and so on. In the next step, the crowding distance between members of each front is calculated by a linear distance criterion. As a binary tournament selection operator based on a crowded-comparison operator is used, it is necessary to calculate both the rank and the crowding distance of each member in the population. Using this selection operator, two members are selected among the population. Then, the member with the larger crowding distance is selected if they share an equal rank. Otherwise, the member with the lower rank is chosen. Next, a new population of offspring with a size of N is created using the random selection, the simulated binary crossover [19], and the polynomial mutation [20] operators to create a population consisting of the current and the new population of the size of $2N$.

4.1.1. Simulated Binary Crossover. This operator simulates the behavior of the single-point crossover on binary strings. Given as parents $\mathbf{x}^{(1,t)}$ and $\mathbf{x}^{(2,t)}$, they generate the ith component ($i = 1, 2, \ldots, n$) of the offspring individuals as follows:

$$x_i^{(1,t+1)} = 0.5 \cdot \left[\left(1+\beta_i\right) \cdot x_i^{(1,t)} + \left(1-\beta_i\right) \cdot x_i^{(2,t)}\right],$$
$$x_i^{(2,t+1)} = 0.5 \cdot \left[\left(1-\beta_i\right) \cdot x_i^{(1,t)} + \left(1+\beta_i\right) \cdot x_i^{(2,t)}\right],$$
$$\beta_i = \begin{cases} (2 \cdot a)^{1/(n_c+1)}, & \text{if } a < 0.5, \\ \left(\dfrac{1}{2 \cdot (1-a)}\right)^{1/(n_c+1)}, & \text{otherwise,} \end{cases} \tag{6}$$

where a is a random number in $[0, 1]$. The parameter n_c determines the separation between the offspring individuals in comparison to their parents.

4.1.2. Polynomial Mutation. This operator employs a polynomial distribution in the following way:

$$x_i^{(t+1)} = x_i^t + \left(x_i^{(U)} - x_i^{(L)}\right) \cdot \delta_i,$$
$$\delta_i = \begin{cases} (2 \cdot a)^{1/(n_m+1)} - 1, & \text{if } a < 0.5, \\ 1 - |2 \cdot (1-a)|^{1/(n_m+1)}, & \text{otherwise,} \end{cases} \tag{7}$$

where $x_i^{(L)}$ and $x_i^{(U)}$ are the low and upper bounds, respectively, for the i decision variable, whereas n_m represents the distribution index.

4.2. Nondominated Sorting Differential Evolution (NSDE). The NSDE [31] algorithm is an extension of the original differential evolution (DE) [39] method for solving multiobjective problems. NSDE works in a similar way to DE except in the selection operation which is modified in order to be coherent with the nondominated criterion.

The algorithm begins by initializing a population of n-dimensional individuals and considers parameter values that are randomly distributed between the prespecified lower initial parameter bound $x_i^{(L)}$ and the upper initial parameter bound $x_i^{(U)}$. In order to generate a trial individual (solution), the DE algorithm first mutates the current individual $\mathbf{x}_{i,t}$ from the population by adding the scaled difference of two vectors from the current population:

$$\mathbf{v}_{i,t} = \mathbf{x}_{i,t} + F \cdot \left(\mathbf{x}_{r_1,t} - \mathbf{x}_{r_2,t}\right); \quad r_1, r_2 \in \left\{1, 2, \ldots, N_p\right\}, \tag{8}$$

with $\mathbf{v}_{i,t}$ being the mutant individual. Indexes r_1 and r_2 are randomly selected with the condition that they are different and have no relation to the individual index i whatsoever (i.e., $r_1 \neq r_2 \neq i$). The mutation scale factor F is a positive real number, typically less than one. In order to increase the diversity of the parameter element, the crossover operation is applied between the mutant individual $\mathbf{v}_{i,t}$ and the original individuals $\mathbf{x}_{i,t}$. The result is the trial individual $\mathbf{u}_{i,t}$ which is computed by considering an element to element operation as follows:

$$u_{j,i,t} = \begin{cases} v_{j,i,t}, & \text{if rand}\,(0,1) \leq \text{CR or } j = j_{\text{rand}}, \\ x_{j,i,t}, & \text{otherwise,} \end{cases} \tag{9}$$

where $j_{\text{rand}} \in \{1, 2, \ldots, D\}$. The subscripts j and i are the parameter and individual indexes, respectively. The crossover parameter ($0.0 \leq \text{CR} \leq 1.0$) controls the fraction of parameters where the mutant individual is contributing to the final trial individual. In addition, the trial individual always inherits the mutant individual parameter according to the randomly chosen index j_{rand}, assuring that the trial individual differs by at least one parameter from $\mathbf{x}_{i,t}$. Finally, a nondominated selection is used to build the Pareto optimal front. Thus, if the trial individual $\mathbf{x}_{i,t}$ dominates the target individual $\mathbf{x}_{i,t}$, the trial individual $\mathbf{x}_{i,t}$ is copied into the population for the next generation; otherwise, the target individual $\mathbf{x}_{i,t}$ is copied:

$$\mathbf{x}_{i,t+1} = \begin{cases} \mathbf{u}_{i,t}, & \text{if } \mathbf{u}_{i,t} \prec \mathbf{x}_{i,t}, \\ \mathbf{x}_{i,t}, & \text{otherwise.} \end{cases} \tag{10}$$

5. The Proposed Approach

5.1. Individual Representation. In the estimation process, each candidate homography $\mathbf{H}_i$ is calculated by using four different point correspondences. The candidate homography $\mathbf{H}_i$ is thus evaluated over the entire dataset $\mathbf{U}$, dividing all elements from the dataset to inliers and outliers, according to a permissible error (Pe).

In order to construct a candidate solution or individual $\mathbf{s}_i$, four indexes, o, p, q, and r, are selected from the set $\{1, 2, \ldots, M\}$ of correspondences. Therefore, the homography $\mathbf{H}_i$ across the two views is computed by solving the linear system produced from the set of four point matches $(\mathbf{x}_o, \mathbf{x}'_o)$, $(\mathbf{x}_p, \mathbf{x}'_p)$, $(\mathbf{x}_q, \mathbf{x}'_q)$, and $(\mathbf{x}_r, \mathbf{x}'_r)$. Additionally, the permissible error Pe that is associated with the individual $\mathbf{s}_i$ is incorporated as a decision variable. Thus, in the proposed algorithm, an individual or candidate solution $\mathbf{s}_i$ is coded as a vector of five decision variables ($\mathbf{s}_i = \{s_i^1, s_i^2, s_i^3, s_i^4, s_i^5\}$) that is defined by

$$\mathbf{s}_i = [o, p, q, r, \text{Pe}]. \tag{11}$$

In our approach, the candidate solution $\mathbf{s}_i$ presents the same functionality, that is, hypothesis h_i in the original RANSAC algorithm.

5.2. Multiobjective Problem Formulation. In the proposed approach, the estimation process is considered as a multiobjective problem where the number of matching points and the permissible error are simultaneously optimized. Under such circumstances, the multiobjective problem can be defined as follows:

$$\begin{aligned}
\text{Maximize} \quad & f_1(\mathbf{s}_i) = \sum_{j=1}^{M} \theta\left(e_j^2(\mathbf{s}_i)\right), \\
\text{Minimize} \quad & f_2(\mathbf{s}_i) = \text{Pe}, \\
\text{Subject to} \quad & 1 \le o \le M, \\
& 1 \le p \le M, \\
& 1 \le q \le M, \\
& 1 \le r \le M, \\
& 0 \le \text{Pe} \le \text{Max}E,
\end{aligned} \tag{12}$$

considering that $e_j^2(\mathbf{s}_i) = [d(\mathbf{x}'_j, \mathbf{H}_i\mathbf{x}_j)]^2 + [d(\mathbf{x}_j, \mathbf{H}_i^{-1}\mathbf{x}'_j)]^2$ and $\theta(e_j^2(\mathbf{s}_i)) = 0$, $e_j^2(\mathbf{s}_i) > \text{Pe}$; 1, $e_j^2(\mathbf{s}_i) \le \text{Pe}$ and where Max$E$ represents the maximal commensurable error produced by a candidate homography. Although Max$E$ could be any high value, a sufficiently small value significantly reduces the search of Pareto fronts. In this work, Max$E$ has been set to 25.

5.3. Computational Procedures. In order to solve the multiobjective formulation, two different evolutionary algorithms have been explored: the Nondominated Sorting Genetic Algorithm II (NSGA-II) and the Nondominated Sorting Differential Evolution (NSDE). In this section, the computational procedure of both methods is described when they face the multiobjective problem described in (12).

(a) Nondominated Sorting Genetic Algorithm II (NSGA-II)

(1) Generate randomly a population P_t of N five-dimensional individuals. The decision variables of each individual $\mathbf{s}_i$ ($i \in \{1, \ldots, N\}$) are produced, considering a random number between their search bounds.

(2) Produce an offspring population Q_t from P_t by using simulated binary crossover and polynomial mutation.

(3) Create a new population R_t as the combination of P_t and Q_t ($R_t = P_t \cup Q_t$).

(4) Perform a nondominated sorting to R_t and identify different fronts: F_j, $j = 1, 2, \ldots$, and so forth.

(5) Set the new population $P_{t+1} = \varnothing$ and the counter $g = 1$.

(6) Perform $P_{t+1} = P_{t+1} \cup F_g$ and increment g ($g = g + 1$).

(7) Assuming that $|P_{t+1}|$ represents the number of elements contained in P_{t+1}, repeat step (6), until ($|P_{t+1}| + |F_g| < N$).

(8) Assuming that $w = |F_g|$, clear the initial distance ($d_z = 0$) of each element z from F_g.

(9) For each objective function $m = 1, 2, \ldots, M$, sort the set in worse order of f_m. Therefore, $\mathbf{I}^m = \text{sort}(f_m, >)$ contains the sorted elements of the objective function m.

(10) For $m = 1, 2, \ldots, M$, assign a large distance to the boundary elements of $\mathbf{I}^m$ ($d_{\mathbf{I}_1^m} = d_{\mathbf{I}_w^m} = \infty$). For all other elements $u = 2, 3, \ldots, w - 1$, assign a distance calculated as follows:

$$d_{\mathbf{I}_u^m} = d_{\mathbf{I}_u^m} + \frac{f_m(\mathbf{I}_{u+1}^m) - f_m(\mathbf{I}_{u-1}^m)}{f_m^{\max} - f_m^{\min}}, \tag{13}$$

where $\mathbf{I}_u^m$ represent the element u from the sorted set $\mathbf{I}^m$. $f_m^{\max}$ and $f_m^{\min}$ symbolize the maximum and minimum value of f_m.

(11) Select the ($N - |P_{t+1}|$) elements from F_g whose distances are the longest and include them in P_{t+1}.

(12) If the maximum number of iterations has been reached, the process is thus completed; otherwise, go back to step (2).

(13) The final population P_{t+1} contains the Pareto optimal set.

(b) Nondominated Sorting Differential Evolution (NSDE)

(1) Set the DE parameters $F = 0.25$ and CR = 0.8.

(2) Generate randomly a population P_t of N five-dimensional individuals. The decision variables of each individual $\mathbf{s}_i$ ($i \in \{1, \ldots, N\}$) are produced, considering a random number between their search bounds.

(3) Generate a trial population $\mathbf{T}$ with N individuals ($\mathbf{T} = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N\}$) of five dimensions ($\mathbf{t}_h = \{t_h^1, t_h^2, t_h^3, t_h^4, t_h^5\}$) under Procedure 1.

(4) Select the elements $\mathbf{s}_i$ ($i \in \{1, \ldots, N\}$) of the next population P_{t+1} under Procedure 2.

(5) If the maximum number of iterations has been reached, the process is thus completed; otherwise, go back to step (3).

(6) The final population P_{t+1} contains the Pareto optimal set.

6. Experimental Results

This part of the paper deals with several experiments performed over a collection of real images. The results exhibit the performance of NSGA-II and NSDE solving the estimation problem as a multiobjective optimization task in comparison to RANSAC. In the experiments, two performance indexes are considered: the mean squared error (MSE) and the

Table 1: Evaluation of the estimation results for NSGA-II and NSDE.

Algorithm	Image pair	MSE		PSNR	
NSGA-II	Figures 3(a) and 3(b)	97.44 (23.4474)	93.11 (19.9691)	8.59 (1.8793)	8.94 (1.6751)
NSDE		**82.64 (0.7574)**	**82.14 (0.0787)**	**9.82 (0.0791)**	**9.87 (0.0083)**
NSGA-II	Figures 3(c) and 3(d)	62.81 (20.6944)	56.09 (10.9807)	12.56 (2.4086)	13.31 (1.3887)
NSDE		**51.81 (0.1233)**	**51.87 (0.2008)**	**13.87 (0.0207)**	**13.86 (0.0336)**
NSGA-II	Figures 3(e) and 3(f)	118.23 (9.2986)	114.46 (0.4646)	6.73 (0.6693)	6.99 (0.0353)
NSDE		**114.56 (0.6316)**	**114.29 (0.2172)**	**6.98 (0.0480)**	**7.00 (0.0165)**

for ($i = 1; i < N + 1; i$++)
do $r_1 = \text{floor}(\text{rand}(0, 1) \cdot N)$; **while** ($r_1 = i$);
do $r_2 = \text{floor}(\text{rand}(0, 1) \cdot N)$; **while** (($r_2 = i$) or ($r_2 = r_2$));
$j$rand = floor(5 · rand(0, 1));
for ($j = 1; j < 5; j$++) // generate a trial vector
if (rand(0, 1) ≤ CR or $j = j$rand)
$t_i^j = s_i^j + F \cdot (s_{r_1}^j - s_{r_2}^j)$;
else
$t_i^j = s_i^j$;
end if
end for
end for

Procedure 1

for ($i = 1; i < N + 1; i$++)
if ($\mathbf{t}_i \prec \mathbf{s}_i$)
$\mathbf{s}_i = \mathbf{t}_i$
else
$\mathbf{s}_i = \mathbf{s}_i$
end if
end for

Procedure 2

Peak Signal to Noise Ratio (PSNR). Such indexes allow appropriately assessing the accuracy of the estimation.

The problem of homography estimation consists in finding a geometric transformation that maps points of a first view ($\mathbf{x}_i$) to a second view ($\mathbf{x}_i'$), taken from different point of view. This projective transformation $\mathbf{H}$ relates corresponding points of the plane projected into the first and second views by $\mathbf{x}_i' = \mathbf{H}\mathbf{x}_i$ or $\mathbf{x}_i = \mathbf{H}^{-1}\mathbf{x}_i'$. In order to calculate the MSE and the PSNR, two different images are defined: the estimated image (**EI**) and the actual image (**AI**). The **EI** is produced by mapping the pixels from the first view in terms of the estimated homography $\mathbf{H}$ ($\mathbf{EI} = \mathbf{H}\mathbf{x}_i$). On the other hand, the actual image (**AI**) corresponds to the second view image.

The mean squared error (MSE) evaluates the squared differences among the pixels of **EI** and **AI**. Considering that $D1 \times D2$ represents the image dimensions, the MSE can be computed as follows:

$$\text{MSE} = \frac{1}{D1 \cdot D2} \sum_{x=1}^{D1} \sum_{y=1}^{D2} \left| \mathbf{IA}(x, y) - \mathbf{IE}(x, y) \right|^2. \quad (14)$$

The Peak Signal to Noise Ratio (PSNR) is commonly used to measure the quality of reconstruction of an image that undergoes some process. The signal in this case is the original data (**AI**), and the noise is the error introduced by the transformation $\mathbf{H}$ (**EI**). When comparing images, PSNR is an approximation to human perception of approximation quality. A higher PSNR value generally indicates that the estimation is of higher quality. PSNR is mainly defined via the mean squared error (MSE). Given $D1 \times D2$ image, the PSNR is defined as

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right), \quad (15)$$

where MAX is the maximum possible pixel value contained in the image.

The images used in the experiments are collected from [40] which contains several two-view images of different objects, considering a dimension of 640 × 480 pixels. Likewise, the set of images used in the experimental test is presented in Figure 3.

For the test, both algorithms, NSGA-II and NSDE, have been configured considering 200 individuals under 200 iterations. In order to conduct a fair comparison between RANSAC and multiobjective approaches, RANSAC has been operated during 40,000 iterations. Such number of calculations (200 × 200) corresponds to the maximum number of evaluations invested by NSGA-II and NSDE during their execution.

Figure 4 shows the Pareto optimal set obtained by NSGA-II and NSDE during the estimation process, considering Figures 5(a) and 5(b) as the first and second views, respectively. In Figure 4, the best RANSAC estimations have been also included as a reference, only to validate the performance of the multiobjective approaches.

In order to illustrate the obtained results, Figure 5 shows the estimations produced by all methods in terms of their resulting estimated images. The single estimation generated by RANSAC is exhibited in Figures 5(c) and 5(d). In case of the multiobjective approaches, three solutions from the Pareto optimal set have been selected: the boundary solution for f_1, the boundary solution for f_2, and the median solution. Such solutions are presented in Figures 5(e)-5(f) and 5(g)-5(h), for NSGA-II and NSDE, respectively.

Table 1 presents the performance results for NSGA-II and NSDE in terms of the mean squared error (MSE) and the Peak Signal to Noise Ratio (PSNR) over three pairs of images. The results present the averaged outcomes obtained throughout 30 different executions. In order to appreciate the differences,

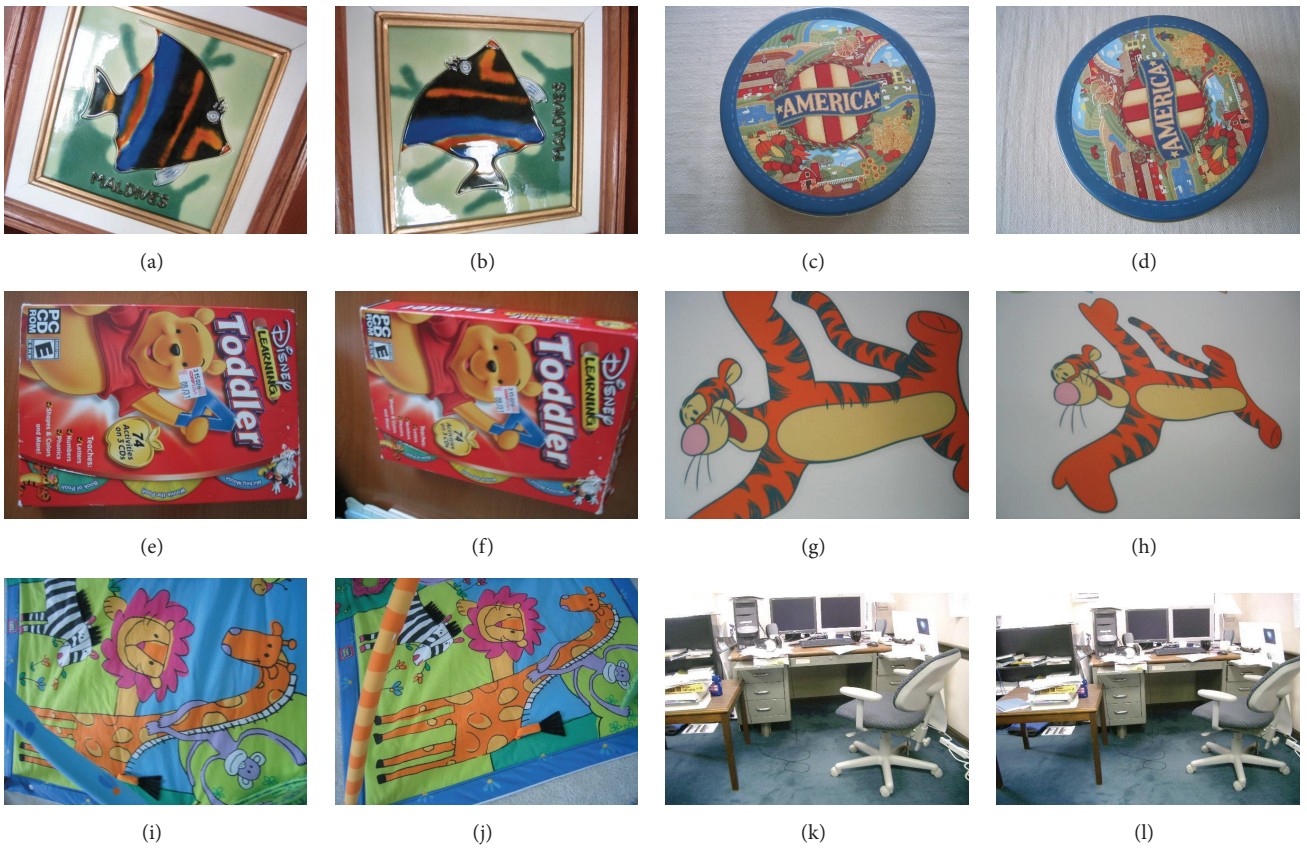


Figure 3: Set of images used in the experimental set.

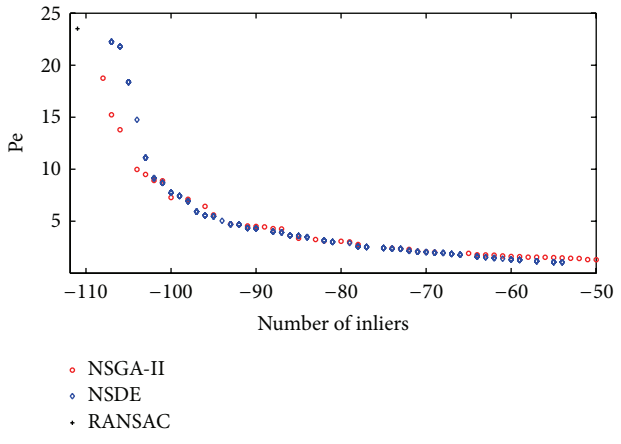


Figure 4: Pareto fronts found by NSGA-II and NSDE. The point obtained by RANSAC is placed only as a reference.

results only report the boundary solutions obtained for f_1 and f_2. The best results at each experiment are highlighted in Table 1.

In order to evaluate the robustness of both algorithms, a set of outliers was added by selecting correspondence random points within the space limits. In the test, the fraction of outliers varies from 85% to 95%. Figure 6 shows the estimations produced by all methods in terms of their resulting estimated images, considering image pairs in Figures 3(k) and 3(l). Table 2 presents the performance results for RANSAC, NSGA-II, and NSDE in terms of the mean squared error (MSE) over the three pairs of images. The results exhibit the averaged outcomes obtained throughout 30 different executions.

From Table 2, it can be easily seen that as the number of outliers increases, the performance of each algorithm also decreases. However, the NSDE algorithm obtains the best performance in almost every case despite outliers rising above 95%.

Figure 5: Estimation results: (a) and (b) original images, (c) and (d) RANSAC, (e) and (f) NSGA-II, and (g) and (h) NSDE.

The approach has been experimentally tested considering a set of benchmark experiments. The efficiency of the method has been evaluated in terms of the mean squared error (MSE) and the Peak Signal to Noise Ratio (PSNR) measurements. Experimental results that consider real images provide evidence on the remarkable performance of the proposed approach in comparison to the classical RANSAC.

The Wilcoxon signed rank test [41] is a nonparametric test used both to compare quantitatively some experimental data and also to determine whether there exists a meaningful difference among them. By applying the test to the data contained in Table 2, it was found that the algorithms NSGA-II and NSDE are substantially different with a 5% significance, so it can be considered that NSDE gives better results than NSGA-II when both algorithms are applied to the homography problem. In the same order of ideas, the same test was used to compare NSGA-II and RANSAC algorithms, causing the first algorithm to be better than the second.

7. Conclusions

In this work the use of two multiobjective evolutionary algorithms in conjunction with point correspondences is proposed to estimate homographies between image pairs. Under this approach, the estimation process is considered as a multiobjective problem with the number of matching points (f_1) and the permissible error (f_2) being simultaneously optimized. Under such circumstances, the approach has the capacity to find the best balance between both objectives.

A close inspection of the standard deviations from Table 1 reveals that NSGA-II maintains a big dispersion in its solutions. This aspect is mainly emphasized in the MSE index. Such an inconsistency is a consequence of the NSGA-II incapacity to produce similar solutions during its executions. On the contrary, NSDE produces better solutions than NSGA-II in terms of accuracy (MSE) and consistency. On the other hand, as a higher PSNR value indicates that the

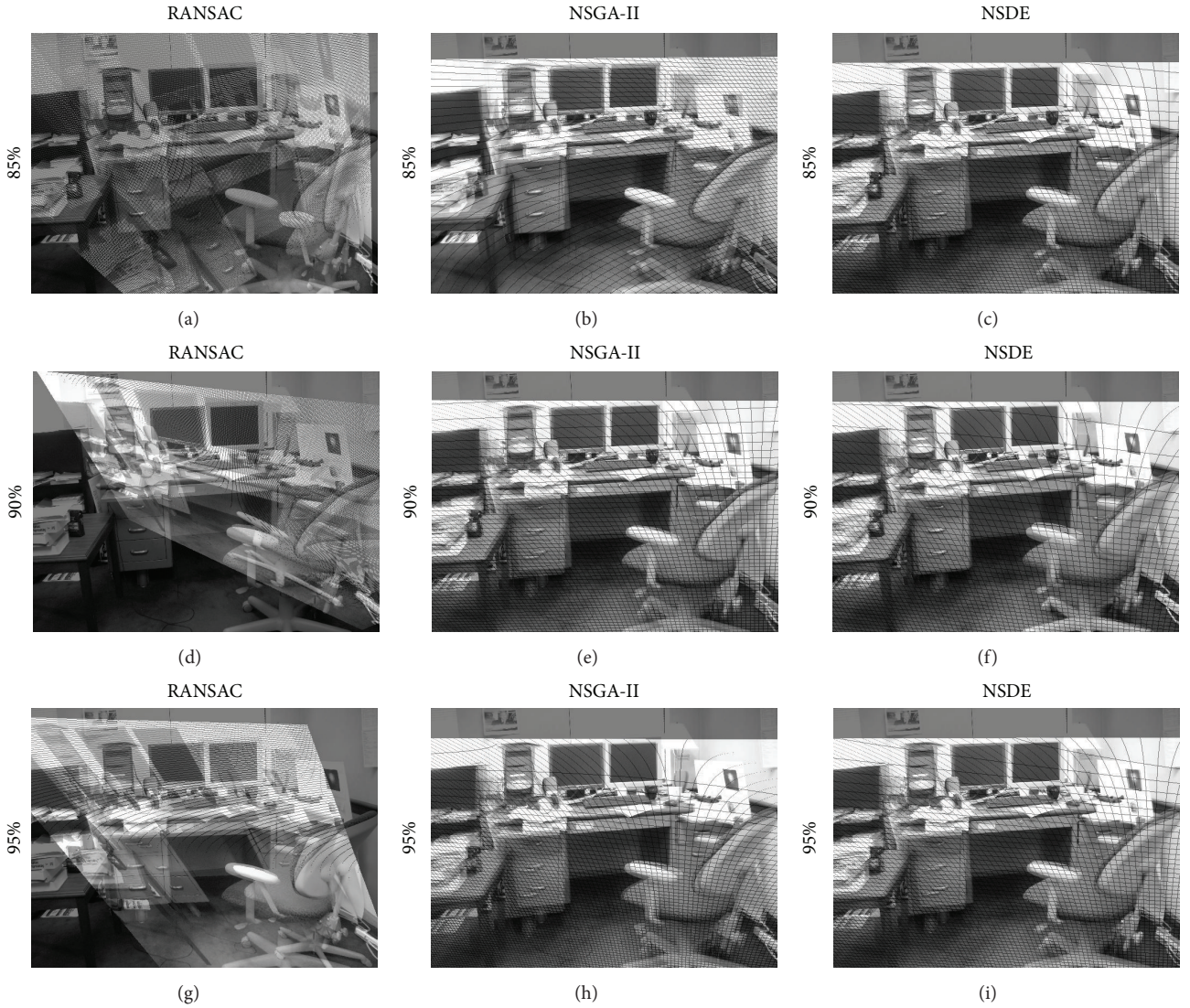


Figure 6: Results obtained by RANSAC, NSGA-II, and NSDE.

Table 2: Quality measures, calculated from single extreme values, found by RANSAC, NSDE, and NSGA-II, varying the outlier number.

Algorithm	Image pair	MSE $\mu(\sigma)$		
		95% of outliers	90% of outliers	85% of outliers
NSDE	Figures 3(g) and 3(h)	**117.9057 (17.2158)**	105.3281 (3.2716)	102.7769 (1.7186)
NSGA-II		146.7161 (3.3511)	140.1202 (13.2941)	122.5373 (22.6950)
RANSAC		135.0995 (17.8106)	**103.6220 (2.8878)**	**100.8954 (1.0529)**
NSDE	Figures 3(i) and 3(j)	**70.2051 (14.8488)**	**52.0970 (0.3064)**	**51.7556 (0.1016)**
NSGA-II		91.8883 (14.9284)	85.6721 (17.6376)	61.3195 (21.0951)
RANSAC		72.2009 (25.6447)	52.1122 (0.2773)	51.8429 (0.1234)
NSDE	Figures 3(k) and 3(l)	**109.3420 (15.7526)**	95.9762 (0.4746)	**95.7580 (0.0121)**
NSGA-II		135.5036 (5.6795)	110.0551 (21.9004)	96.9498 (2.5278)
RANSAC		120.9408 (21.7259)	**95.8901 (0.1421)**	95.8631 (0.1470)

estimation is of higher quality, results produced by the NSDE algorithm exhibit the best performance.

In order to solve the multiobjective formulation, two different evolutionary algorithms have been explored: the Nondominated Sorting Genetic Algorithm II (NSGA-II) and the Nondominated Sorting Differential Evolution (NSDE).

After several tests, it was found that NSDE gives better results in solving the image matching problem presented,

according to a known statistical test over a set of experimental results.

Conflict of Interests

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

Acknowledgment

This work was supported by the Consejo Nacional de Ciencia y Tecnología, CONACYT, under various grants.

References

[1] R. Szeliski and H.-Y. Y. Shum, "Creating full view panoramic image mosaics and environment maps," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*, pp. 251–258, ACM, Los Angeles, Calif, USA, August 1997.

[2] Y. He and R. Chung, "Image mosaicking for polyhedral scene and in particular singly visible surfaces," *Pattern Recognition*, vol. 41, no. 3, pp. 1200–1213, 2008.

[3] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International Journal of Computer Vision*, vol. 74, no. 1, pp. 59–73, 2007.

[4] A. Akyol and M. Gökmen, "Super-resolution reconstruction of faces by enhanced global models of shape and texture," *Pattern Recognition*, vol. 45, no. 12, pp. 4103–4116, 2012.

[5] H. Huang, H. T. He, X. Fan, and J. P. Zhang, "Super-resolution of human face image using canonical correlation analysis," *Pattern Recognition*, vol. 43, no. 7, pp. 2532–2543, 2010.

[6] K. Jia and S. G. Gong, "Hallucinating multiple occluded face images of different resolutions," *Pattern Recognition Letters*, vol. 27, no. 15, pp. 1768–1775, 2006.

[7] O. Deniz, G. Bueno, E. Bermejo, and R. Sukthankar, "Fast and accurate global motion compensation," *Pattern Recognition*, vol. 44, no. 12, pp. 2887–2901, 2011.

[8] H. Zhang and J. P. Ostrowski, "Visual motion planning for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 199–208, 2002.

[9] C. Sagüés and J. J. Guerrero, "Visual correction for mobile robot homing," *Robotics and Autonomous Systems*, vol. 50, no. 1, pp. 41–49, 2005.

[10] G. López-Nicolás, J. J. Guerrero, and C. Sagüés, "Visual control of vehicles using two-view geometry," *Mechatronics*, vol. 20, no. 2, pp. 315–325, 2010.

[11] E. Montijano and C. Sagues, "Distributed multi-camera visual mapping using topological maps of planar regions," *Pattern Recognition*, vol. 44, no. 7, pp. 1528–1539, 2011.

[12] J. Su, R. Chung, and L. Jin, "Homography-based partitioning of curved surface for stereo correspondence establishment," *Pattern Recognition Letters*, vol. 28, no. 12, pp. 1459–1471, 2007.

[13] T. T. Santos and C. H. Morimoto, "Multiple camera people detection and tracking using support integration," *Pattern Recognition Letters*, vol. 32, no. 1, pp. 47–55, 2011.

[14] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[15] X. Qian and C. Ye, "NCC-RANSAC: a fast plane extraction method for 3-D range data segmentation," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2771–2783, 2014.

[16] J. Matas and O. Chum, "Randomized RANSAC with $T_{d,d}$ test," *Image and Vision Computing*, vol. 22, no. 10, pp. 837–842, 2004.

[17] C.-M. Cheng and S.-H. Lai, "A consensus sampling technique for fast and robust model fitting," *Pattern Recognition*, vol. 42, no. 7, pp. 1318–1329, 2009.

[18] E. Cuevas and M. Díaz, "A method for estimating view transformations from image correspondences based on the harmony search algorithm," *Computational Intelligence and Neuroscience*, vol. 2015, Article ID 434263, 15 pages, 2015.

[19] P. H. S. Torr and A. Zisserman, "MLESAC: a new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, 2000.

[20] S. B. Heinrich, "Efficient and robust model fitting with unknown noise scale," *Image and Vision Computing*, vol. 31, no. 10, pp. 735–747, 2013.

[21] R. Subbarao and P. Meer, "Beyond RANSAC: user independent robust regression," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition Workshop (CVPRW '06)*, p. 101, IEEE, June 2006.

[22] H. Wang and D. Suter, "Robust adaptive-scale parametric model estimation for computer vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1459–1474, 2004.

[23] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, New York, NY, USA, 2001.

[24] A. Rosales-Pérez, J. A. Gonzalez, C. A. Coello Coello, H. J. Escalante, and C. A. Reyes-Garcia, "Multi-objective model type selection," *Neurocomputing*, vol. 146, pp. 83–94, 2014.

[25] I. Giagkiozis and P. J. Fleming, "Methods for multi-objective optimization: an analysis," *Information Sciences*, vol. 293, pp. 338–350, 2014.

[26] M. Li, S. Yang, and X. Liu, "Diversity comparison of Pareto front approximations in many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2568–2584, 2014.

[27] Z.-H. Zhan, J. Li, J. Cao, J. Zhang, H. S.-H. Chung, and Y.-H. Shi, "Multiple populations for multiple objectives: a coevolutionary technique for solving multiobjective optimization problems," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 445–463, 2013.

[28] X.-B. Hu, M. Wang, and E. Di Paolo, "Calculating complete and exact pareto front for multiobjective optimization: a new deterministic approach for discrete problems," *IEEE Transactions on Cybernetics*, vol. 43, no. 3, pp. 1088–1101, 2013.

[29] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimization for feature selection in classification: a multi-objective approach," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1656–1671, 2013.

[30] C. Liu, J. Liu, and Z. Jiang, "A multiobjective evolutionary algorithm based on similarity for community detection from signed social networks," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2274–2287, 2014.

[31] H. J. Sun, C. H. Peng, J. F. Guo, and H. S. Li, "Non-dominated sorting differential evolution algorithm for multi-objective optimal integrated generation bidding and scheduling," in *Proceedings of the IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS '09)*, vol. 1, pp. 372–376, Shanghai, China, November 2009.

[32] K. Deb, A. Pratap, S. Agarwal, 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.

[33] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Surf: speeded up robust features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[34] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, pp. 147–151, 1988.

[35] R. I. Hartley and A. Zisserman, *Multiple View Geometry Try in Computer Vision*, Cambridge University Press, 2nd edition, 2004.

[36] P. Meer, "Robust techniques in computer vision," in *Emerging Topics in Computer Vision*, G. Medioni and S. B. Kang, Eds., pp. 107–190, Prentice-Hall, Boston, Mass, USA, 2004.

[37] M. Ray and D. P. Mohapatra, "Multi-objective test prioritization via a genetic algorithm," *Innovations in Systems and Software Engineering*, vol. 10, no. 4, pp. 261–270, 2014.

[38] M. Ehrgott, J. Ide, and A. Schöbel, "Minmax robustness for multi-objective optimization problems," *European Journal of Operational Research*, vol. 239, no. 1, pp. 17–31, 2014.

[39] R. Storn and K. Price, "Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces," Tech. Rep. TR-95-012, 012, International Computer Science Institute, Berkeley, Calif, USA, 1995.

[40] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, vol. 2, pp. 2161–2168, June 2006.

[41] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2015, Article ID 875735, 8 pages
http://dx.doi.org/10.1155/2015/875735

Research Article

Saliency Mapping Enhanced by Structure Tensor

Zhiyong He,[1] Xin Chen,[2] and Lining Sun[1]

[1]*School of Mechanical and Electric Engineering, Soochow University, Suzhou 215021, China*
[2]*NovuMind Inc., Santa Clara, CA 95131, USA*

Correspondence should be addressed to Zhiyong He; he-zhiyong@139.com

Received 4 June 2015; Revised 11 September 2015; Accepted 27 September 2015

Academic Editor: Paolo Del Giudice

We propose a novel efficient algorithm for computing visual saliency, which is based on the computation architecture of Itti model. As one of well-known bottom-up visual saliency models, Itti method evaluates three low-level features, color, intensity, and orientation, and then generates multiscale activation maps. Finally, a saliency map is aggregated with multiscale fusion. In our method, the orientation feature is replaced by edge and corner features extracted by a linear structure tensor. Following it, these features are used to generate contour activation map, and then all activation maps are directly combined into a saliency map. Compared to Itti method, our method is more computationally efficient because structure tensor is more computationally efficient than Gabor filter that is used to compute the orientation feature and our aggregation is a direct method instead of the multiscale operator. Experiments on Bruce's dataset show that our method is a strong contender for the state of the art.

1. Introduction

Visual saliency (also called *visual salience*) refers to the quality or state by which information stands out relative to its neighbors and often attracts human attention [1]. In the subsequent stages, the salient images are preferentially taken as inputs instead of the whole image. As a consequence, visual saliency has been widely applied to various computer vision tasks such as segmentation [2, 3], image retargeting [4–6], object detection [7, 8], image collection [9], and object recognition [10].

Koch and Ullman introduced a basic biologically inspired architecture of visual saliency, referred to as Koch and Ullman model [11]. Then Itti et al. presented a computational architecture to implement and verify the Koch and Ullman model [12]. As summarized in [13], most of the implementation techniques of the visual saliency models generally have three stages: (1) *extraction:* extracting low-level features at locations over the image plane, (2) *activation:* forming activation maps from the features, and (3) *normalization/combination:* normalizing the activation maps and combining them into a single saliency map.

For Itti method, the objective of the first stage is to extract three low-level features, intensity, color, and orientation, and followed by using Difference of Gaussian (DOG) to form total of forty activation maps. Finally, a linear operator is employed to normalize these maps, in which the most salient location is selected by the winner-take-all neural network to generate a saliency map. However, results of Itti method are sometimes blurry and prefer small and local features, which are less useful for some further computer vision applications such as object segmentation and detection.

Despite many advances of the visual saliency made in recent years, the various evaluation results in [14] indicate that there are still some questions about the mechanism of visual saliency. In addition to a motivation of investigation on some questions such as low-level features and the combination of activation maps, in this paper, we also focus on the performance of algorithm and whether the saliency results can greatly benefit computer vision applications.

Results from some recent research works have shown that features of edge and corner also play important roles in visual saliency [13, 15, 16]. In our study, we also note that orientation features are less likely to win in the combination of activation maps. Moreover, Gabor filters used for orientation extraction are computationally expensive.

We therefore propose an efficient method to compute saliency map, referred to as *structure tensor* (ST) saliency map. The computational architecture of our method is shown

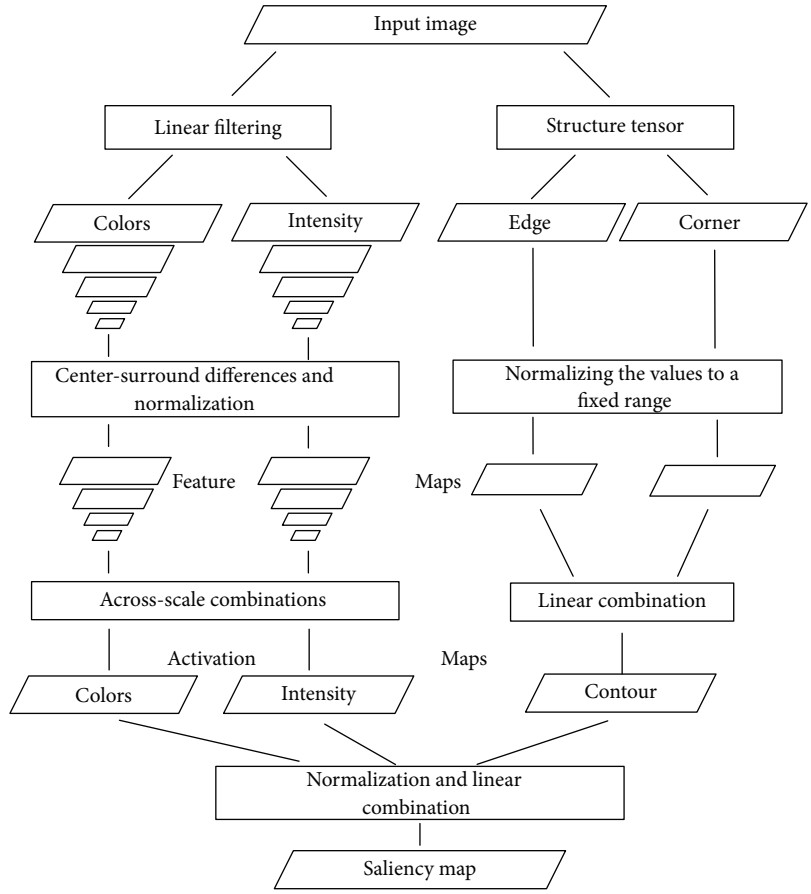


Figure 1: General architecture of our method. In our method, we call the activation map generated by edge and corner features contour activation map. The final saliency map combined these activation maps into ST saliency map.

in Figure 1 and is the same as Itti method in feature extraction and activation maps generation for intensity and color features. The features of edge and corner are extracted by structure tensor and directly combined into an activation map, called contour map. After obtaining three activation maps, we use linear combination to aggregate activation maps to a saliency map instead of multiscale combination and winner-take-all rule.

This paper makes two major contributions as follows:

(1) We propose a novel efficient algorithm to calculate the saliency map. Compared to other methods performed on a challenging dataset, besides the best performance achieved, the results of our method obtain sharper boundaries which are useful in some further applications such as object segmentation and detection.

(2) Our work has shown that edge and corner are two important low-level features in saliency generation.

The paper is organized as follows. Section 2 briefly reviews the state-of-the-art methods with particular emphasis on saliency algorithms related to Itti method, and Section 3 introduces some backgrounds of structure tensor and formally describes our algorithm of saliency map computation. In Section 4, we present our experimental results and quantitative evaluations on a challenging dataset and discuss them. This paper closes with a conclusion of our work in Section 5.

2. Related Work

Visual saliency methods are generally categorized into biologically inspired methods and computationally oriented methods. There is an extensive literature on the areas, but here we mention just a few relevant papers. Some surveys are found in [17–19], and some recent progress is reported in [20].

Koch and Ullman [11] proposed a basic architecture of biologically inspired methods and defined a saliency map as a topographic map that represents conspicuousness of scene locations. Their work also introduced a winner-take-all neural network that selects the most salient location and employs an inhibition of return mechanism to allow the focus of attention to shift to the next most salient location. Then Itti et al. presented a computational model to implement and verify Koch and Ullman model. Since then, the works related to the saliency map have quickly become one of the hot research fields.

Itti method employs a Difference of Gaussian (DOG) operator to evaluate color, intensity, and orientation features

to generate total of forty activation maps and across-scale-combines these maps into a saliency map. Besides the expensive computation, one big problem of Itti method is that the results are sometimes blurry and prefers small purely local features. On the other hand, many algorithms of computer vision need input features related to contours because they require the distinct boundary information. Recently, some methods have been proposed to obtain sharp edges, for example, local dissimilarities at the pixel level [21], multiscale DOG [22], and histogram analysis [15]. However, the common problem of these methods is that they are more sensitive to the noises.

As mentioned in the previous section, improving on Itti method, we propose an efficient algorithm for calculating the saliency map, and the computational architecture of our method is shown in Figure 1. The computational architecture of our method is similar to Itti method, and our method evaluates intensity, color, edge, and corner features instead of intensity, color, and orientation features. The structure tensor is used to extract the features of the edge and corner. In the final step, we use the linear combination to generate a saliency map instead of the winner-take-all rule of Itti method.

3. The Proposed Saliency Model

In this section, we briefly introduce the background of structure tensor and formally describe our algorithm.

3.1. Introduction to Structure Tensor. In mathematics, structure tensor is a matrix representation of partial derivative information. In the field of image processing and computer vision, it typically represents the gradient or edge information and has a more powerful description of local patterns as opposed to the directional derivative through its coherence measure [23, 24].

There are two categories of structure tensor: linear structure tensor and nonlinear structure tensor. Compared to the nonlinear structure tensor, the linear structure tensor is fast and easy to implement with Fast Fourier Transform (FFT). We therefore select the linear structure tensor to extract the features of edges and corners.

Given an image $I(x, y)$, if pixel (x, y) translates to $(x + \Delta x, y + \Delta y)$, the energy E is defined as

$$E = \sum_{(u,v)\in W(x,y)} w(u,v)\left(I(u,v) - I(u+\Delta x, v+\Delta y)\right)^2, \quad (1)$$

where $W(x, y)$ is a window at center point (x, y) and $w(u, v)$ is a weight function at pixel (x, y). In the rest of this paper, $\sum_{(u,v)\in W(x,y)} w(u, v)$ is simply written as $\sum_w$.

It is approximated by a first-order Taylor series:

$$I(u+\Delta x, v+\Delta y) \approx I(u,v) + \frac{\partial I}{\partial x}(u,v)\Delta x + \frac{\partial I}{\partial y}(u,v)\Delta y = I(u,v) + \left[\frac{\partial I}{\partial x}(u,v), \frac{\partial I}{\partial y}(u,v)\right]\begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}. \quad (2)$$

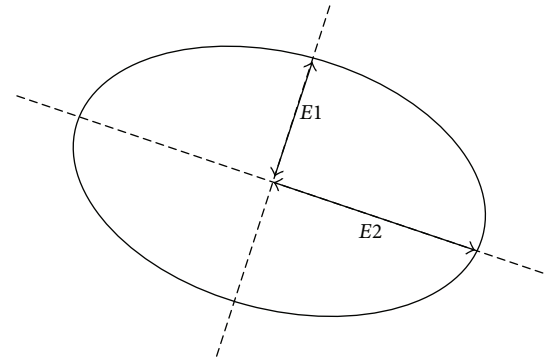


Figure 2: The relation between ellipse and structure tensor.

Hence, (1) can be rewritten as

$$E = \sum_w \left(I(u,v) - I(u+\Delta x, v+\Delta y)\right)^2 \approx [\Delta x, \Delta y]\,\mathbf{T}\begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}, \quad (3)$$

where matrix $\mathbf{T}$ is

$$\mathbf{T} = \sum_w \begin{bmatrix}\left(\frac{\partial I}{\partial x}\right)^2 & \frac{\partial I}{\partial x}\cdot\frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial x}\cdot\frac{\partial I}{\partial y} & \left(\frac{\partial I}{\partial y}\right)^2\end{bmatrix} = \begin{bmatrix}\sum_w\left(\frac{\partial I}{\partial x}\right)^2 & \sum_w\frac{\partial I}{\partial x}\cdot\frac{\partial I}{\partial y} \\ \sum_w\frac{\partial I}{\partial x}\cdot\frac{\partial I}{\partial y} & \sum_w\left(\frac{\partial I}{\partial y}\right)^2\end{bmatrix}. \quad (4)$$

Matrix $\mathbf{T}$ is a structure tensor, which is also considered as a covariance matrix.

We also consider (3) as an approximation of a binomial function, and from a view of geometry, a binomial function is an ellipse where short axis and long axis are represented as eigenvalues $E1$ and $E2$, respectively. The direction of ellipse is determined by the eigenvectors. As shown in Figure 2, the equation of ellipse is written as

$$[\Delta x, \Delta y]\,\mathbf{T}\begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix} = 1. \quad (5)$$

(1) Input:
(2) Input image **I**: three-channel and size (m, n)
(3) Output:
(4) Edge feature map **A**: one channel and size $(m1, n1)$
(5) Corner feature map **B**: one channel and size $(m1, n1)$
(6) Contour activation map **C**: one channel and size $(m1, n1)$
(7) Begin
(8) Resize the input image **I** to $(m1, n1)$, called Im-Re
(9) **for** $j \leftarrow 1, n1$ **do**
(10) **for** $i \leftarrow 1, m1$ **do**
(11) For Im-Re, calculate structure tensor J_σ using (6)
(12) Calculate eigenvalues λ_1 and λ_2 using (7) and (8), respectively
(13) $A(i, j) = \lambda_1 - \lambda_2$
(14) $B(i, j) = \lambda_1 + \lambda_2$
(15) **end for**
(16) **end for**
(17) Normalize **A** and **B** into a fixed range $[0 \cdots 1]$
(18) Combine normalized **A** and normalized **B** into **CT**
(19) End

Algorithm 1: Algorithm of contour activation map.

Based on (4), some types of structure tensor have been constructed. In our work, we use a linear structure tensor to analyze the input image, and it is defined as

$$J_\sigma = \sum_{i=1}^{3} \begin{bmatrix} K_\sigma * \left(\frac{\partial I_i}{\partial x}\right)^2 & K_\sigma * \left(\frac{\partial I_i}{\partial x} \cdot \frac{\partial I_i}{\partial y}\right) \\ K_\sigma * \left(\frac{\partial I_i}{\partial x} \cdot \frac{\partial I_i}{\partial y}\right) & K_\sigma * \left(\frac{\partial I_i}{\partial y}\right)^2 \end{bmatrix}, \quad (6)$$

where K_σ is a Gaussian kernel with variance σ and $*$ is a convolution operator. The parameter i is the image channel number.

For any kind of structure tensor, we use $\left[\begin{smallmatrix} G & F \\ F & H \end{smallmatrix}\right]$ to simply represent matrix **T** of (4). Then the two eigenvalues are calculated as

$$\lambda_1 = \frac{G + H + \sqrt{(G-H)^2 + 4F^2}}{2}, \quad (7)$$

$$\lambda_2 = \frac{G + H - \sqrt{(G-H)^2 + 4F^2}}{2}. \quad (8)$$

3.2. Contour Activation Map. As shown in Figure 1, we calculate the activation maps of color and intensity with Itti method, and the contour activation map is detailed in Algorithm 1.

For computation of **A** and **B**, we do not need to compute λ_1 and λ_2 with (7) and (8) and add and subtract these values to calculate **A** and **B**. We directly compute them as

$$\begin{aligned} \lambda_1 - \lambda_2 &= \sqrt{(G-H)^2 + 4F^2}, \\ \lambda_1 + \lambda_2 &= G + H. \end{aligned} \quad (9)$$

In the final step, we combine feature maps into a contour activation map **CT** as follows:

$$\mathbf{CT} = \frac{1}{2}\left(N(A) + N(B)\right), \quad (10)$$

where $N(A)$ is the normalized edge feature map and $N(B)$ is the normalized corner feature map.

3.3. ST Saliency Map Generation. We assume that all features equally contribute to the ST saliency map generation. After obtaining the contour activation map, the intensity activation map, and the color activation map, we combine them into a saliency map as follows:

$$S = \frac{1}{3}\left(\bigoplus(\mathbf{CL}) + \bigoplus(I) + \bigoplus(\mathbf{CT})\right), \quad (11)$$

where $\bigoplus$ is a normalization operation which is defined in [12], **CL** is the color activation map, **I** is the intensity activation map, and **CT** is the contour activation map.

Some saliency maps of our method are shown in Figure 3 and these maps have distinct boundaries.

4. Experimental Results

In this section, we present subjective evaluation and quantitative analysis of our method and some state-of-the-art methods and analysis of performance of these methods.

4.1. Saliency Maps. We compared saliency maps of our method with saliency maps of some state-of-the-art methods including Itti method [12], Attention Based on Information Maximization (AIM) method [25, 26], Dynamic Visual Attention (DVA) method [27], Graphic-Based Visual Saliency (GBVS) method [13], and Image Signature (IS) method [28]. The MATLAB implementation of these methods is based on

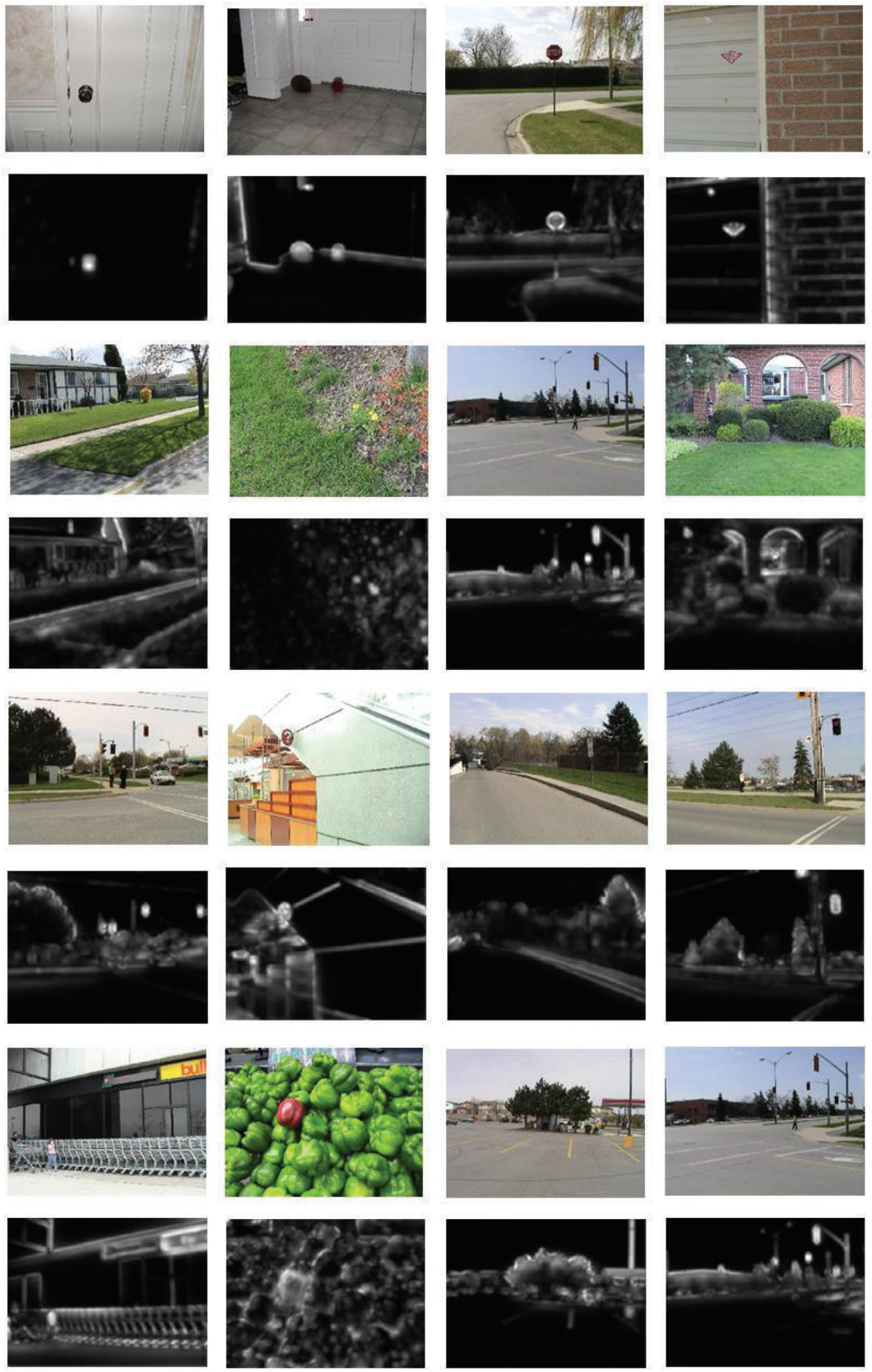


Figure 3: Saliency maps of our method. The odd row is the input images from Bruce dataset, and the even row is the saliency generated by our method. Obviously, the saliency maps of our method have sharp edges.

Figure 4: Saliency maps on the Bruce dataset. (a) Input image, (b) our method, (c) Itti method, (d) AIM method, (e) DVA method, (f) GBVS method, and (g) IS method using LAB color space. Since our method includes the edge and corner information, saliency maps of our method have sharp edges that are useful for the further steps in some computer vision tasks.

the codes on the authors' websites. Saliency maps are shown in Figure 4.

4.2. Analysis of Performance. We evaluated our method on Bruce dataset containing 120 natural images with eye fixation ground truth data. In Bruce dataset, the size of all images is 681×511. Some of methods are sensitive to different sizes of the input image. As a consequence, in order to fairly evaluate results of different methods, we resize the input images to the same size (170×128) for each method.

Results from perceptual research works [29, 30] have found that human fixations have strong center bias which may affect the performance of a saliency algorithm. To remove this center bias, following the procedure of Tatler et al.'s work [29], Hou et al. [28] introduced ROC Area Under the Curve (AUC) score to quantitatively evaluate the performance of different algorithms. Good results should maximize the ROC AUC score. To compare the ROC AUC scores, we follow the computation method provided by [28], but the size (170×128) is different with two input image sizes used in [28]. Comparison of the ROC AUC scores is shown in Figure 5.

We conducted our tests on a laptop with Intel Dual-Core i5-4210U 1.7 GHz CPU and 4 G RAM memory. All codes were written in MATLAB.

The execution times of the methods are summarized in Figure 6, in which the time is an average time of 120 images.

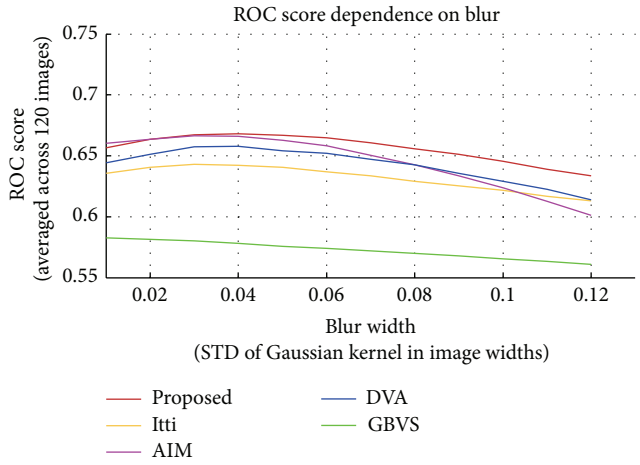


Figure 5: Comparison of the ROC AUC scores of all methods. Our method achieves the best ROC AUC score.

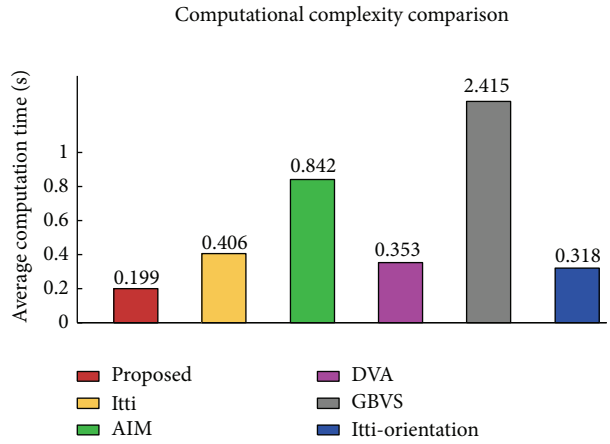


Figure 6: Results of the performance of these different methods. Time measurements are given in seconds. The results are the average times of 120 images of Bruce dataset.

The figure shows that our method is about twice as fast as Itti method and outperforms other state-of-the-art methods. The reason lies in two parts. First, structure tensor is an efficient algorithm of feature extraction. Second, we directly aggregate three activation maps into a saliency map. It is obvious that the performance will increase greatly if our method is implemented by C/C++, and it should satisfy most of the real time applications.

5. Conclusion

In this paper we have proposed an efficient algorithm for computing the saliency map, which has a distinct boundary that contributes to further computer vision applications such as segmentation and detection. The computational architecture of our method is close to Itti method, but we have made two improvements in low-level features extraction and combination of activation maps. Since features of edge and corner are important cues in visual saliency, we use a linear structure tensor to extract these features. The reason that our algorithm is efficient lies in the following: (1) linear structure tensor is an efficient feature extraction algorithm and (2) our linear combination method is fast. On the basis of experimental results on Bruce dataset, our method has shown that some computer vision tasks, in particular real time applications, can benefit from our method as a preprocessing step.

Conflict of Interests

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

Acknowledgments

This work was supported in part by the Chinese National Natural Science Foundation (NSFC 61473201, 51405320), the Natural Science Foundation of Jiangsu Province (BK20150339), and the Science and Technology Program of Suzhou (SYG201424).

References

[1] M. Carrasco, "Visual attention: the past 25 years," *Vision Research*, vol. 51, no. 13, pp. 1484–1525, 2011.

[2] J. Han, K. N. Ngan, M. Li, and H.-J. Zhang, "Unsupervised extraction of visual attention objects in color images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 1, pp. 141–145, 2006.

[3] E. Rahtu, J. Kannala, M. Salo, and J. Heikkilä, "Segmenting salient objects from images and videos," in *Computer Vision—ECCV 2010*, vol. 6315 of *Lecture Notes in Computer Science*, pp. 366–379, Springer, Berlin, Germany, 2010.

[4] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," *ACM Transactions on Graphics*, vol. 26, no. 3, article 10, 2007.

[5] S. Goferman, L. Zelnik-Manor, and A. Tal, "Context-aware saliency detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 10, pp. 1915–1926, 2012.

[6] D. Vaquero, M. Turk, K. Pulli, M. Tico, and N. Gelfand, "A survey of image retargeting techniques," in *Applications of Digital Image Processing XXXIII*, vol. 7798 of *Proceedings of SPIE*, pp. 779–814, SPIE Optical Engineering + Applications, San Diego, Calif, USA, August 2010.

[7] A. Oliva, A. Torralba, M. S. Castelhano, and J. M. Henderson, "Top-down control of visual attention in object detection," in *Proceedings of the International Conference on Image Processing (ICIP '03)*, pp. I-253–I-256, September 2003.

[8] X. Shen and Y. Wu, "A unified approach to salient object detection via low rank matrix recovery," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 853–860, Providence, RI, USA, June 2012.

[9] M.-M. Cheng, N. J. Mitra, X. Huang, and S.-M. Hu, "SalientShape: group saliency in image collections," *The Visual Computer*, vol. 30, no. 4, pp. 443–453, 2014.

[10] U. Rutishauser, D. Walther, C. Koch, and P. Perona, "Is bottom-up attention useful for object recognition?" in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '04)*, vol. 2, pp. II-37–II-44, IEEE, July 2004.

[11] C. Koch and S. Ullman, "Shifts in selective visual attention: towards the underlying neural circuitry," *Human Neurobiology*, vol. 4, no. 4, pp. 219–227, 1985.

[12] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998.

[13] J. Harel, C. Koch, and P. Perona, "Graph-based visual saliency," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS '06)*, pp. 545–552, Vancouver, Canada, December 2006.

[14] M.-M. Cheng, G.-X. Zhang, N. J. Mitra, X. Huang, and S.-M. Hu, "Global contrast based salient region detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '11)*, pp. 409–416, Providence, RI, USA, June 2011.

[15] T. Liu, Z. Yuan, J. Sun et al., "Learning to detect a salient object," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 2, pp. 353–367, 2011.

[16] R. Valenti, N. Sebe, and T. Gevers, "Image saliency by isocentric curvedness and color," in *Proceedings of the 12th IEEE International Conference on Computer Vision*, pp. 2185–2192, IEEE, Kyoto, Japan, September-October 2009.

[17] A. Borji and L. Itti, "State-of-the-art in visual attention modeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 185–207, 2013.

[18] A. Borji, H. R. Tavakoli, D. N. Sihite, and L. Itti, "Analysis of scores, datasets, and models in visual saliency prediction," in *Proceedings of the 14th IEEE International Conference on Computer Vision (ICCV '13)*, pp. 921–928, Sydney, Australia, December 2013.

[19] S. Frintrop, E. Rome, and H. I. Christensen, "Computational visual attention systems and their cognitive foundations: a survey," *ACM Transactions on Applied Perception*, vol. 7, no. 1, article 6, 2010.

[20] Z. Bylinskii, T. Judd, A. Borji et al., "MIT Saliency Benchmark," 2015, http://saliency.mit.edu/index.html.

[21] Y.-F. Ma and H.-J. Zhang, "Contrast-based image attention analysis by using fuzzy growing," in *Proceedings of the 11th ACM International Conference on Multimedia (MM '03)*, pp. 374–381, ACM, November 2003.

[22] L. Itti and P. F. Baldi, "Bayesian surprise attracts human attention," in *Advances in Neural Information Processing Systems*, pp. 547–554, MIT Press, 2005.

[23] T. Brox, J. Weickert, B. Burgeth, and P. Mrázek, "Nonlinear structure tensors," *Image and Vision Computing*, vol. 24, no. 1, pp. 41–55, 2006.

[24] U. Köthe, "Edge and junction detection with an improved structure tensor," in *Pattern Recognition*, pp. 25–32, Springer, Berlin, Germany, 2003.

[25] N. Bruce and J. Tsotsos, "Saliency based on information maximization," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS '05)*, pp. 155–162, Vancouver, Canada, December 2005.

[26] N. D. B. Bruce and J. K. Tsotsos, "Saliency, attention and visual search: an information theoretic approach," *Journal of Vision*, vol. 9, no. 3, article 5, 2009.

[27] X. Hou and L. Zhang, "Dynamic visual attention: searching for coding length increments," in *Advances in Neural Information Processing Systems*, pp. 681–688, MIT Press, 2009.

[28] X. Hou, J. Harel, and C. Koch, "Image signature: highlighting sparse salient regions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 194–201, 2012.

[29] B. W. Tatler, R. J. Baddeley, and I. D. Gilchrist, "Visual correlates of fixation selection: effects of scale and time," *Vision Research*, vol. 45, no. 5, pp. 643–659, 2005.

[30] L. Zhang, M. H. Tong, T. K. Marks, H. Shan, and G. W. Cottrell, "SUN: a bayesian framework for saliency using natural statistics," *Journal of Vision*, vol. 8, no. 7, article 32, 2008.

Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2015, Article ID 285730, 15 pages
http://dx.doi.org/10.1155/2015/285730

Research Article

An Enhanced Differential Evolution Algorithm Based on Multiple Mutation Strategies

Wan-li Xiang, Xue-lei Meng, Mei-qing An, Yin-zhen Li, and Ming-xia Gao

School of Traffic & Transportation, Lanzhou Jiaotong University, Lanzhou, Gansu 730070, China

Correspondence should be addressed to Wan-li Xiang; xiangwl@tju.edu.cn

Received 12 May 2015; Accepted 5 July 2015

Academic Editor: Yufeng Zheng

Differential evolution algorithm is a simple yet efficient metaheuristic for global optimization over continuous spaces. However, there is a shortcoming of premature convergence in standard DE, especially in DE/best/1/bin. In order to take advantage of direction guidance information of the best individual of DE/best/1/bin and avoid getting into local trap, based on multiple mutation strategies, an enhanced differential evolution algorithm, named EDE, is proposed in this paper. In the EDE algorithm, an initialization technique, opposition-based learning initialization for improving the initial solution quality, and a new combined mutation strategy composed of DE/current/1/bin together with DE/*p*best/bin/1 for the sake of accelerating standard DE and preventing DE from clustering around the global best individual, as well as a perturbation scheme for further avoiding premature convergence, are integrated. In addition, we also introduce two linear time-varying functions, which are used to decide which solution search equation is chosen at the phases of mutation and perturbation, respectively. Experimental results tested on twenty-five benchmark functions show that EDE is far better than the standard DE. In further comparisons, EDE is compared with other five state-of-the-art approaches and related results show that EDE is still superior to or at least equal to these methods on most of benchmark functions.

1. Introduction

Optimization problems are ubiquitous in the various areas including production, life, and scientific community. These optimization problems are usually nonlinear and nondifferentiable. Particularly, the number of their local optima may increase exponentially with the problem size. Thus, evolutionary algorithms (EAs) only needing the value information of objective functions have many more advantages and have drawn more and more attention of many researchers all over the world. In this way, a lot of researchers have developed a great number of evolutionary algorithms, such as genetic algorithms (GAs), particle swarm optimization (PSO), ant colony optimization (ACO), and differential evolution (DE) algorithm. Among them, differential evolution is one of the most powerful stochastic real-parameter optimization algorithms [1]. It was originally developed by Storn and Price [2, 3] in 1995.

Due to its simple implementation, few control parameters, and fast convergence, DE has been widely and successfully applied in function optimization problems [2–26], constrained optimization problems [27–29], multiobjective optimization problems [30], scheduling [31–33], and others [34–39].

According to the aforementioned statements, it can be seen that DE has been very successful in solving various optimization problems. As far as the type of optimization problems is concerned, more researches mainly focus on continuous function optimization. However, the convergence precision and convergence speed over function optimization are still to be improved. That is, the exploration ability and exploitation ability of DE cannot be well balanced. To overcome the shortage of imbalance of the two abilities, more and more researchers have developed a large number of DE variants. For example, Noman and Iba [11] proposed a kind of accelerated differential evolution by incorporating an adaptive local search technique. Rahnamayan et al. [13] proposed an opposition-based differential evolution (ODE for short), in which a novel opposition-based learning (OBL) technique and a generation-jumping scheme are employed.

Qin et al. [14] proposed a self-adaptive differential evolution algorithm, called SaDE, in which both trial vector generation strategies and their associated parameter values are dynamically self-adapted during the process of producing promising solutions. Zhang and Sanderson [15] proposed a novel differential evolution referred to as JADE, in which a novel self-adaptive parameters scheme and a new mutation strategy with optional archive are proposed. And these improvements made JADE achieve a very fast convergence speed and high-quality solutions. Subsequently, Gong et al. [22, 23] proposed a few enhanced DE versions based on JADE by introducing adaptive strategy selection schemes or control parameters adaption mechanisms. In summary, all these state-of-the-art DE variants have achieved better convergence performance than the traditional DE.

Unfortunately, up to now, there exists no specific DE version to substantially achieve the best solution for all optimization problems because the exploration and the exploitation often mutually contradict in reality. Hence, searching for better approaches is very necessary. In order to solve continuous optimization problems more efficiently, an enhanced differential evolution algorithm based on multiple mutation strategies, called EDE for short, is presented in this paper.

The structure of the paper is organized as follows. The standard differential evolution algorithm is described briefly in Section 2. In Section 3, an enhanced differential evolution algorithm is presented and described in detail. Subsequently, Section 4 employs a set of benchmark functions to comprehensively investigate the performance of the proposed algorithm through experimental results of these functions and comparisons with other well-known evolutionary algorithms. Finally, conclusions and further study directions are given in Section 5.

2. Differential Evolution Algorithm

Differential evolution algorithm was first proposed by Storn and Price [2, 3]. Like other evolutionary algorithms, an initialization phase is its first task. In addition, it also consists of three major operations: mutation, crossover, and selection. Meanwhile, there exist a few mutation strategies proposed in the work [3]. In order to distinguish the different DE versions with various mutation strategies or different crossover schemes, the famous notation DE/x/y/z was introduced in the literature [3], where x represents the vector to be mutated, y is the number of differential vectors used, and z denotes the crossover scheme employed. DE/rand/1/bin was applied most commonly and it was also usually considered as the canonical DE version. To be specific, the canonical DE version can be described as follows.

2.1. Initialization. At the first step, a population of NP individuals is generated randomly by the following form:

$$x_{ij} = x_j^{\min} + \left(x_j^{\max} - x_j^{\min}\right) \cdot \text{rand}\,(0,1)\,, \tag{1}$$

where $i = 1, 2, \ldots, \text{NP}$, $j = 1, 2, \ldots, D$; $x_j^{\min}$ and $x_j^{\max}$ are the lower bound and upper bounds of the parameter j, respectively. Then, the cost function of each solution is evaluated.

2.2. Mutation. Mutation strategy is very important in DE. At the step, a mutant vector v_i is generated by the following formula for each D-dimensional target vector x_i:

$$v_i = x_a + F \cdot (x_b - x_c)\,, \tag{2}$$

where $i = 1, 2, \ldots, \text{NP}$, $a, b, c \in \{1, 2, \ldots, \text{NP}\}$ are mutually different random integer number, and they are such that $a \neq b \neq c \neq i$. The mutation scale factor F is a real and constant factor $\in [0, 2]$ which controls the amplification of the differential variation $(x_b - x_c)$ [3].

2.3. Crossover. In order to exchange information between a mutant vector v_i and the current target vector x_i, crossover operation is introduced. At this time, a trial vector $u_i = (u_{i1}, u_{i2}, \ldots, u_{iD})$ is produced by the following form:

$$u_{ij} = \begin{cases} v_{ij}, & \text{if rand}\,[0,1]_j \leq \text{Cr} \vee j == j_{\text{rand}}, \\ x_{ij}, & \text{otherwise}, \end{cases} \tag{3}$$

where $j = 1, 2, \ldots, D$, rand$[0, 1]_j$ is a random real number between $[0, 1]$, and $j_{\text{rand}} \in \{1, 2, \ldots, D\}$ is a randomly chosen index, which ensures that the trial vector u_i obtains at least one parameter from the mutant vector v_i. Crossover rate Cr is a predefined constant within the range $[0, 1]$ and it controls the fraction of parameter values copied from the mutant vector.

2.4. Selection. After crossover operation, the trial vector u_i is compared to the target vector x_i through a greedy selection mechanism. The winner is retained and it will become a member of next generation. For a minimization problem, the selection process can be described according to the following equation:

$$x_i^{\star} = \begin{cases} u_i, & \text{if } f(u_i) < f(x_i)\,, \\ x_i, & \text{otherwise}, \end{cases} \tag{4}$$

where $f(x)$ denotes the objective of solution x and $x_i^{\star}$ is an offspring corresponding to the target vector x_i.

In a word, except for the initialization phase, the aforementioned steps will be repeated in turn until a stopping criterion is reached.

3. An Enhanced Differential Evolution Algorithm

3.1. Initialization Based on Opposition-Based Learning. Recently, Rahnamayan et al. [12, 13] proposed a new scheme for generating random numbers, called opposition-based learning (OBL), which can effectively make use of random numbers and their opposites. Moreover, the ability of OBL accelerating the optimization, search, or learning process in many soft computing techniques has been reported in

(1) **for** $i = 1$ to NP **do**
(2) **for** $j = 1$ to D **do**
(3) $x_{i,j} = x_j^{\min} + \text{rand}(0,1) \cdot \left(x_j^{\max} - x_j^{\min}\right)$
(4) $ox_{i,j} = x_j^{\min} + x_j^{\max} - x_{i,j}$ //opposition-based learning
(5) **end for**
(6) **end for**

Algorithm 1: Initialization based on opposition-based learning.

the literatures [12, 13]. At first, a state-of-the-art algorithm, named ODE, was proposed by applying the OBL scheme to accelerate DE [13]. After that, the OBL scheme has been successfully used in other evolutionary algorithms such as artificial bee colony algorithm [40], harmony search algorithm [41], particle swarm optimization [42, 43], and teaching learning based algorithm [44]. A comprehensive survey about the OBL scheme can be found in [45].

In order to improve the solution quality of initial population, the OBL scheme is employed to initialize the population individuals of EDE in the work. The initial process can be described as shown in Algorithm 1. In Algorithm 1, two sets, that is, sets X and OX, are generated, where $X = \{x_1, x_2, \ldots, x_{\text{NP}}\}$ and $OX = \{ox_1, ox_2, \ldots, ox_{\text{NP}}\}$. The initial population consists of the top NP individuals chosen from the set $X \cup OX$ according to their fitness values.

3.2. Multiple Mutation Strategies. A mutation strategy DE/current/1/bin is employed. Namely, the target vector x_i is employed as the base vector in this DE version. That is, a mutant vector v_i will be generated by the following equation:

$$v_i = x_i + F \cdot \left(x_a - x_b\right), \tag{5}$$

where $i \in \{1, 2, \ldots, \text{NP}\}$ represents the index of current individual, $a \in \{1, 2, \ldots, \text{NP}\}$ and $b \in \{1, 2, \ldots, \text{NP}\}$ are random integers, and $a \neq b \neq i$.

In order to better take advantage of the guiding information of best individual, a new version of DE/best/1/bin, DE/$p$best/1/bin, proposed by Zhang and Sanderson [15], is further employed in the work to speed up the convergence speed of the proposed approach EDE. That is, a mutant vector v_i is produced as follows:

$$v_i = x_{\text{best}}^p + F \cdot \left(x_a - x_b\right), \tag{6}$$

where $p \in \{1, 2, \ldots, M\} \subseteq \{1, 2, \ldots, \text{NP}\}$ is a random number and it denotes the top p individuals according to the fitness values of individuals. It should be noted that p of DE/$p$best/1/bin in JADE [15] is a proportional number between $[0, 1]$.

More specifically, according to the first mutation strategy, it can be seen that new generated mutant vectors will be scattered around the respective target vectors, which can not only keep good population diversity but also avoid the overrandomness of classic mutation strategy DE/rand/1/bin. According to the second mutation strategy DE/$p$best/1/bin, owing to the guidance of one of several better individuals (x_{best}^p) rather than the only best individual x_{best}, the used mutation strategy can drive population towards better individuals so as to enhance the convergence speed. In addition, it can also prevent EDE from congregating the vicinity of global best individual to some extent.

In the meantime, a probabilistic parameter r_1 is time varying and designed to control which of the two mutation strategies is to be executed at the mutation step. The parameter r_1 can be described as follows:

$$r_1 = r_{\max} - \frac{\text{FEs}}{\max \text{FEs}} \cdot \left(r_{\max} - r_{\min}\right), \tag{7}$$

where $r_{\max}$ and $r_{\min}$ denote the maximum probability value and the minimum probability value, respectively. FEs is an iterative variable. max FEs represents the maximum number of fitness function evaluations.

As a matter of fact, the probability parameter r_1 plays an important role in balancing the exploration ability and the exploitation ability. That is, it is hoped that good population diversity is kept at the beginning of evolution and fast convergence speed is achieved at the end of search.

3.3. Perturbation. After repeating all operations (mutation, crossover, and select operations) of differential evolution, a perturbed scheme is conducted over the best individual in order to further trade off the searching ability of the aforementioned solution search equations. During the process, two perturbed equations are introduced and the best individual is perturbed dimension by dimension according to them, which are described by (8) and (9), respectively. One has

$$\mu_j = x_{\text{best},n} + (2 \cdot \text{rand}\,(0,1) - 1) \cdot \left(x_{\text{best},n} - x_{k,n}\right), \tag{8}$$

where $j = 1, 2, \ldots, D$, $\mu = (\mu_1, \mu_2, \ldots, \mu_D)$ is a temporary copy of the best individual, best represents the index of best individual in current population, $k \in \{1, 2, \ldots, \text{NP}\} \wedge k \neq \text{best}$ is a uniform random number, and $n \in \{1, 2, \ldots, D\}$ is also a random number. One has

$$\mu_j = x_{\text{best},j} + (2 \cdot \text{rand}\,(0,1) - 1) \cdot \left(x_{\text{best},n} - x_{k,n}\right), \tag{9}$$

where all the notations are the same as those in (8).

From (9), it can be observed that perturbation operation occurs on the current component j of best individual, and the differential variation $(x_{\text{best},n} - x_{k,n})$ acts as perturbed scales.

Notice that dimension n may be different from j, which is helpful to enrich perturbation scales to some extent. That is, it may increase the probability of getting out of local minima trap.

What is more, the term $x_{\text{best},j}$ of (9) is different from the first term on the right hand side of formulation (8). The reason for (8) introduced is that information between different dimensions of best individual could be shared. Thus, the EDE algorithm could get out of local optimal trap with a larger probability.

Like the aforementioned tradeoff scheme, a probability parameter r_2 is employed. The parameter r_2 is linear time-varying during the evolution process as follows:

$$r_2 = w_{\min} + \frac{\text{FEs}}{\max \text{FEs}} \cdot (w_{\max} - w_{\min}), \tag{10}$$

where $w_{\max}$ and $w_{\min}$ denote the maximum probability value and the minimum probability value, respectively. The rest of these parameters are the same as those in (7).

Concretely speaking, (8) is executed with a probability value r_2, but (9) is executed with a probability value $(1 - r_2)$.

3.4. Boundary Constraints Handling Technique. In order to keep solutions subject to boundary constraints, some components of a solution violating the predefined boundary constraints should be repaired. That is, if a parameter value produced by solution search equations exceeds its predefined boundaries, the parameter should be set to an acceptable value. The following repair rule used in the literature [17] is employed in this work:

$$x_{ij} = \begin{cases} x_j^{\min} + \text{rand}(0,1) \cdot \left(x_j^{\max} - x_j^{\min}\right), & \text{if } x_{ij} < x_j^{\min}, \\ x_j^{\max} - \text{rand}(0,1) \cdot \left(x_j^{\max} - x_j^{\min}\right), & \text{if } x_{ij} > x_j^{\max}. \end{cases} \tag{11}$$

3.5. The Proposed Approach. In order to effectively take use of the guidance information of best individual, mutation strategy DE/best/1/bin is considered. In order to prevent a large number of individuals from clustering around the global best individual, inspired by JADE [15], mutation strategy DE/$p$best/1/bin is actually used. In addition, another mutation strategy DE/current/1/bin is employed to further trade off the exploitation ability of DE/$p$best/1/bin. At the same time, a selective probability r_1 with linear time-varying nature is introduced to decide which mutation strategy works at the mutation phase of DE. Subsequently, a perturbation scheme for the best individual is incorporated into the modified DE version. In short, the pseudocode of EDE can be given in Algorithm 2 based on the above explanation.

4. Experimental Study and Discussion

4.1. Benchmark Functions and Parameter Settings. To verify the optimization effectiveness of EDE, twenty-five benchmark functions with different characteristics taken from Yao et al. [46], Gong et al. [23], and Gao and Liu [40] are employed here.

These benchmark functions are listed briefly in Table 1, in which D designates the dimensionality of test functions. All the functions are scalable and high-dimensional problems. Functions f_{01}–f_{05}, f_{14}, and f_{15} are unimodal. Function f_{06}, that is, the step function, has one minimum and is discontinuous. Function f_{07} is a quartic function with noise. Functions f_{08}–f_{13} and f_{16}–f_{19} are difficult multimodal functions where the number of local minima increases exponentially as the dimension of test function increases. In addition, six shifted functions are chosen to evaluate the performance of EDE. Namely, functions f_{20}–f_{25} are shifted functions and $o = (o_1, o_2, \ldots, o_D)$ representing a shifted vector is generated randomly in the corresponding search range.

In our experimental study, all benchmark functions are tested in 30 dimensions and 100 dimensions. The corresponding maximum number of fitness function evaluations (max FEs) is $15e4$ and $50e4$, respectively. Moreover, the other specific parameters of DE and EDE are set as follows.

DE Settings. In canonical DE/rand/1/bin, the scale factor F is set to 0.5, the parameter of crossover rate Cr is set to 0.9, and the population size SN is 100. It should be noted that the values of three parameters are the same as those of the state-of-the-art algorithm ODE [13].

EDE Settings. In our proposed algorithm, the scale factor F is set to 0.5. The parameter of crossover rate Cr is set to 0.9. And the population size SN is 20. A few other parameters are set as follows: $r_{\max} = 1$, $r_{\min} = 0.1$, $w_{\max} = 0.2$, $w_{\min} = 0$, and $M = 4$.

For the set of experiments tested on 25 benchmark functions, we use the aforementioned parameter settings unless a change is mentioned. Furthermore, each test case is optimized thirty runs independently. Then, experimental results for these well-known problems as well as some comparisons with other famous methods are reported as follows.

4.2. Comparison between DE and EDE. For the purpose of validating the enhancing effectiveness of EDE, EDE is first compared with canonical DE in terms of best, worst, median, mean, and standard deviation (Std.) values of solutions achieved by each algorithm in 30 independent runs. The corresponding results are listed in Table 2. Furthermore, the Wilcoxon rank sum test is conducted to compare the significant difference between DE and EDE at $\alpha = 0.05$ significance level. The related test results are also reported in Table 2. And then, some representatives of convergence curves of DE and EDE are shown in Figure 1 in order to show the convergence speed of EDE more clearly.

From Table 2, it can be seen that EDE is significantly superior to DE in most cases. To be specific, EDE is significantly better than DE on 20 functions, that is, f_{01}, f_{02}, f_{03}, f_{04}, f_{05}, f_{06}, f_{08}, f_{09}, f_{10}, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}, f_{19}, f_{20}, f_{21}, f_{22}, f_{24}, and f_{25}, in terms of related Wilcoxon rank sum test results. In addition, for function f_{07} with $D = 30$, EDE is still better than DE. For function f_{07} with $D = 100$, EDE is equal to DE; actually, the mean result achieved by EDE is slightly better than that of DE. For function f_{11} with $D = 30$, EDE is similar

(1) Initialize a population of NP individuals based on the opposition-based learning technique
(2) Set FEs = 2 ∗ NP // FEs represents the iteration counter
// max FEs represents the maximum number of fitness function evaluations
(3) **while** FEs ⩽ max FEs **do**
(4) **for** $i = 1$ to NP **do**
(5) Sort the population from best to worst
(6) Randomly choose a relatively better individual from the top $p \in \{1, 2, \ldots, M\}$ individuals, and let $p$best represent the index of chosen individual
(7) Select uniform randomly $a \neq b \neq i$
(8) **if** rand ⩽ r_1 **then**
(9) Generate a mutant individual v according to (5)
(10) **else**
(11) Generate a mutant individual v according to (6)
$v = x_{p\text{best}} + F \cdot (x_a - x_b)$
(12) **end if**
(13) Let $u = x_i$
//rand is a function for generating a random number in the range of [0, 1]
(14) Let $j$rand = $\lfloor D * \text{rand} \rfloor + 1$
(15) **for** $j = 1$ to D **do**
(16) **if** rand ⩽ Cr ‖ $j ==$ $j$rand **then**
(17) $u_j = v_j$
(18) **end if**
(19) **end for**
(20) Evaluate the new produced individual u
(21) Set FEs = FEs + 1
(22) **if** $f(u)$ is better than $f(x_i)$ **then**
(23) Replace x_i with u
(24) **end if**
(25) **end for**
(26) //Perturb the best individual dimension by dimension
(27) **for** $j = 1$ to D **do**
(28) Set $\mu = x_{\text{best}}$ // best denotes the index of best individual
(29) $k = \lfloor \text{rand} * \text{NP} \rfloor + 1 \wedge k \neq \text{best}$
(30) $n = \lfloor \text{rand} * D \rfloor + 1$
(31) **if** rand < r_2 **then**
(32) Modify μ_j according to (8)
(33) **else**
(34) Modify μ_j according to (9)
(35) **end if**
(36) Evaluate the modified individual μ
(37) Set FEs = FEs + 1
(38) Choose a better individual from the set $\{x_{\text{best}}, \mu\}$ to represent x_{best}
(39) **end for**
(40) Record the best solution found so far
(41) **end while**

Algorithm 2: The EDE algorithm.

to DE. For the function f_{11} with $D = 100$, DE outperforms EDE. Nevertheless, the results obtained by EDE are very close to those found by DE. For the functions f_{17}, f_{18} and f_{23}, DE is better than EDE. And yet, the results obtained by EDE are very close to those found by DE on the functions f_{17}, f_{18} at $D = 30$ and f_{23} at $D = 100$.

From Figure 1, it can also be observed that EDE is far better than DE in terms of solutions accuracy and convergence speed on the representative cases.

According to the aforementioned analyses, it can be concluded that EDE is better than or approximately equal to DE on almost all the functions. In other words, multiple

Table 1: Benchmark functions used in experiments.

Test functions	Search space	Optimum
$f_{01}(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0
$f_{02}(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$	0
$f_{03}(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^D$	0
$f_{04}(x) = \max_i \{ \lvert x_i \rvert, 1 \leqslant i \leqslant D \}$	$[-100, 100]^D$	0
$f_{05}(x) = \sum_{i=1}^{D-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + \left(x_i - 1 \right)^2 \right]$	$[-30, 30]^D$	0
$f_{06}(x) = \sum_{i=1}^{D} \left(\lfloor x_i + 0.5 \rfloor \right)^2$	$[-100, 100]^D$	0
$f_{07}(x) = \sum_{i=1}^{D} i x_i^4 + \text{random}[0, 1)$	$[-1.28, 1.28]^D$	0
$f_{08}(x) = -418.98288727243369 \times D + \sum_{i=1}^{D} \left[-x_i \sin \left(\sqrt{\lvert x_i \rvert} \right) \right]$	$[-500, 500]^D$	0
$f_{09}(x) = \sum_{i=1}^{D} \left(x_i^2 - 10 \cos \left(2\pi x_i \right) + 10 \right)$	$[-5.12, 5.12]^D$	0
$f_{10}(x) = -20 \exp \left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2} \right) - \exp \left(\frac{1}{D} \sum_{i=1}^{D} \cos \left(2\pi x_i \right) \right) + 20 + e$	$[-32, 32]^D$	0
$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i}^{D} \cos \left(\frac{x_i}{\sqrt{i}} \right) + 1$	$[-600, 600]^D$	0
$f_{12}(x) = \frac{\pi}{D} \left\{ 10 \sin^2 \left(\pi y_1 \right) + \sum_{i=1}^{D-1} \left(y_i - 1 \right)^2 \left[1 + 10 \sin^2 \left(\pi y_{i+1} \right) \right] \left(y_n + 1 \right)^2 \right\} + \sum_{i=1}^{D} u \left(x_i, 10, 100, 4 \right),$ where $y_i = 1 + \frac{1}{4}(x_i + 1)$, $u(x_i, a, k, m) = \begin{cases} k \left(x_i - a \right)^m, & x_i > a; \\ 0, & -a \leqslant x_i \leqslant a; \\ k \left(-x_i - a \right)^m, & x_i < -a. \end{cases}$	$[-50, 50]^D$	0
$f_{13}(x) = 0.1 \left\{ \sin^2 \left(3\pi x_1 \right) + \sum_{i=1}^{D} \left(x_i - 1 \right)^2 \left[1 + \sin^2 \left(3\pi x_i + 1 \right) \right] + \left(x_n - 1 \right)^2 \left[1 + \sin^2 \left(2\pi x_n \right) \right] \right\} + \sum_{i=1}^{D} u \left(x_i, 5, 100, 4 \right)$	$[-50, 50]^D$	0
$f_{14}(x) = \sum_{i=1}^{D} i x_i^2$	$[-10, 10]^D$	0
$f_{15}(x) = \sum_{i=1}^{D} i x_i^4$	$[-1.28, 1.28]^D$	0
$f_{16}(x) = \sum_{i=1}^{D} \left[y_i^2 - 10 \cos \left(2\pi y_i \right) + 10 \right]$, where $y_i = \begin{cases} x_i & \text{if } \lvert x_i \rvert < 0.5, \\ \frac{\text{round}(2x_i)}{2} & \text{else } \lvert x_i \rvert \geqslant 0.5. \end{cases}$	$[-5.12, 5.12]^D$	0
$f_{17}(x) = 0.5 + \frac{\sin^2 \sqrt{\sum_{i=1}^{D} x_i^2} - 0.5}{\left(1 + 0.001 \sum_{i=1}^{D} x_i^2 \right)^2}$	$[-100, 100]^D$	0
$f_{18}(x) = -\cos \left(2\pi \lVert x \rVert \right) + 0.1 \lVert x \rVert + 1$, where $\lVert x \rVert = \sqrt{\sum_{i=1}^{D} x_i^2}$	$[-100, 100]^D$	0
$f_{19}(x) = \sum_{i=1}^{D} \lvert x_i \sin \left(x_i \right) + 0.1 x_i \rvert$	$[-10, 10]^D$	0
$f_{20}(x) = \sum_{i=1}^{D} z_i^2$, $z = x - o$	$[-100, 100]^D$	0

Table 1: Continued.

Test functions	Search space	Optimum
$f_{21}(x) = \sum_{i=1}^{D}\left(z_i^2 - 10\cos(2\pi z_i) + 10\right), z = x - o$	$[-5, 5]^D$	0
$f_{22}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right) + 20 + e, z = x - o$	$[-32, 32]^D$	0
$f_{23}(x) = \frac{1}{4000}\sum_{i=1}^{D} z_i^2 - \prod_{i}^{D}\cos\left(\frac{z_i}{\sqrt{i}}\right) + 1, z = x - o$	$[-600, 600]^D$	0
$f_{24}(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} z_j\right)^2, z = x - o$	$[-100, 100]^D$	0
$f_{25}(x) = \sum_{i=1}^{D-1}\left(100\left(z_{i+1} - z_i^2\right)^2 + (z_i - 1)^2\right), z = x - o + 1$	$[-100, 100]^D$	0

mutation strategies and perturbation schemes are beneficial to the performance of EDE.

4.3. Comparison between EDE and Other Three DE Variants. In this subsection, EDE is further compared with some representatives of state-of-the-art DE variants, such as SaDE [14], JADE [15], and SaJADE [23]. Here sixteen test functions are used for the comparison. The related comparison results are listed in Table 3. For a fair comparison, except for the proposed algorithm EDE, the rest of the results reported in Table 3 are directly taken from Gong et al. [23].

From Table 3, it can be seen that EDE is obviously better than JADE on twelve functions, that is, f_{01}, f_{02}, f_{04}, f_{05}, f_{06}, f_{08}, f_{09}, f_{10}, f_{12}, f_{13}, f_{19}, and f_{21}. JADE works better than EDE on four functions. Notice that EDE is just slightly inferior to JADE on the three functions f_{03}, f_{07}, and f_{18}. When compared with SaDE, EDE performs better than it does on thirteen functions. And the results found by EDE are very close to those found by SaDE on other two functions f_{07} and f_{18}. When compared with SaJADE, SaJADE is better than EDE on four functions, but the superiority of SaJADE is not obvious on the three functions f_{05}, f_{07}, and f_{18} except for function f_{11}. Yet EDE is better than or equal to SaJADE on other twelve functions.

It should be pointed out that the results are summarized as $w/t/l$ in the last line of Table 3, which means that EDE wins in w function cases, ties in t cases, and loses in l cases when compared with its competitor. For JADE, SaDE, and SaJADE, they are 12/0/4, 13/0/3, and 11/1/4, respectively. The results show that EDE is superior to or similar to other three approaches on the majority of benchmark functions.

4.4. Comparison among EDE and Two Artificial Bee Colony Algorithms. Artificial bee colony algorithm introduced by Karaboga and Basturk is a relatively new swarm-based optimization algorithm [47]. And it has become a promising technique [48]. Particularly, a modified artificial bee colony algorithm, named MABC, proposed by Gao and Liu [40], is an outstanding representative of many enhanced ABC versions. In order to further demonstrate the superiority of EDE, EDE is compared with standard ABC and MABC on twenty-one functions again. In the experimental study, the maximum number of fitness function evaluations (max FEs) is set to 15*e*4 for all compared algorithms as recommended by Gao and Liu [40].

The further comparison results are given in Table 4. For convenience, besides the data achieved by the EDE algorithm, the rest of the results are gained by Gao and Liu [40] directly.

From Table 4, it is clear that EDE is better than or at least even with ABC on nineteen functions, but ABC only works better than EDE on two functions. EDE is better than or equal to MABC on eighteen functions. MABC also only surpasses EDE on three functions. In addition, the accuracy of solution obtained by EDE is far better than that obtained by ABC on many benchmark functions such as f_{01}, f_{02}, f_{08}, f_{14}, f_{15}, and f_{19}. Meanwhile, the accuracy of solution obtained by EDE is far better than that obtained by MABC on some test functions including f_{01}, f_{02}, and f_{14}. In summary, EDE is superior to both ABC and MABC.

5. Conclusion

In order to achieve a better compromise between the exploration ability and the exploitation ability of DE, in this work, an enhanced differential evolution algorithm, called EDE, is presented. In EDE, first, an initialization technique, opposition-based learning initialization, is employed. Next, inspired by JADE [15], a mutation strategy DE/*p*best/1/bin is introduced in EDE. At the same time, a new mutation strategy DE/current/bin/1 is also introduced. That is, there are multiple mutation strategies composed of the two mutation strategies in EDE to better balance the exploration and the exploitation of DE. When performing the EDE algorithm, one of the two mutation strategies is chosen randomly with a linear time-varying scheme. Last, a perturbation scheme for the best individual is presented in order to get out of local minima, where the perturbation scheme is also composed of two solution search equations. Specifically, the best individual is perturbed dimension by dimension in two modes. All these modifications make up the proposed algorithm EDE.

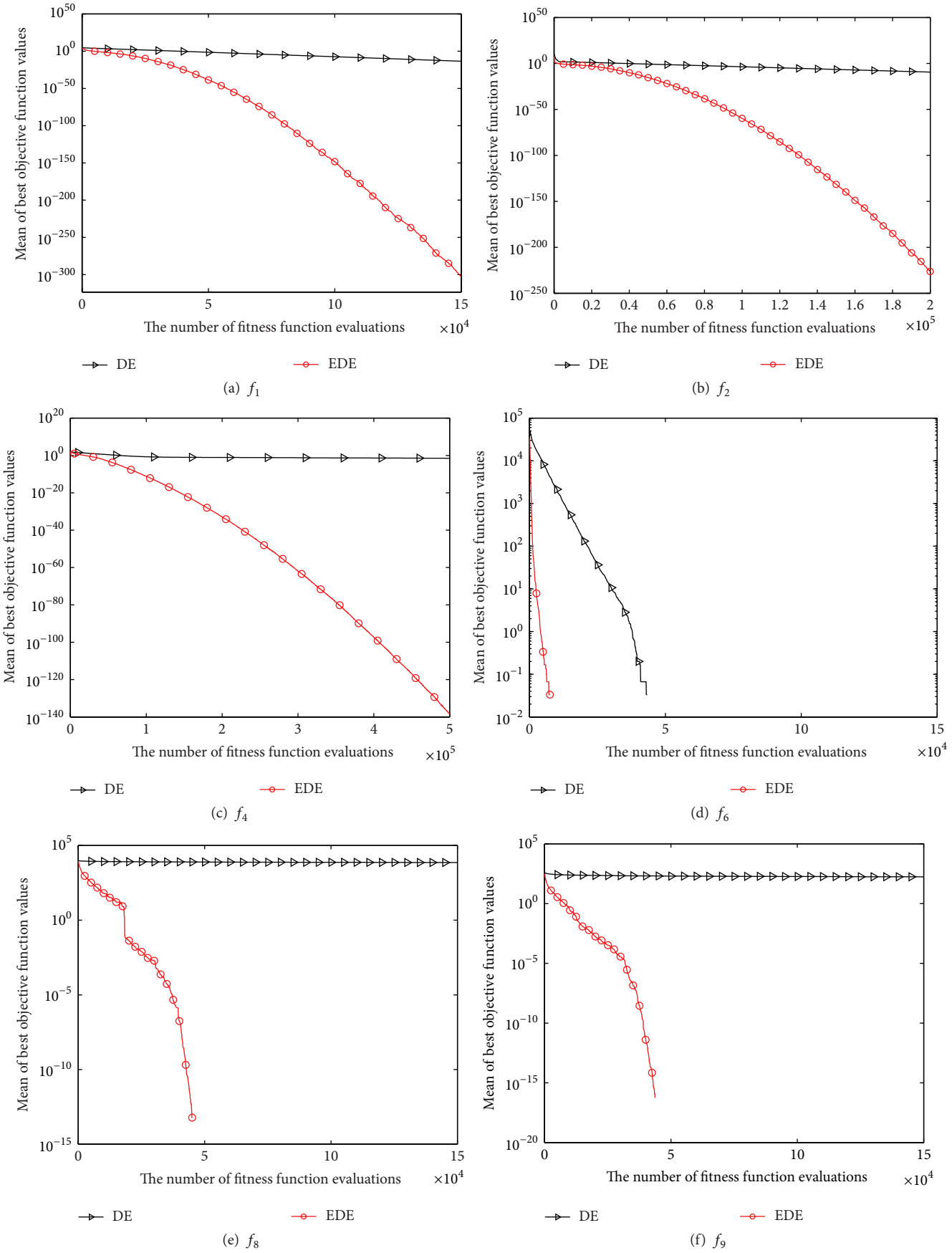


Figure 1: Continued.

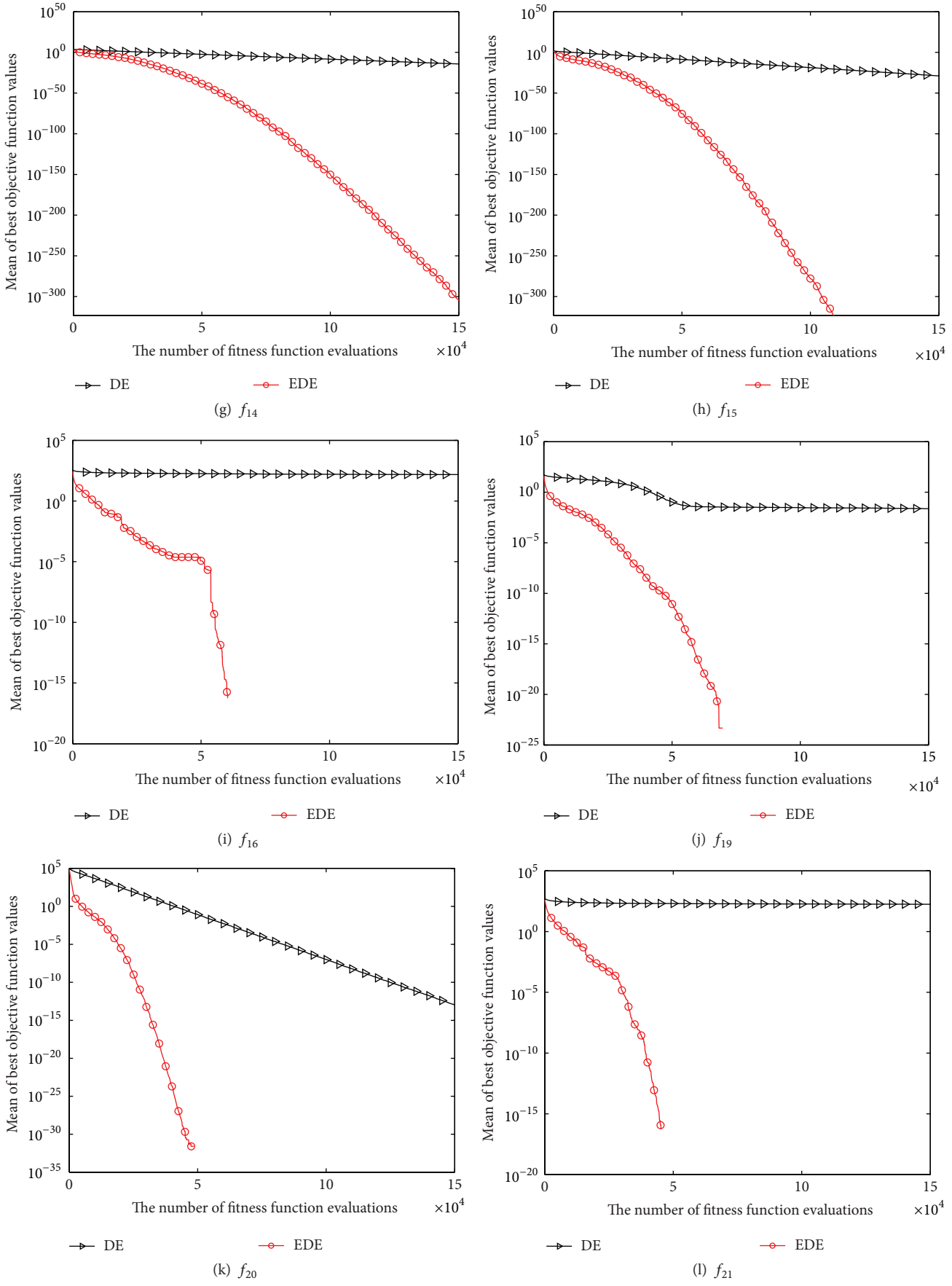


Figure 1: Convergence performance of DE and EDE on the twelve test functions at $D = 30$.

Table 2: Best, worst, median, mean, and standard deviation values achieved by DE and EDE through 30 independent runs.

Number	Dim.	max FEs	Methods	Best	Worst	Median	Mean	Std.	Sig.
f_{01}	30	15e4	DE	1.48e − 014	1.00e − 013	3.38e − 014	3.81e − 014	1.87e − 014	†
			EDE	0.00e − 000	1.13e − 302	1.12e − 315	4.19e − 304	0.00e − 000	
	100	50e4	DE	1.50e − 018	9.93e − 017	7.79e − 018	1.29e − 017	1.86e − 017	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{02}	30	20e4	DE	1.40e − 010	9.17e − 010	3.57e − 010	3.95e − 010	1.93e − 010	†
			EDE	6.18e − 234	1.00e − 225	1.05e − 229	4.65e − 227	0.00e − 000	
	100	50e4	DE	1.39e − 010	6.55e − 010	3.61e − 010	3.71e − 010	1.30e − 010	†
			EDE	8.41e − 229	2.02e − 221	5.42e − 225	7.34e − 223	0.00e − 000	
f_{03}	30	50e4	DE	8.22e − 013	1.93e − 010	2.21e − 011	4.23e − 011	4.84e − 011	†
			EDE	7.63e − 085	7.80e − 079	3.78e − 081	8.01e − 080	1.71e − 079	
	100	50e4	DE	5.27e + 003	2.23e + 004	9.65e + 003	1.01e + 004	3.62e + 003	†
			EDE	3.64e − 004	1.18e − 002	2.50e − 003	3.70e − 003	3.20e − 003	
f_{04}	30	50e4	DE	4.40e − 012	2.50e − 001	1.00e − 003	3.15e − 002	6.32e − 002	†
			EDE	1.43e − 144	2.81e − 138	2.95e − 140	2.79e − 139	6.39e − 139	
	100	50e4	DE	1.01e + 001	2.67e + 001	1.97e + 001	1.97e + 001	3.30e − 000	†
			EDE	3.61e − 027	7.99e − 025	3.92e − 026	1.02e − 025	1.59e − 025	
f_{05}	30	15e4	DE	1.41e + 001	1.84e + 001	1.70e + 001	1.68e + 001	1.06e − 000	†
			EDE	1.52e − 024	2.43e − 001	4.03e − 015	8.50e − 003	4.43e − 002	
		50e4	DE	3.47e − 016	3.98e − 000	1.99e − 013	1.32e − 001	7.27e − 001	†
			EDE	0.00e − 000	7.28e − 027	2.17e − 029	4.42e − 028	1.38e − 027	
	100	50e4	DE	7.79e + 001	1.96e + 002	1.41e + 002	1.33e + 002	3.63e + 001	†
			EDE	1.19e − 004	1.61e + 002	7.44e + 001	5.31e + 001	4.70e + 001	
f_{06}	30	8e3	DE	1.79e + 003	5.38e + 003	4.07e + 003	3.90e + 003	8.52e + 002	
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	†
		15e4	DE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	5e4	DE	4.27e + 002	1.06e + 003	6.83e + 002	6.81e + 002	1.65e + 002	
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	†
		50e4	DE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{07}	30	30e4	DE	2.50e − 003	8.10e − 003	4.40e − 003	4.70e − 003	1.40e − 003	†
			EDE	7.02e − 004	3.80e − 003	2.30e − 003	2.30e − 003	8.87e − 004	
	100	50e4	DE	1.80e − 002	9.09e − 002	2.87e − 002	3.25e − 002	1.34e − 003	≈
			EDE	2.29e − 002	4.25e − 002	3.03e − 002	3.05e − 002	4.90e − 003	
f_{08}	30	15e4	DE	6.56e + 003	7.72e + 003	7.26e + 003	7.26e + 003	2.91e + 002	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	50e4	DE	2.56e + 004	3.30e + 004	2.89e + 004	2.93e + 004	1.84e + 003	†
			EDE	9.45e − 011	9.45e − 011	9.45e − 011	9.45e − 011	0.00e − 000	
f_{09}	30	15e4	DE	1.46e + 002	1.94e + 002	1.77e + 002	1.74e + 002	1.34e + 001	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	50e4	DE	1.91e + 002	6.65e + 002	5.70e + 002	5.49e + 002	1.03e + 002	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{10}	30	15e4	DE	2.82e − 008	2.07e − 007	5.86e − 008	7.10e − 008	3.55e − 008	†
			EDE	4.44e − 015	4.44e − 015	4.44e − 015	4.44e − 015	0.00e − 000	
	100	50e4	DE	2.06e − 010	2.79e − 009	5.90e − 010	6.86e − 010	4.79e − 010	†
			EDE	4.44e − 015	7.99e − 015	7.99e − 015	7.40e − 015	1.34e − 015	
f_{11}	30	15e4	DE	1.16e − 014	7.40e − 003	1.11e − 013	4.93e − 004	1.90e − 003	≈
			EDE	0.00e − 000	7.34e − 002	1.60e − 002	2.09e − 002	2.19e − 002	
	100	50e4	DE	0.00e − 000	5.37e − 002	0.00e − 000	3.20e − 003	1.01e − 002	-
			EDE	0.00e − 000	6.11e − 002	0.00e − 000	1.05e − 002	1.76e − 002	

Table 2: Continued.

Number	Dim.	max FEs	Methods	Best	Worst	Median	Mean	Std.	Sig.
f_{12}	30	15e4	DE	8.61e − 016	2.17e − 014	3.90e − 015	5.28e − 015	4.65e − 015	†
			EDE	1.57e − 032	1.57e − 032	1.57e − 032	1.57e − 032	5.56e − 048	
	100	50e4	DE	5.48e − 019	1.55e − 001	2.46e − 016	8.30e − 003	2.94e − 002	†
			EDE	4.71e − 033	4.71e − 033	4.71e − 033	4.71e − 033	1.39e − 048	
f_{13}	30	15e4	DE	8.68e − 015	1.06e − 013	2.61e − 014	3.55e − 014	2.46e − 014	†
			EDE	1.34e − 032	1.34e − 032	1.34e − 032	1.34e − 032	5.56e − 048	
	100	50e4	DE	2.74e − 017	2.75e + 001	9.82e − 000	1.02e + 001	6.88e − 000	†
			EDE	1.34e − 032	1.34e − 032	1.34e − 032	1.34e − 032	5.56e − 048	
f_{14}	30	15e4	DE	7.58e − 016	1.86e − 014	5.13e − 015	5.71e − 015	4.17e − 015	†
			EDE	0.00e − 000	6.48e − 304	1.00e − 313	2.19e − 305	0.00e − 000	
	100	50e4	DE	2.21e − 019	2.62e − 017	3.41e − 018	4.70e − 018	4.93e − 018	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{15}	30	15e4	DE	5.46e − 031	1.13e − 028	5.45e − 030	1.53e − 029	2.50e − 029	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	50e4	DE	1.58e − 027	3.78e − 024	3.00e − 026	1.97e − 025	6.84e − 025	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{16}	30	15e4	DE	1.18e + 002	1.70e + 002	1.52e + 002	1.51e + 002	1.37e + 001	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	50e4	DE	3.64e + 002	7.47e + 002	6.07e + 002	6.05e + 002	9.01e + 001	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{17}	30	15e4	DE	3.72e − 002	3.72e − 002	3.72e − 002	3.72e − 002	5.43e − 007	-
			EDE	7.82e − 002	2.72e − 001	1.78e − 001	1.63e − 001	5.62e − 002	
	100	50e4	DE	7.82e − 002	1.41e − 001	7.82e − 002	9.29e − 002	2.29e − 002	-
			EDE	4.79e − 001	4.94e − 001	4.90e − 001	4.89e − 001	4.50e − 003	
f_{18}	30	15e4	DE	1.12e − 001	2.00e − 001	1.99e − 001	1.97e − 001	1.60e − 002	-
			EDE	2.99e − 001	6.99e − 001	4.99e − 001	5.03e − 001	1.09e − 001	
	100	50e4	DE	2.99e − 001	3.99e − 001	3.07e − 001	3.39e − 001	4.65e − 002	-
			EDE	2.09e − 000	3.39e − 000	2.79e − 000	2.77e − 000	3.31e − 001	
f_{19}	30	15e4	DE	1.85e − 002	2.87e − 002	2.39e − 002	2.38e − 002	2.70e − 002	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	50e4	DE	2.74e − 010	3.27e − 008	2.84e − 009	4.88e − 009	6.52e − 009	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{20}	30	15e4	DE	1.55e − 014	5.02e − 013	7.96e − 014	1.07e − 013	1.07e − 013	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	50e4	DE	1.51e − 017	1.97e − 016	4.43e − 017	5.78e − 017	3.86e − 017	†
			EDE	0.00e − 000	4.93e − 032	0.00e − 000	6.57e − 033	1.70e − 032	
f_{21}	30	15e4	DE	1.49e + 002	1.95e + 002	1.79e + 002	1.77e + 002	1.18e + 001	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	50e4	DE	2.56e + 002	7.10e + 002	6.09e + 002	5.90e + 002	1.07e + 002	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{22}	30	15e4	DE	3.20e − 008	2.01e − 007	6.56e − 008	7.75e − 008	3.80e − 008	†
			EDE	4.44e − 015	7.99e − 015	7.99e − 015	7.40e − 015	1.34e − 015	
	100	50e4	DE	3.52e − 010	1.42e − 009	7.04e − 010	7.54e − 010	2.85e − 010	†
			EDE	7.99e − 015	7.99e − 015	7.99e − 015	7.99e − 015	0.00e − 000	
f_{23}	30	15e4	DE	1.17e − 013	2.35e − 012	5.77e − 013	7.79e − 013	6.66e − 013	-
			EDE	0.00e − 000	1.29e − 001	9.90e − 003	2.10e − 002	2.70e − 002	
	100	50e4	DE	0.00e − 000	9.90e − 003	0.00e − 000	9.03e − 004	2.80e − 003	-
			EDE	0.00e − 000	3.94e − 002	7.40e − 003	8.50e − 003	1.05e − 002	

Table 2: Continued.

Number	Dim.	max FEs	Methods	Best	Worst	Median	Mean	Std.	Sig.
f_{24}	30	15e4	DE	8.85e − 001	5.87e − 000	1.77e − 000	2.29e − 000	1.30e − 000	†
			EDE	1.62e − 023	2.05e − 019	5.25e − 021	2.05e − 020	4.02e − 020	
	100	50e4	DE	1.93e + 004	5.73e + 004	2.76e + 004	2.90e + 004	6.85e + 003	†
			EDE	6.45e − 004	2.96e − 003	6.20e − 003	7.40e − 003	6.60e − 003	
f_{25}	30	15e4	DE	1.64e + 001	7.61e + 001	1.84e + 001	2.03e + 001	1.05e + 001	†
			EDE	4.23e − 015	7.36e − 001	2.72e − 007	6.93e − 002	1.79e − 001	
	100	50e4	DE	7.78e + 001	3.01e + 002	1.43e + 002	1.54e + 002	5.62e + 001	†
			EDE	2.62e − 005	1.72e + 002	7.59e + 001	6.08e + 001	4.64e + 001	

† indicates that EDE is better than its competitor by the Wilcoxon rank sum test at $\alpha = 0.05$.
- means that EDE is worse than its competitor.
≈ means that there is no significant difference between DE and EDE.

Table 3: Performance comparison between EDE and other three DEs over 30 independent runs for the 16 test functions at $D = 30$, where "$w/t/l$" means that EDE wins in w functions, ties in t functions, and loses in l functions, compared with its competitors.

Number	max FEs	JADE-w	SaDE	SaJADE	EDE
f_{01}	15e4	2.69e − 56 (1.41e − 55)†	3.42e − 37 (3.63e − 37)†	1.10e − 79 (7.52e − 79)†	**4.19e − 304 (0.00e − 000)**
f_{02}	20e4	3.18e − 25 (2.05e − 24)†	3.51e − 25 (2.74e − 25)†	1.35e − 47 (7.53e − 47)†	**4.65e − 227 (0.00e − 000)**
f_{03}	50e4	**6.11e − 81 (1.62e − 80)**‡	1.54e − 14 (4.56e − 14)†	1.77e − 77 (3.39e − 77)†	8.01e − 080 (1.71e − 079)
f_{04}	50e4	5.29e − 14 (2.05e − 14)†	6.39e − 27 (8.27e − 27)†	1.26e − 19 (1.35e − 19)†	**2.79e − 139 (6.39e − 139)**
f_{05}	50e4	1.59e − 01 (7.89e − 01)†	7.98e − 02 (5.64e − 01)†	**1.60e − 30 (6.32e − 30)**‡	4.42e − 028 (1.38e − 027)
f_{06}	1e4	5.62e − 00 (1.87e − 00)†	5.07e + 01 (1.34e + 01)†	**0.00e − 00 (0.00e − 00)**≈	**0.00e − 000 (0.00e − 000)**
f_{07}	30e4	6.14e − 04 (2.55e − 04)‡	2.06e − 03 (5.21e − 04)‡	**4.10e − 04 (1.48e − 04)**‡	2.30e − 003 (8.87e − 004)
f_{08}	10e4	2.62e − 04 (3.59e − 04)†	1.13e − 08 (1.08e − 08)†	6.83e − 07 (2.70e − 06)†	**0.00e − 000 (0.00e − 000)**
f_{09}	10e4	1.33e − 01 (9.74e − 02)†	2.43e − 00 (1.60e − 00)†	1.54e − 01 (2.25e − 01)†	**0.00e − 000 (0.00e − 000)**
f_{10}	5e4	3.35e − 09 (2.84e − 09)†	3.81e − 06 (8.26e − 07)†	1.12e − 12 (1.07e − 12)†	**5.03e − 015 (1.34e − 015)**
f_{11}	5e4	1.57e − 08 (1.09e − 07)‡	2.52e − 09 (1.24e − 08)‡	**0.00e − 00 (0.00e − 00)**‡	2.31e − 002 (2.44e − 002)
f_{12}	5e4	1.67e − 15 (1.02e − 14)†	8.25e − 12 (5.12e − 12)†	2.10e − 23 (6.89e − 23)†	**1.57e − 032 (5.56e − 048)**
f_{13}	5e4	1.87e − 10 (1.09e − 09)†	1.93e − 09 (1.53e − 09)†	3.83e − 21 (1.56e − 20)†	**1.35e − 032 (5.56e − 048)**
f_{18}	30e4	2.00e − 01 (1.63e − 02)‡	**1.56e − 01 (5.01e − 02)**‡	1.76e − 01 (4.28e − 02)‡	4.33e − 001 (8.02e − 002)
f_{19}	30e4	1.87e − 10 (2.09e − 09)†	1.93e − 09 (1.53e − 09)†	3.83e − 21 (1.56e − 20)†	**0.00e − 000** (0.00e − 000)
f_{21}	10e4	1.35e − 00 (6.08e − 01)†	1.46e − 00 (1.02e − 00)†	1.13e − 01 (1.60e − 01)†	**0.00e − 000 (0.00e − 000)**
$w/t/l$		12/0/4	13/0/3	11/1/4	—

† indicates that EDE is better than its competitor.
‡ means that EDE is worse than its competitor.
≈ means that the performance of the corresponding algorithm is even with that of EDE.
Bold entities mean the best results.

To testify the convergence performance of EDE, twenty-five benchmark functions with different characteristics from literatures are employed. The first experimental results demonstrate that EDE significantly enhances the performance of standard DE in terms of the best, worst, median, mean, and standard deviation (Std.) values of final solutions in most cases. Moreover, other two comparisons also show that EDE performs significantly better than or at least highly competitive with other five well-known algorithms, that is, JADE, SaDE, SaJADE, ABC, and MABC, on the majority of the corresponding benchmark functions. Therefore, it can be concluded that EDE is an efficient method and it may be a good alternative for solving complex numerical optimization problems.

Last but not least, it is desirable to further apply the EDE algorithm to deal with other optimization problems such as the training of neural networks, system parameter identification, and data clustering.

Conflict of Interests

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

Table 4: Comparison between EDE and other two ABCs over 30 independent runs on the 21 test functions with $D = 30$ in terms of mean and standard deviation.

Number	max FEs	ABC	MABC	EDE
f_{01}	15e4	5.21e − 010 (2.46e − 010)	9.43e − 032 (6.67e − 032)	**4.19e − 304 (0.00e − 000)**
f_{02}	15e4	1.83e − 006 (4.80e − 007)	2.40e − 017 (9.02e − 018)	**1.24e − 169 (0.00e − 000)**
f_{04}	15e4	1.80e + 001 (2.25e − 000)	1.02e + 001 (1.49e − 000)	**1.65e − 041 (2.79e − 041)**
f_{05}	15e4	4.23e − 001 (4.34e − 001)	6.11e − 001 (4.55e − 001)	**8.50e − 003 (4.43e − 002)**
f_{06}	15e4	**0.00e − 000 (0.00e − 000)**	**0.00e − 000 (0.00e − 000)**	**0.00e − 000 (0.00e − 000)**
f_{07}	15e4	8.74e − 002 (1.77e − 002)	3.71e − 002 (8.53e − 003)	**4.90e − 003 (2.40e − 003)**
f_{08}	15e4	8.86e + 001 (8.62e + 001)	**−1.21e − 013 (4.53e − 013)**	0.00e − 000 (0.00e − 000)[a]
f_{09}	15e4	4.81e − 003 (2.57e − 002)	**0.00e − 000 (0.00e − 000)**	**0.00e − 000 (0.00e − 000)**
f_{10}	15e4	4.83e − 006 (2.12e − 006)	4.13e − 014 (2.17e − 015)	**4.44e − 015 (0.00e − 000)**
f_{11}	15e4	1.61e − 008 (3.99e − 008)	**0.00e − 000 (0.00e − 000)**	2.09e − 002 (2.19e − 002)
f_{12}	15e4	1.39e − 011 (3.82e − 012)	1.90e − 032 (3.70e − 033)	**1.57e − 032(5.56e − 048)**
f_{13}	15e4	1.06e − 009 (4.24e − 010)	2.23e − 031 (1.46e − 031)	**1.34e − 032(5.56e − 048)**
f_{14}	15e4	2.22e − 011 (1.14e − 011)	2.10e − 032 (1.56e − 032)	**2.19e − 305 (0.00e − 000)**
f_{15}	15e4	5.51e − 029 (6.70e − 029)	1.45e − 067 (2.28e − 067)	**0.00e − 000 (0.00e − 000)**
f_{16}	15e4	1.12e − 001 (2.97e − 001)	**0.00e − 000 (0.00e − 000)**	**0.00e − 000 (0.00e − 000)**
f_{17}	15e4	4.41e − 001 (1.81e − 002)	2.95e − 001 (3.17e − 002)	**1.63e − 001 (5.62e − 002)**
f_{19}	15e4	7.66e − 005 (2.76e − 005)	1.58e − 016 (2.48e − 016)	**0.00e − 000 (0.00e − 000)**
f_{20}	15e4	1.55e − 009 (5.54e − 010)	**0.00e − 000 (0.00e − 000)**	**0.00e − 000 (0.00e − 000)**
f_{21}	15e4	1.49e − 001 (3.55e − 001)	**0.00e − 000 (0.00e − 000)**	**0.00e − 000 (0.00e − 000)**
f_{22}	15e4	9.73e − 005 (5.69e − 005)	4.92e − 014 (5.31e − 015)	**7.40e − 015 (1.34e − 015)**
f_{23}	15e4	4.93e − 004 (2.25e − 003)	**0.00e − 000 (0.00e − 000)**	2.10e − 002 (2.70e − 002)
$w/t/l$		18/1/2	13/5/3	—

Bold entities mean the best results.

Here "a" means that the results obtained by EDE are set to zero on the function f_8 when the results are less than 1e − 308. This is the reason that the coefficient −418.98288727243369 with low precision in function f_8 may result in the negative results. As a matter of fact, the results should be zero.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant nos. 61064012, 61164003, 61263027, 61364026, and 61563028), the Natural Science Foundation of Gansu Province (Grant no. 148RJZA030), New Teacher Project of Research Fund for the Doctoral Program of Higher Education of China (Grant no. 20126204120002), and the Science and Technology Foundation of Lanzhou Jiaotong University (Grant no. ZC2014010).

References

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

[2] R. Storn and K. Price, "Differential evolution—a simple and efficient adaptive scheme for global opitmization over continuous spaces," Report TR-95-012, 1995.

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

[4] M. M. Ali and A. Törn, "Population set-based global optimization algorithms: some modifications and numerical studies," *Computers & Operations Research*, vol. 31, no. 10, pp. 1703–1725, 2004.

[5] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, no. 6, pp. 448–462, 2005.

[6] J. Sun, Q. Zhang, and E. P. Tsang, "DE/EDA: a new evolutionary algorithm for global optimization," *Information Sciences*, vol. 169, no. 3-4, pp. 249–262, 2005.

[7] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[8] P. Kaelo and M. M. Ali, "A numerical study of some modified differential evolution algorithms," *European Journal of Operational Research*, vol. 169, no. 3, pp. 1176–1184, 2006.

[9] M. M. Ali, "Differential evolution with preferential crossover," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1137–1147, 2007.

[10] Y.-J. Wang and J.-S. Zhang, "Global optimization by an improved differential evolutionary algorithm," *Applied Mathematics and Computation*, vol. 188, no. 1, pp. 669–680, 2007.

[11] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 107–125, 2008.

[12] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition versus randomness in soft computing techniques," *Applied Soft Computing Journal*, vol. 8, no. 2, pp. 906–918, 2008.

[13] R. S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.

[14] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[15] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–928, 2009.

[16] Z. Yang, K. Tang, and X. Yao, "Scalability of generalized adaptive differential evolution for large-scale continuous optimization," *Soft Computing*, vol. 15, no. 11, pp. 2141–2155, 2011.

[17] W. Gong, Z. Cai, and C. X. Ling, "DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization," *Soft Computing*, vol. 15, no. 4, pp. 645–665, 2011.

[18] M. Weber, F. Neri, and V. Tirronen, "A study on scale factor in distributed differential evolution," *Information Sciences*, vol. 181, no. 12, pp. 2488–2511, 2011.

[19] Y. Wang, Z. X. Cai, and Q. F. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.

[20] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, no. 2, pp. 1679–1696, 2011.

[21] A. Ghosh, S. Das, A. Chowdhury, and R. Giri, "An improved differential evolution algorithm with fitness-based adaptation of the control parameters," *Information Sciences*, vol. 181, no. 18, pp. 3749–3765, 2011.

[22] W. Gong, Á. Fialho, Z. Cai, and H. Li, "Adaptive strategy selection in differential evolution for numerical optimization: an empirical study," *Information Sciences. An International Journal*, vol. 181, no. 24, pp. 5364–5386, 2011.

[23] W. Y. Gong, Z. H. Cai, C. X. Ling, and H. Li, "Enhanced differential evolution with adaptive strategies for numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 2, pp. 397–413, 2011.

[24] M. Weber, F. Neri, and V. Tirronen, "Shuffle or update parallel differential evolution for large-scale optimization," *Soft Computing*, vol. 15, no. 11, pp. 2089–2107, 2011.

[25] A. P. Piotrowski, J. J. Napiorkowski, and A. Kiczko, "Differential evolution algorithm with separated groups for multi-dimensional optimization problems," *European Journal of Operational Research*, vol. 216, no. 1, pp. 33–46, 2012.

[26] S. Ghosh, S. Das, A. V. Vasilakos, and K. Suresh, "On convergence of differential evolution over a class of continuous functions with unique global optimum," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 1, pp. 107–124, 2012.

[27] E. Mezura-Montes, M. E. Miranda-Varela, and R. del Carmen Gómez-Ramón, "Differential evolution in constrained numerical optimization: an empirical study," *Information Sciences*, vol. 180, no. 22, pp. 4223–4262, 2010.

[28] H. Liu, Z. Cai, and Y. Wang, "Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization," *Applied Soft Computing Journal*, vol. 10, no. 2, pp. 629–640, 2010.

[29] D. Zou, H. Liu, L. Gao, and S. Li, "A novel modified differential evolution algorithm for constrained optimization problems," *Computers and Mathematics with Applications*, vol. 61, no. 6, pp. 1608–1623, 2011.

[30] W. Gong and Z. Cai, "An improved multiobjective differential evolution based on Pareto-adaptive ϵ-dominance and orthogonal design," *European Journal of Operational Research*, vol. 198, no. 2, pp. 576–601, 2009.

[31] X. Li and M. Yin, "An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure," *Advances in Engineering Software*, vol. 55, pp. 10–31, 2013.

[32] M. Fatih Tasgetiren, Q.-K. Pan, P. N. Suganthan, and O. Buyukdagli, "A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem," *Computers and Operations Research*, vol. 40, no. 7, pp. 1729–1743, 2013.

[33] R. Zhang, S. Song, and C. Wu, "A hybrid differential evolution algorithm for job shop scheduling problems with expected total tardiness criterion," *Applied Soft Computing Journal*, vol. 13, no. 3, pp. 1448–1458, 2013.

[34] A. Nobakhti and H. Wang, "A simple self-adaptive differential Evolution algorithm with application on the ALSTOM gasifier," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 350–370, 2008.

[35] Y. Liu and F. Sun, "A fast differential evolution algorithm using k-Nearest Neighbour predictor," *Expert Systems with Applications*, vol. 38, no. 4, pp. 4254–4258, 2011.

[36] L. Wang, X. Fu, Y. Mao, M. Ilyas Menhas, and M. Fei, "A novel modified binary differential evolution algorithm and its applications," *Neurocomputing*, vol. 98, pp. 55–75, 2012.

[37] Y. Tang, X. Zhang, C. Hua, L. Li, and Y. Yang, "Parameter identification of commensurate fractional-order chaotic system via differential evolution," *Physics Letters A*, vol. 376, no. 4, pp. 457–464, 2012.

[38] H.-C. Lu, M.-H. Chang, and C.-H. Tsai, "Parameter estimation of fuzzy neural network controller based on a modified differential evolution," *Neurocomputing*, vol. 89, pp. 178–192, 2012.

[39] F. Fabris and R. A. Krohling, "A co-evolutionary differential evolution algorithm for solving min-max optimization problems implemented on GPU using C-CUDA," *Expert Systems with Applications*, vol. 39, no. 12, pp. 10324–10333, 2012.

[40] W.-F. Gao and S.-Y. Liu, "A modified artificial bee colony algorithm," *Computers & Operations Research*, vol. 39, no. 3, pp. 687–697, 2012.

[41] A. K. Qin and F. Forbes, "Dynamic regional harmony search with opposition and local learning," in *Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference (GECCO '11)*, pp. 53–54, Dublin, Ireland, July 2011.

[42] W.-f. Gao, S.-Y. Liu, and L.-l. Huang, "Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 11, pp. 4316–4327, 2012.

[43] N. Dong, C.-H. Wu, W.-H. Ip, Z.-Q. Chen, C.-Y. Chan, and K.-L. Yung, "An opposition-based chaotic GA/PSO hybrid algorithm and its application in circle detection," *Computers and*

Mathematics with Applications, vol. 64, no. 6, pp. 1886–1902, 2012.

[44] P. K. Roy, C. Paul, and S. Sultana, "Oppositional teaching learning based optimization approach for combined heat and power dispatch," *International Journal of Electrical Power & Energy Systems*, vol. 57, pp. 392–403, 2014.

[45] Q. Xu, L. Wang, N. Wang, X. Hei, and L. Zhao, "A review of opposition-based learning from 2005 to 2012," *Engineering Applications of Artificial Intelligence*, vol. 29, pp. 1–12, 2014.

[46] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[47] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[48] D. Karaboga and B. Akay, "A comparative study of Artificial Bee Colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2015, Article ID 147843, 10 pages
http://dx.doi.org/10.1155/2015/147843

Research Article

Feed-Forward Neural Network Soft-Sensor Modeling of Flotation Process Based on Particle Swarm Optimization and Gravitational Search Algorithm

Jie-Sheng Wang and Shuang Han

School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, Liaoning 114044, China

Correspondence should be addressed to Jie-Sheng Wang; wang_jiesheng@126.com

Received 18 March 2015; Accepted 25 May 2015

Academic Editor: Adel Elmaghraby

For predicting the key technology indicators (concentrate grade and tailings recovery rate) of flotation process, a feed-forward neural network (FNN) based soft-sensor model optimized by the hybrid algorithm combining particle swarm optimization (PSO) algorithm and gravitational search algorithm (GSA) is proposed. Although GSA has better optimization capability, it has slow convergence velocity and is easy to fall into local optimum. So in this paper, the velocity vector and position vector of GSA are adjusted by PSO algorithm in order to improve its convergence speed and prediction accuracy. Finally, the proposed hybrid algorithm is adopted to optimize the parameters of FNN soft-sensor model. Simulation results show that the model has better generalization and prediction accuracy for the concentrate grade and tailings recovery rate to meet the online soft-sensor requirements of the real-time control in the flotation process.

1. Introduction

Flotation is known as froth flotation, and it is a physico-chemical reaction process. Flotation is the process which is based on the differences of the surface property of solid materials to separate useful minerals and gangue by means of the buoyancy of air bubbles from ore pulp by this method to improve the concentrate grade [1]. In the production process of flotation, concentrate grade and other economic and technical indicators are key control indicators of the production process. Process control indicators of domestic flotation process are mainly based on an experienced operator to observe the information (such as foam color, size, flow rate, and texture features) which is provided by the bubble state formed on the surface of the flotation tank and to adjust the flotation level and change agents system [2, 3]. Inference estimate (soft-sensor) technology can effectively solve the online estimation problems where the flotation process is difficult to online measure the economic and technical indicators.

Domestic and foreign scholars carry through the research on soft-sensor modeling of the key technical indicators in the flotation process and make a lot of achievements [4–16]. Hargrave and Hall study the diagnosis and analysis methods of the metal grade, quality, and flow rate in flotation process by using the color and surface tissue [4]. Bartolacci et al. use multivariate image analysis (MIA) and partial least squares (PLS) methods to establish the experience prediction model of flotation grade [5]. Morar et al. utilize the machine vision method to predict the performance of the flotation process, such as concentrate grade and tailings recovery rate [6]. Moolman and many other scholars created a bubble dynamic model based on image processing through researching flotation foam structure and calculated the content of useful minerals in foam through this model [7].

At home, Yang et al. put forward a bubble image segmentation method based on the clustering presplit and the accuracy distance reconstruction [8]. In that the soft-sensor method with multiple models can improve the overall prediction accuracy and have the characteristic of robustness;

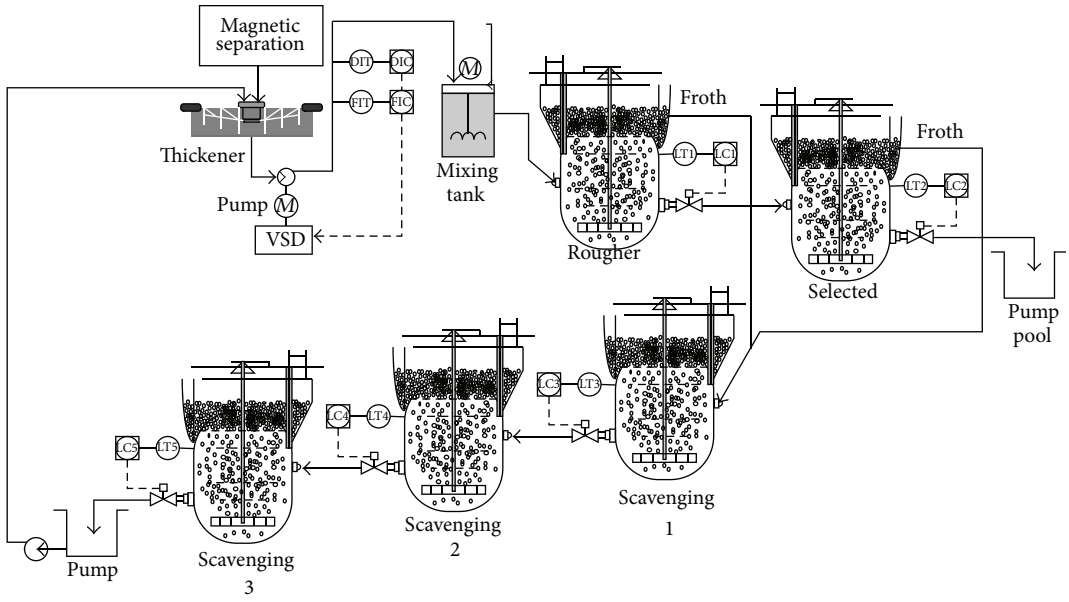


Figure 1: Technique flowchart of flotation process.

Wang et al. present a multi-T-S fuzzy neural network soft-sensor model of flotation process based on the FCM clustering algorithm [9]. Yang et al. use the flotation froth video image features as auxiliary variables and establish a soft-sensor model of the flotation pulp pH value based on the sparse polynuclear least squares support vector machine (SVM) and use Schmidt orthogonalization theory to reduce the multinuclear matrix [10]. Li et al. set up a soft-sensor mode by combining the principal component analysis (PCA) and extreme learning machine (ELM) methods [11]. Zhou et al. extracted color and size characteristics of the foam by using digital image processing method and established a recovery prediction model [12]. Wang and Zhang proposed a kind of soft-sensor model of economic and technical index based on PCA and ANFIS and combining PSO algorithm with LSM put forward a new learning process to optimize parameters of ANFIS [13]. Geng and Chai utilized least squares support vector machine to establish soft-sensor model of concentrate grade and tailing grade in the flotation process based on analyzing related influencing factors of concentrate grade and tailing grade of the flotation process technology indicators [14]. Wang et al. proposed the features extraction method of flotation froth images and BP neural network soft-sensor model of concentrate grade optimized by shuffled cuckoo searching algorithm [15]. Wang et al. proposed an echo state network (ESN) based fusion soft-sensor model optimized by the improved glowworm swarm optimization (GSO) algorithm. Simulation results show that the model has better generalization and prediction accuracy [16].

This paper proposes a feed-forward neural network (FNN) soft-sensor model by using process datum in the flotation process for predicting the flotation concentrate grade and recovery rate, which is optimized by the PSO-GSA algorithm. Simulation results verify the validity of the proposed soft-sensor model. The paper is organized as follows. In Section 2, the technique flowchart of flotation process is introduced. The FNN soft-sensor model of flotation process optimized by PSO-GSA algorithm is presented in Section 3. In Section 4, experiment and simulation results are introduced in detail. Finally, the conclusion illustrates the last part.

2. Technique Flowchart of Flotation Process

Flotation process is used to separate useful minerals and gangue based on the differences of the surface property of solid materials. Figure 1 is a typical iron ore flotation process consisting of the roughing, concentration, and scavenging [11]. The system input is the fine concentrate pulp which is early output of beneficiation process in the forepart. The pulp density is about 38% and concentrate grade is about 64%. Inlet pulp is fed into the high-stirred tank through the pulp pipeline by feed pump. At the same time, the flotation reagent according to a certain concentration ratio is also fed into high-stirred tank through dosing pump. On the other hand, the pulp temperature must reach a suitable flotation temperature by heating. If the dosage is appropriate, the flotation cells can output a grade of 68.5%–69.5% concentrate [15, 16].

The control objective of flotation process is to ensure the concentrate grade and the tailings recovery rate are within a certain target range. In common, based on the offline artificial laboratory to get grade values, the operators adjust the flotation cell level and the amount of flotation reagent addition. Due to the artificial laboratory for two hours at a time, when the process variables and boundary conditions change in the flotation process, they cannot timely adjust the flotation operation variables, which results in such phenomena that the flotation concentrate grade and the tailings recovery rate are too high or too low [15, 16].

Table 1: Soft-sensor modeling data of flotation process.

Number	Auxiliary variables					Dominant variables	
	Feed concentration (%)	Feed flow (m^3/h)	Feed grade (%)	Feed granularity (%)	Medicament flow (L/min)	Concentrate grade (%)	Recovery rate (%)
1	62.76	329	35	90	15.5	70.51	97.7
2	63.67	297	35	90	11.5	69.74	97.2
3	65.07	285	37	92	11.3	69.69	97.0
4	65.48	214	36	95	7.5	68.98	93.5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
600	65.9	310	36	96	5.5	67.29	90.2

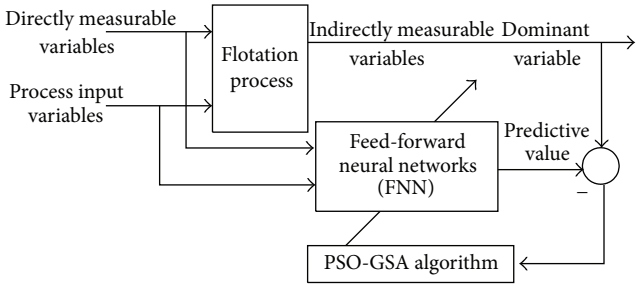


Figure 2: Structure of the proposed soft-sensor model of flotation process.

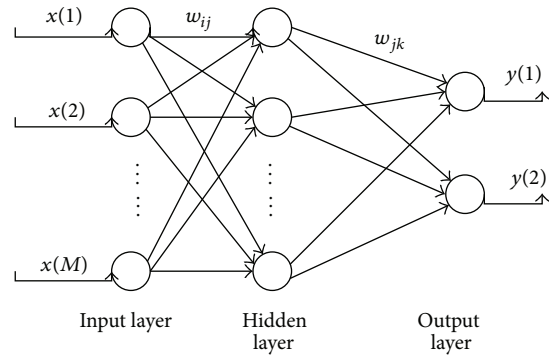


Figure 3: Structure of feed-forward neural network (FNN).

By analyzing the flotation technique, the process variables and boundary conditions mainly include feed grade x_1, feed flow rate x_2, feed concentration x_3, feed granularity x_4, and medicament flow rate x_5. The modeling data is shown in Table 1.

3. Soft-Sensor Modeling of Flotation Process

3.1. Structure of Soft-Sensor Model. The structure of the proposed feed-forward neural network (FNN) soft-sensor model of the flotation process based on PSO-GSA algorithm is shown in Figure 2.

The auxiliary variables of soft-sensor model proposed in this paper are feed grade, feed concentration, feed flow, feed granularity, and medicament flow. Then the samples composed of the auxiliary variable are normalized as the model input. Finally, parameters of FNN soft-sensor model are optimized by PSO-GSA algorithm. Thereby, the accurate prediction of concentrate grade and tailing recovery rate of the flotation process is achieved. Considering a multi-input single-output (MISO) system, the training set can be represented as $D = \{Y, X_i \mid i = 1, 2, \ldots, m\}$, where Y represents the output and X_i denotes the ith input vector and it can be represented as $X_i = [x_{1i}, x_{2i}, \ldots, x_{ni}]'$ (n is the number of the training samples and m is the number of the input variables). The establishment of soft-sensor model needs a data set from the normal working condition as the modeling data. Assuming the m process variables, n data vector samples comprise the test data matrix $X \in R^{n \times m}$. In order to avoid the effect of different dimensions of process variables for results and be convenient for mathematical handling, it is necessary to normalize the data. Assume the mean vector of X is μ and the standard deviation vector is σ. The process variables after the normalization are described as follows:

$$\widehat{X} = \frac{(X - \mu)}{\sigma}. \tag{1}$$

Then the input vectors $\widehat{X}$ of the training samples are fed into FNN soft-sensor model to obtain the predictive output $\widehat{Y}$. On the other hand, the root mean square error (RMSE) is adopted as the fitness value of soft-sensor model:

$$\text{RMSE} = \sqrt{\sum_{k=1}^{n} \frac{\left(\widehat{Y}_k - Y_k^*\right)^2}{n}}, \tag{2}$$

where Y^* is the actual output of the training samples.

3.2. Feed-Forward Neural Network (FNN). In accordance with different layers of the feed-forward neural network (FNN), it can be divided into single-layer feed-forward neural network and multilayer feed-forward neural network. The multilayer FNN is adopted in this paper, which includes an input layer, a hidden layer, and an output layer, whose structure is shown in Figure 3.

Assume that the input layer has M inputs; i ($i = 1, 2, \ldots, M$) represents any input. The hidden layer has N inputs; j ($j = 1, 2, \ldots, N$) represents any inputs. The connection weight values between the input layer and the hidden layer are w_{ij} ($i = 1, 2, \ldots, M$; $j = 1, 2, \ldots, N$). The connection weight values between the hidden layer and the output layer are w_{jk} ($k = 1, 2$). Assume that the input of the hidden layer neurons is u_j; output is v_j. Input of output layer neurons is o_k; total output is y_k. So the calculation of FNN can be represented as follows:

$$\begin{aligned} u_j &= \sum_{i=1}^{M} w_{ij} x(i), \\ v_j &= f\left[u_j\right] = f\left[\sum_{i=1}^{M} w_{ij} x(i)\right], \\ o_k &= \sum_{j=1}^{N} w_{jk} v_j, \\ y_k &= f\left[o_k\right] = f\left[\sum_{j=1}^{N} w_{jk} v_j\right], \end{aligned} \tag{3}$$

where $f()$ represents the transfer function between input and output of the hidden layer and the output layer, which is also called the activation function.

3.3. FNN Soft-Sensor Model Optimized by Hybrid PSO-GSA Algorithm

3.3.1. Particle Swarm Optimization (PSO) Algorithm. Particle swarm optimization (PSO) algorithm is a kind of swarm intelligent optimization algorithm, which is inspired by the birds' migration and swarming behavior during the foraging process. Due to its simplicity, it has been widely used in many optimization problems. It uses a large number of particles (potential solutions) to find the best solution in the search space, where each particle corresponds to a fitness value, and the velocity of the particles is decided by their flight direction and distance. Adjust the individual best value and the global best value to meet the requirements dynamically [17, 18].

Assume that, in an S-dimensional searching space, n particles consist of the population $X = (x_1, x_2, \ldots, x_n)$, where the ith particle is expressed as an S-dimensional vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{is})^T$. It represents the position of the ith particle in the searching space (the potential solutions of the discussed problem). $v_i = (v_{i1}, v_{i2}, \ldots, v_{is})^T$ represents the velocity of the ith particle, $pbest_i = (p_{i1}, p_{i2}, \ldots, p_{is})^T$ represents the individual best value, and $gbest = (g_1, g_2, \ldots, g_s)^T$ represents the global best value. According to the following equations, the position and velocity vector of the particles are updated:

$$\begin{aligned} v_i^{t+1} &= w v_i^t + c_1 \times \text{rand} \times \left(pbest_i - x_i^t\right) + c_2 \times \text{rand} \\ &\quad \times \left(gbest - x_i^t\right), \end{aligned} \tag{4}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}, \tag{5}$$

where w is the inertia weight, t is the number of iterations, v_i^t represents the velocity of ith particle in t iteration, c_1 is the particle's acceleration weighting coefficient, c_2 is a global acceleration weighting coefficient, c_1 and c_2 are learning factors (usually $c_1 = c_2 = 2$), rand is a random number between 0 and 1, and x_i^t represents the current location of the ith particle in tth iteration.

The first part wv of (4) represents the search capability of PSO algorithm, and the second part $c_1 \times \text{rand} \times (pbest_i - x_i^t)$ and the third part $c_2 \times \text{rand} \times (gbest - x_i^t)$ represent individual and global optimization ability of particles. In the searching space, the position vector of the particles is randomly generated. In each iteration, (4) is used to update the velocity vector of particles. After determining the velocity of the particles, the particle position vector is updated by (5). The position vector of the particles will be constantly changed until the termination condition is satisfied.

3.3.2. Gravitational Search Algorithm (GSA). In 2009, Rashedi et al. put forward a gravitational search algorithm (GSA), which is a heuristic optimization algorithm [19]. It uses the physics law to find the optimal solution in the searching space. Inspiration of the GSA comes from Newton's universal gravitation law. Gravity is a force of attraction that exists between any two masses, any two bodies, or any two particles. The size of the gravitational force is proportional to their product of the quality and inversely proportional to the distance between them. In this algorithm, each individual represents a potential solution, each of the potential solutions corresponds to a fitness value, and the fitness value is represented by the quality of the individual. Everything with massive particle in the universe attracts all other massive particles; a large mass of individuals is subject to greater gravitation. So a large mass of individuals is near to the global best value. The flowchart of gravitation search algorithm is described as in Figure 4.

Suppose, in an n-dimensional searching space, N substances constitute a population. Each individual's position (potential solution) is defined as follows:

$$x_i = \left(x_i^1, \ldots, x_i^d, \ldots, x_i^n\right), \quad i = 1, 2, \ldots, N, \tag{6}$$

where x_i^d is the position of substance i in d-dimension of the space.

In the searching space, all individuals are randomly placed in the tth generation. So the gravity of substance j attracting substance i in d-dimensional space is defined as

$$F_{ij}^d(t) = G(t) \frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon} \left(x_j^d(t) - x_i^d(t)\right), \tag{7}$$

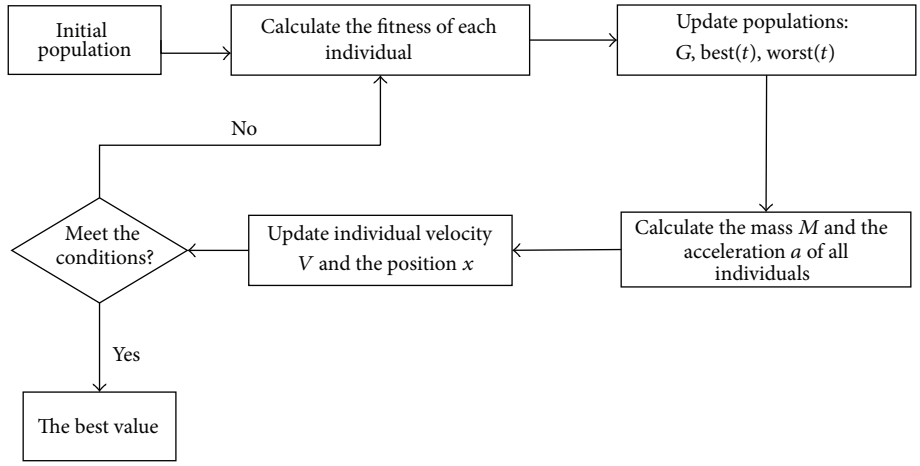


Figure 4: Flowchart of gravitation search algorithm.

where M_{aj} represents the active gravitational mass of individual j, M_{pi} represents the passive gravitational mass of individual i, $G(t)$ represents the gravitational constant in tth generation, ε is a small constant, and $R_{ij}(t)$ represents the Euclidean distance between substance i and substance j.

The gravitational constant G and the Euclidean distance between substance i and substance j are calculated as follows:

$$G(t) = G_0 \times \exp\left(-\alpha \times \frac{\text{iter}}{\text{maxiter}}\right), \qquad R_{ij} = \left\|x_i(t), x_j(t)\right\|_2, \tag{8}$$

where α is the decreasing coefficient (constant), G_0 represents the initial gravitational constant, iter represents the number of current iterations, and maxiter represents the number of maximum iterations.

In the d-dimensional searching space, all gravity which acts on the material i is calculated as follows:

$$F_i^d(t) = \sum_{j=1, j\neq i}^{N} \text{rand}_j F_{ij}^d(t), \tag{9}$$

where rand_j is a random number between 0 and 1.

According to Newton's motion law, the acceleration of the material i is proportional to force in d dimension and inversely proportional to the inverse of the mass. The acceleration of substance is calculated as follows:

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)'}, \tag{10}$$

where t represents the number of current iterations and M_i represents the mass of substance i. Speed and position of substance i are updated by the following equations:

$$v_i^d(t+1) = \text{rand}_i \times v_i^d(t) + a_i^d(t), \tag{11}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1), \tag{12}$$

where rand_i is a random number between 0 and 1.

It can be seen, from the above two equations, that the current speed of a substance is defined as the part of the final speed ($0 \leq \text{rand}_i \leq 1$) and its acceleration. The current position of substance is equal to its final velocity and the current position. The fitness represents the quality of the material, which means that the greater the quality of the material, the higher the efficiency. According to the above formula, the heavier the material, the greater the gravity and the slower the movement. The quality of materials is updated by using the following equation:

$$m_i(t) = \frac{\text{fit}_i(t) - \text{worst}(t)}{\text{best}(t) - \text{worst}(t)}, \tag{13}$$

where $\text{fit}_i(t)$ represents the fitness value of substance i in tth generation, best(t) represents the best individual fitness value in tth generation, and worst(t) represents the worst individual fitness value. With regard to the minimization problem, best(t) and worst(t) are calculated as follows:

$$\text{best}(t) = \min_{j\in\{1,\ldots,N\}} \text{fit}_j(t), \qquad \text{worst}(t) = \max_{j\in\{1,\ldots,N\}} \text{fit}_j(t). \tag{14}$$

The standardization of quality is defined by the following formula:

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)}. \tag{15}$$

3.3.3. PSO-GSA Algorithm. Although GSA has better optimization capability, the material appeared to have low convergence in the process of moving to the optimal value and to be easy to fall into local optimum phenomenon. So PSO algorithm is used to update the position and velocity of the individual in order to make up this shortcoming of GSA. The basic idea of PSO-GSA is described as follows. Firstly, generate the initial position vector $X = (x_1, x_2, \ldots, x_n)$ and velocity vector $v_i = (v_{i1}, v_{i2}, \ldots, v_{is})^T$ of N individuals

randomly. According to the initial positions of all individuals, calculate the fitness value corresponding to each individual and record the best fitness value best(t) and the worst fitness value worst(t) and the corresponding position vector of the individuals. The quality of individuals is calculated by (13)–(15). Then calculate the gravitational constant G and the Euclidean distance between two individuals and the individual's gravitation F and acceleration a. At this time, according to (11)-(12), the global search ability of PSO algorithm is used to update velocity and position of the individual, and then the fitness value and the corresponding optimal value are calculated. The best individual is obtained until the number of iterations is reached. Consider

$$v_i(t+1) = w \times v_i(t) + c_1' \times \text{rand} \times ac_i(t) + c_2' \times \text{rand} \times \left(gbest - x_i(t)\right), \tag{16}$$

where $v_i(t)$ represents the velocity of the material i in tth generation, c_j' is an acceleration coefficient, w is the inertia weight, rand represents a random number between 0 and 1, $ac_i(t)$ indicates the acceleration of substance i in tth generation, and $gbest$ represents the optimal solution so far.

After updating the velocity vector, the location vector of substance is updated based on the following equation:

$$x_i(t+1) = x_i(t) + v_i(t+1). \tag{17}$$

3.3.4. Procedure of PSO-GSA Algorithm Optimizing FNN. In this paper, the PSO-GSA hybrid algorithm is applied to optimize the parameters of the FNN soft-sensor model, whose aim is to improve the convergence speed and predictive accuracy. PSO-GSA algorithm is different from GSA. It adopts PSO algorithm to update its velocity and position, until it reaches the number of iterations or accuracy. Because the prediction accuracy of FNN soft-sensor model is related to the initial connection weights and thresholds, if the parameters are improper, it will lead to decline of prediction accuracy. Therefore, the hybrid PSO-GSA algorithm is used to optimize FNN soft-sensor model. The flowchart of PSO-GSA algorithm optimizing FNN is shown in Figure 5.

The algorithm procedure is described as follows in detail.

Step 1 (initialize parameters). Determine the topology of FNN. Initialize the weights w and the threshold value b of FNN and predispose the training samples of flotation process. Initialize the population size N and the number of iterations t.

Step 2 (generate the population randomly). Set the initial position $X = (x_1, x_2, \ldots, x_n)$ and initial velocity $v_i = (v_{i1}, v_{i2}, \ldots, v_{is})^T$, the learning factors c_1 and c_2, and the inertia weight w. Initialize the global best value $gbest$ and individual best value $pbest$, the decreasing coefficient α, and the gravitational constant G_0.

Step 3. The position of each individual corresponds to a set of weights and thresholds of FNN soft-sensor model. Train FNN and calculate the fitness value fit(t), of each individual. Find the optimal fitness value best(t) and the worst fitness value worst(t) and record the best position $gbest$.

Step 4. According to (13)–(15), calculate the mass of the individual, the gravitational constant G, and the Euclidean distance between two individuals. Then calculate the gravitation F and the acceleration a of the individual. At this time, according to (11)-(12), the global search ability of PSO algorithm is adopted to update the velocity and position of the individuals. Then calculate the corresponding fitness value fit(t) and the optimal value $gbest'$ of the individual. The optimal value $gbest'$ is compared with the optimal value $gbest$ in Step 3; the optimal individual after comparison is recorded as $gbest''$.

Step 5. Determine whether the termination condition is reached or not (the objective function reaches a certain value or the number of iterations reaches the maximum). If the termination condition is not met, the procedure returns to Step 3.

Step 6 (model validation). The corresponding parameters of the best individual $gbest$ are set as the weights and thresholds of FNN soft-sensor model and verify the established FNN soft-sensor model with the testing data.

4. Simulation Results

In this paper, 600 pieces of input data are selected as input and the concentrate grade and recovery rate are output of FNN soft-sensor model for the flotation process, where 550 samples are training data and the remaining 50 samples are testing data. Finally, the weights and thresholds of FNN are optimized by PSO-GSA algorithm. This paper selects the following five performance indexes to verify the prediction accuracy of different soft-sensor models, whose calculation equations are described as follows:

$$\begin{aligned}
\text{NRMSE} &= \sqrt{\frac{1}{T\left\|y_d\right\|^2}\sum_{t=1}^{T}\left(y(t) - y_d(t)\right)^2}, \\
\text{MSE} &= \frac{1}{T}\sum_{t=1}^{T}\left(y(t) - y_d(t)\right)^2, \\
\text{MAPE} &= \frac{100}{T}\sum_{t=1}^{T}\frac{\left|y(t) - y_d(t)\right|}{y_d(t)}, \\
\text{RMSE} &= \left[\frac{1}{T}\sum_{t=1}^{n}\left(y_d(t) - y(t)\right)^2\right]^{1/2}, \\
\text{SSE} &= \sum_{t=1}^{n}\left(y_d(t) - y(t)\right)^2,
\end{aligned} \tag{18}$$

where T is the number of the prediction samples, $y(t)$ is the predicted value, and $y_d(t)$ is the actual value.

The input dimension of FNN is 5, the number of neurons is 20 in the hidden layer, and the output dimension is 2.

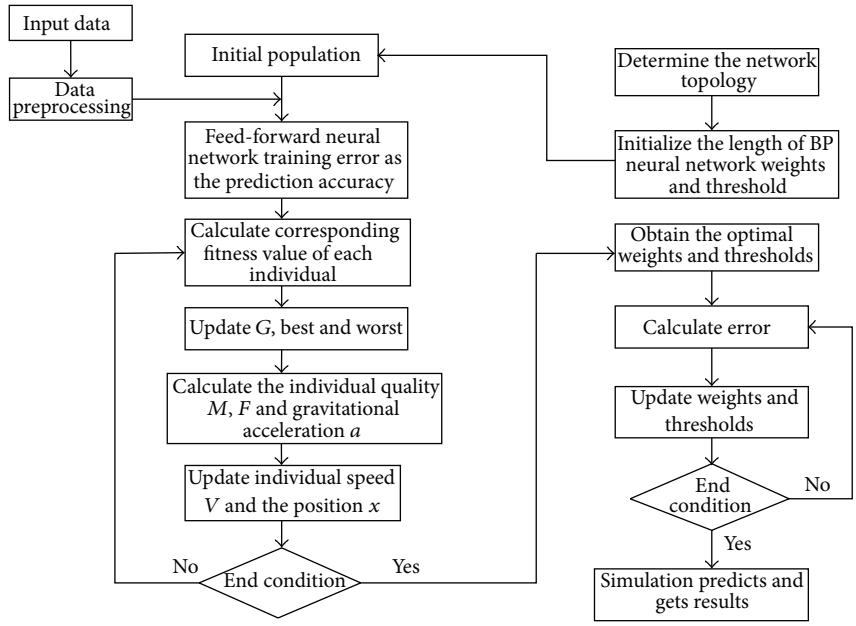
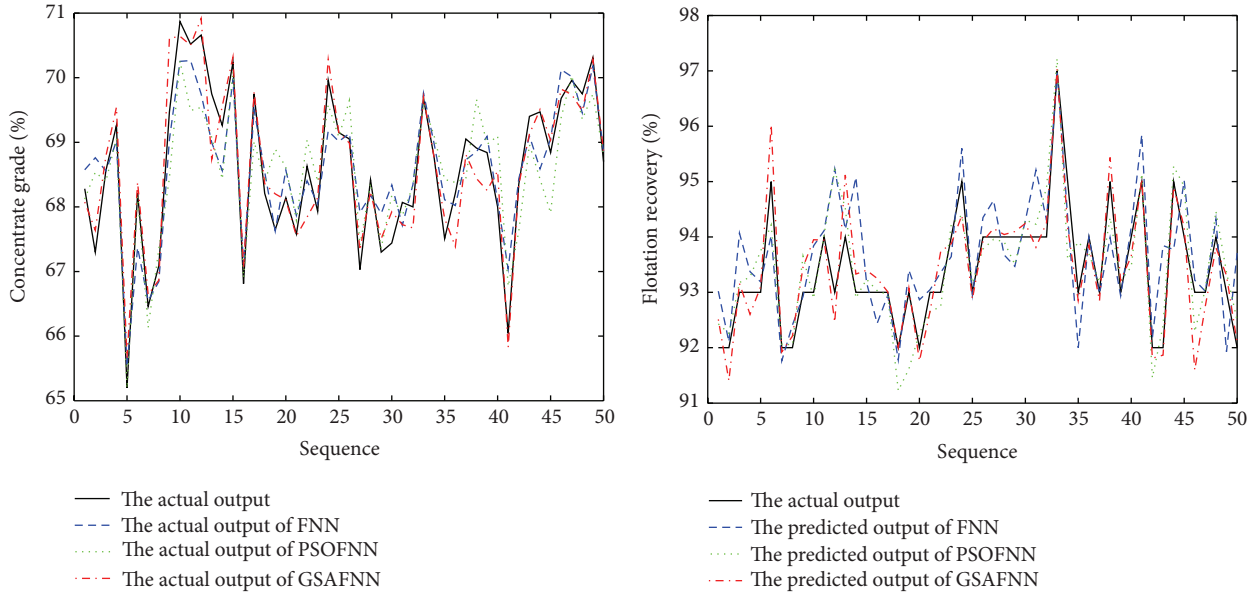


Figure 5: Flowchart of PSO-GSA optimizing FNN.

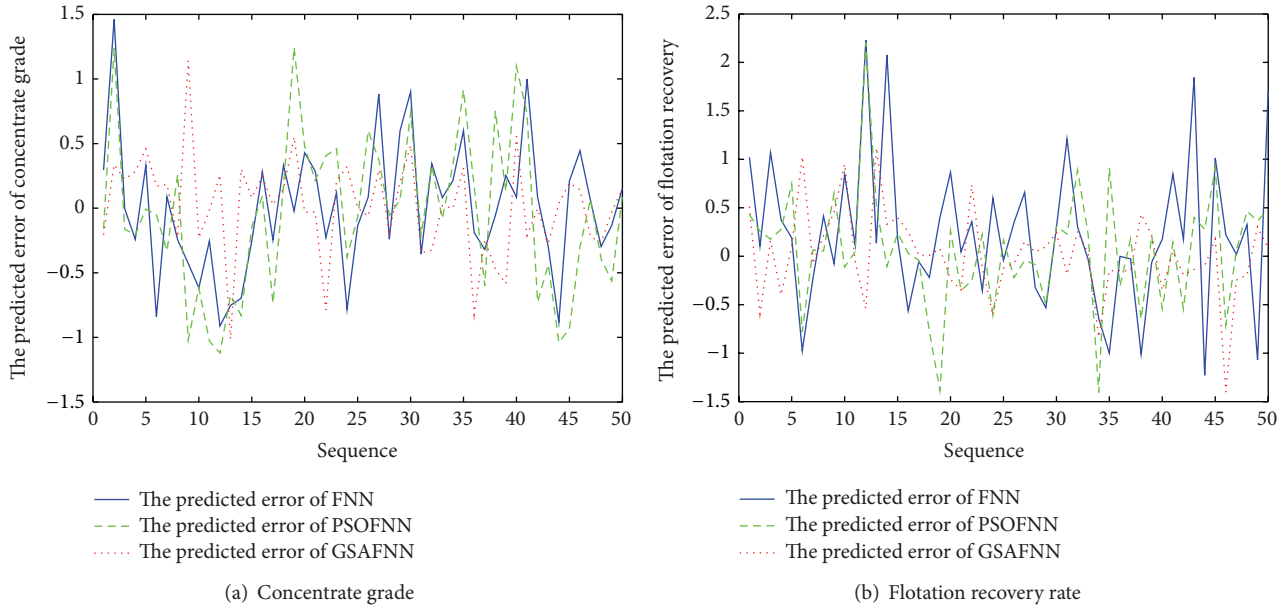


(a) Concentrate grade

(b) Flotation recovery rate

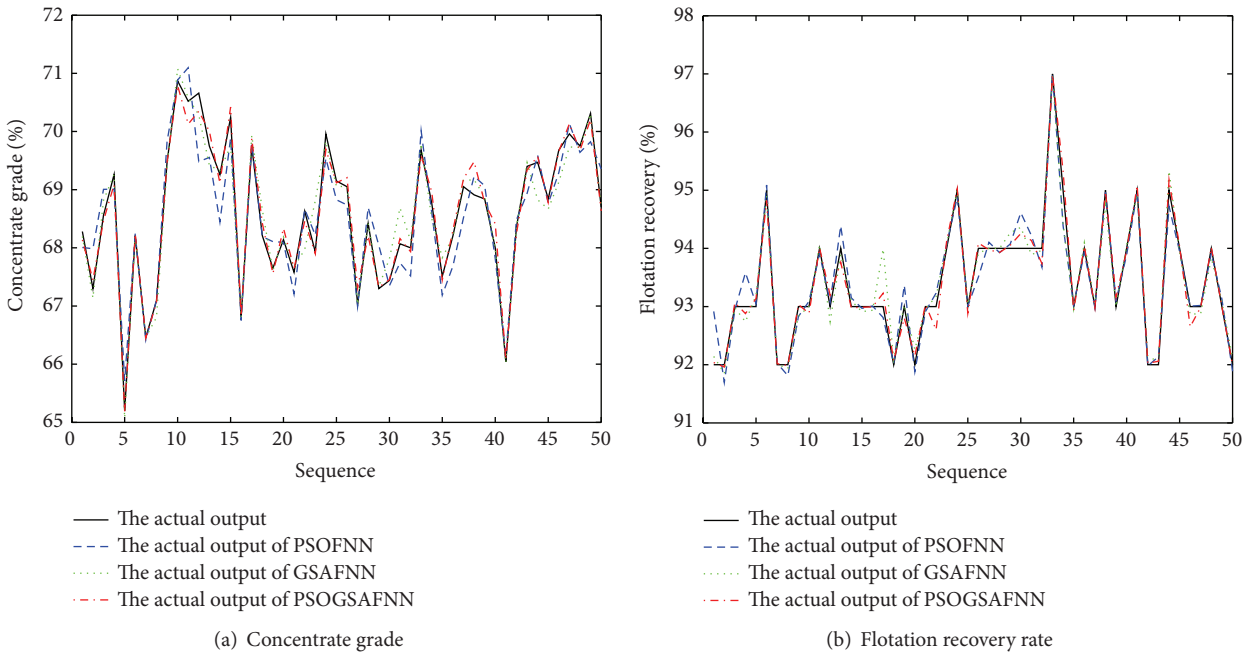
Figure 6: Predicted results of soft-sensor models.

The activation function of FNN is tanh and the output uses the linear activation function. The initialization parameters of PSO-GSA algorithm are described as follows: $N = 30$, $c_1 = c_2 = 2$, $G_0 = 1$, inertia weight $w = 2$, and the number of maximum iterations is $t = 300$. Firstly, three soft-sensor models based on FNN, FNN optimized by PSO algorithm, and FNN optimized by GSA are established to realize the prediction of concentrate grade and recovery rate in the flotation process. Figure 6 is a comparison of the predicted output and the actual output under three models. Figure 7 is a comparison of the output prediction error curves under three models. It can be seen, from the predicted output curves and the prediction error curves, in these three models, that FNN soft-sensor models optimized by PSO algorithm and GSA have better predictive accuracy than the standard FNN soft-sensor model. Therefore, in order to verify the validity of PSO-GSA hybrid algorithm, the proposed PSO-GSA FNN soft-sensor model is compared with PSO-FNN model and



(a) Concentrate grade (b) Flotation recovery rate

Figure 7: Predicted errors of soft-sensor models.



(a) Concentrate grade (b) Flotation recovery rate

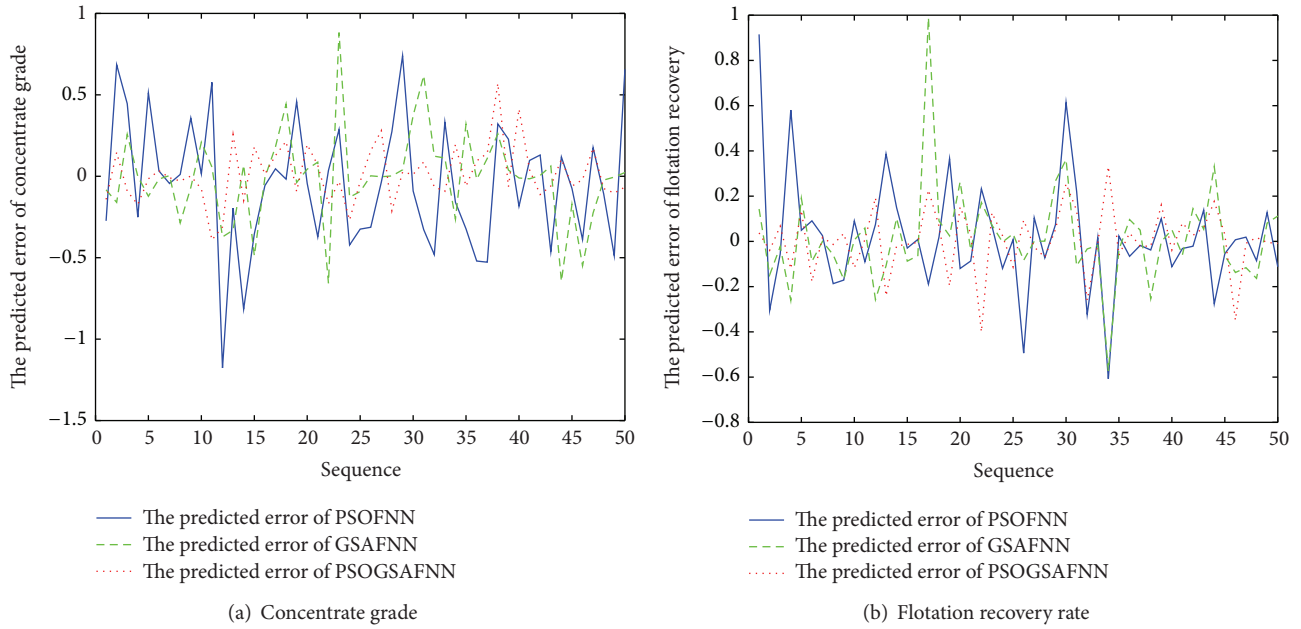
Figure 8: Predicted results of soft-sensor models.

GSA-FNN model. Figure 8 is a comparison of the predicted output and the actual output under three models. Figure 9 is a comparison of the output prediction error curves under three models.

In order to compare the predictive ability and precision of these soft-sensor models based on the above-defined performance index, the performances are calculated and the results are shown in Table 2. Seen from Section 4, the prediction error of FNN soft-sensor model based on PSO-GSA is minimum.

5. Conclusion

The five variables (feed grade, feed concentration, feed flow, agents flow, and feed granularity) are selected as the input variables of the discussed soft-sensor model. The flotation



(a) Concentrate grade

(b) Flotation recovery rate

Figure 9: The predicted error of soft-sensor model.

Table 2: Predictive performance comparison of soft-sensor models.

Predictive object	Predictive method	MSE	RMSE	NRMSE	SSE	MAPE
Concentrate grade (%)	FNN	0.2416	0.4915	0.0223	12.081	0.5564
	PSO-FNN	0.1528	0.3909	0.0008	7.6402	0.4471
	GSA-FNN	0.0764	0.2764	0.0006	3.8209	0.2649
	PSOGSA-FNN	0.0300	0.1733	0.0004	1.5009	0.1890
Flotation recovery rate (%)	FNN	0.1751	0.4184	0.0163	8.7535	0.3089
	PSO-FNN	0.0627	0.2505	0.0004	3.1372	0.1751
	GSA-FNN	0.0447	0.2114	0.0003	2.2341	0.1422
	PSOGSA-FNN	0.0194	0.1393	0.0002	0.9707	0.1054

concentrate grade and recovery rate are output variables. The hybrid algorithm combining PSO algorithm and GSA is used to optimize the parameters of FNN soft-sensor model in order to improve the predictive accuracy. It can be seen, from the prediction results and comparison results, that the FNN soft-sensor model based on the proposed PSO-GSA algorithm has the highest prediction accuracy compared with the other soft-sensor models, which can meet the online soft-sensor requirements of the real-time control in the flotation process.

Conflict of Interests

The authors declare no conflict of interests.

Authors' Contribution

Jie-Sheng Wang participated in the concept, design, and interpretation and commented on the paper. Shuang Han participated in the data collection, analysis, algorithm simulation, the draft writing, and critical revision of this paper.

Acknowledgments

This work is partially supported by the Program for China Postdoctoral Science Foundation (Grant no. 20110491510), the Program for Liaoning Excellent Talents in University (Grant no. LR2014008), the Project by Liaoning Provincial Natural Science Foundation of China (Grant no. 2014020177), and the Program for Research Special Foundation of University of Science and Technology of Liaoning (Grant no. 2011ZX10).

References

[1] D. Hodouin, S.-L. Jämsä-Jounela, M. T. Carvalho, and L. Bergh, "State of the art and challenges in mineral processing control," *Control Engineering Practice*, vol. 9, no. 9, pp. 995–1005, 2001.

[2] J. Kaartinen, J. Hätönen, H. Hyötyniemi, and J. Miettunen, "Machine-vision-based control of zinc flotation—a case study," *Control Engineering Practice*, vol. 14, no. 12, pp. 1455–1466, 2006.

[3] B. J. Shean and J. J. Cilliers, "A review of froth flotation control," *International Journal of Mineral Processing*, vol. 100, no. 3-4, pp. 57–71, 2011.

[4] J. M. Hargrave and S. T. Hall, "Diagnosis of concentrate grade and mass flowrate in tin flotation from colour and surface texture analysis," *Minerals Engineering*, vol. 10, no. 6, pp. 613–621, 1997.

[5] G. Bartolacci, P. Pelletier Jr., J. Tessier Jr., C. Duchesne, P.-A. Bossé, and J. Fournier, "Application of numerical image analysis to process diagnosis and physical parameter measurement in mineral processes. Part I. Flotation control based on froth textural characteristics," *Minerals Engineering*, vol. 19, no. 6–8, pp. 734–747, 2006.

[6] S. H. Morar, M. C. Harris, and D. J. Bradshaw, "The use of machine vision to predict flotation performance," *Minerals Engineering*, vol. 36–38, pp. 31–36, 2012.

[7] D. W. Moolman, C. Aldrich, J. S. J. Van Deventer, and D. J. Bradshaw, "The interpretation of flotation froth surfaces by using digital image analysis and neural networks," *Chemical Engineering Science*, vol. 50, no. 22, pp. 3501–3513, 1995.

[8] C.-H. Yang, J.-Y. Yang, X.-M. Mou, K.-J. Zhou, and W.-H. Gui, "A segmentation method based on clustering pre-segmentation and high-low scale distance reconstruction for colour froth image," *Journal of Electronics & Information Technology*, vol. 30, no. 6, pp. 1286–1290, 2008.

[9] J. Wang, Y. Zhang, and S. Sun, "Multiple T-S fuzzy neural networks soft sensing modeling of flotation process based on fuzzy C-means clustering algorithm," in *Advances in Neural Network Research and Applications*, vol. 67 of *Lecture Notes in Electrical Engineering*, pp. 137–144, Springer, Berlin, Germany, 2010.

[10] C.-H. Yang, H.-F. Ren, C.-H. Xu, and W.-H. Gui, "Soft sensor of key index for flotation process based on sparse multiple kernels least squares support vector machines," *The Chinese Journal of Nonferrous Metals*, vol. 21, no. 12, pp. 3149–3154, 2011.

[11] H. Li, T. Chai, and H. Yue, "Soft sensor of technical indices based on KPCA-ELM and application for flotation process," *CIESC Journal*, vol. 63, no. 9, pp. 2892–2898, 2012.

[12] K.-J. Zhou, C.-H. Yang, X.-M. Mou, and W.-H. Gui, "Intelligent prediction algorithm for floatation key parameters based on image features extraction," *Control and Decision*, vol. 24, no. 9, pp. 1300–1305, 2009.

[13] J.-S. Wang and Y. Zhang, "Application of the soft sensing model based on the adaptive network-based fuzzy inference system (ANFIS) to the flotation process," *Journal of Hefei University of Technology*, vol. 29, no. 11, pp. 1365–1369, 2006.

[14] Z.-X. Geng and T.-Y. Chai, "Soft sensor of technical indices based on LS-SVM for flotation process," *Journal of System Simulation*, vol. 20, no. 23, pp. 6321–6324, 2008.

[15] J. Wang, S. Han, N. Shen, and S. Li, "Features extraction of flotation froth images and BP neural network soft-sensor model of concentrate grade optimized by shuffled cuckoo searching algorithm," *The Scientific World Journal*, vol. 2014, Article ID 208094, 17 pages, 2014.

[16] J.-S. Wang, S. Han, and N.-N. Shen, "Improved GSO optimized ESN soft-sensor model of flotation process based on multi-source heterogeneous information fusion," *The Scientific World Journal*, vol. 2014, Article ID 262368, 12 pages, 2014.

[17] S.-Y. Ho, H.-S. Lin, W.-H. Liauh, and S.-J. Ho, "OPSO: orthogonal particle swarm optimization and its application to task assignment problems," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 2, pp. 288–298, 2008.

[18] R. Coban, "A fuzzy controller design for nuclear research reactors using the particle swarm optimization algorithm," *Nuclear Engineering and Design*, vol. 241, no. 5, pp. 1899–1908, 2011.

[19] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.