Cellular Artificial Bee Colony algorithm with Gaussian distribution

Zhang, M., Tian, N., Palade, V., Ji, Z. & Wang, Y.

Author post-print (accepted) deposited by Coventry University's Repository

Original citation & hyperlink:
Zhang, M, Tian, N, Palade, V, Ji, Z & Wang, Y 2018, 'Cellular Artificial Bee Colony algorithm with Gaussian distribution' Information Sciences, vol. 462, pp. 374-401. https://dx.doi.org/10.1016/j.ins.2018.06.032

DOI 10.1016/j.ins.2018.06.032
ISSN 0020-0255
ESSN 1872-6291

Publisher: Elsevier

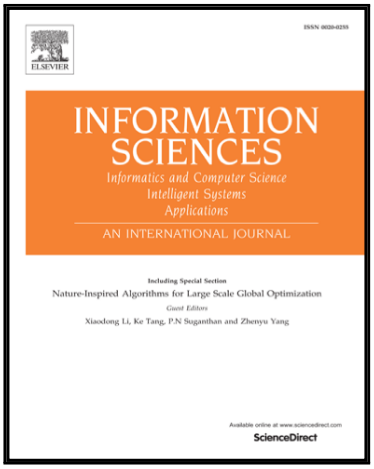
NOTICE: this is the author's version of a work that was accepted for publication in *Information Sciences.* Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in *Information Sciences* Vol 462, 2018. DOI: 10.1016/j.ins.2018.06.032

© 2017, Elsevier. Licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International http://creativecommons.org/licenses/by-nc-nd/4.0/

Copyright © and Moral Rights are retained by the author(s) and/ or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This item cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder(s). The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

This document is the author's post-print version, incorporating any revisions agreed during the peer-review process. Some differences between the published version and this version may remain and you are advised to consult the published version if you wish to cite from it.

Accepted Manuscript



Cellular Artificial Bee Colony Algorithm with Gaussian Distribution

Ming Zhang, Na Tian, Vasile Palade, Zhicheng Ji, Yan Wang

PII: S0020-0255(18)30472-9
DOI: 10.1016/j.ins.2018.06.032
Reference: INS 13725

To appear in: *Information Sciences*

Received date: 9 May 2017
Revised date: 9 June 2018
Accepted date: 12 June 2018

Please cite this article as: Ming Zhang, Na Tian, Vasile Palade, Zhicheng Ji, Yan Wang, Cellular Artificial Bee Colony Algorithm with Gaussian Distribution, *Information Sciences* (2018), doi: 10.1016/j.ins.2018.06.032

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

ACCEPTED MANUSCRIPT

Cellular Artificial Bee Colony Algorithm with Gaussian Distribution

Ming Zhang[a], Na Tian[b], Vasile Palade[c,*], Zhicheng Ji[a], Yan Wang[a,**]

[a]*Engineering Research Center of Internet of Things Technology Applications Ministry of Education, Jiangnan University, Wuxi City, 214122, PR China*
[b]*School of Humanities, Jiangnan University, Wuxi City, 214122, China*
[c]*School of Computing, Electronics and Mathematics, Coventry University, Coventry, CV1 5FB, UK*

Abstract

The Artificial bee colony (ABC) algorithm has shown competitive performance for handling various optimization problems. However, despite its strong global search ability, it suffers from a poor convergence rate and it loses the balance between exploitation and exploration. To compensate for this weakness, our paper proposes a cellular structured neighborhood, with Gaussian-based search equation and local attractor, and a redefined probability calculation for the ABC algorithm after an empirical analysis. The proposed algorithm is named as CGABC-Cellular neighborhood with Gaussian distribution ABC. The cellular automata (CA) model can keep individuals interact with specific neighbors while maintaining the population diversity. The Gaussian-based search equation combined with the local attractor can help exploit locally the search space, and the modified probability calculation based on rank sorting can make the selection of onlooker bees more robust and appropriate. Theoretical analysis are made to prove the global convergence of the CGABC algorithm based on the theory of probability metric spaces, and the results show that CGABC will converge to the global optimum. The proposed algorithm is tested on a set of benchmark functions and three real-world problems (the "Lennard Jones potential problem", the "frequency-modulated sound wave synthesis problem" and the "feature selection problem"), and the results demonstrate that our proposed strategies help ABC achieve higher accuracy and faster convergence when compared with other ABC variants and swarm-based evolutionary algorithms (EAs).

Keywords: Artificial bee colony, Cellular automata, Gaussian distribution, Probability calculation

1. Introduction

Evolutionary Algorithms (EAs) [13], as an important branch of derivative-free techniques, have been demonstrated to be efficient tools for solving difficult optimization problems characterized as multi-modal, non-differentiable, non-convex, nonlinear as well as other challenging problems. As a class of stochastic optimization and adaptive techniques, EAs draw inspirations from natural evolution and the collective behavior of social insect colonies or animal groups in nature. They provide a framework that mainly includes Genetic Algorithms (GA) [6], Differential Evolution (DE) [43] algorithm, Particle Swarm Optimization (PSO) [27], etc.

Artificial bee colony (ABC) algorithm was initially proposed by Karaboga [23] in 2005, inspired from the intelligent foraging behavior of honeybees. In this algorithm, there are three kinds of bees to perform different tasks. Employed bees take the responsibility for searching food sources in a given multidimensional continuous search space and propagating food information to onlooker bees. After receiving the information,

*corresponding author
**corresponding author
Email addresses: zmjiangnan@126.com (Ming Zhang), tianna@jiangnan.edu.cn (Na Tian), vasile.palade@coventry.ac.uk (Vasile Palade), zcji@jiangnan.edu.cn (Zhicheng Ji), wangyan@jiangnan.edu.cn (Yan Wang)

ACCEPTED MANUSCRIPT

onlooker bees would perform an exploitative search around the neighborhood of better food sources selected by fitness values. The scouts are designed to help jump out of local minima. Because of its easy implementation with fewer control parameters as well as simple structure, ABC has been compared favourably to other EAs [24], and thus has been applied to many real-world problems [38].

However, mainly due to the search equation which performs well in exploration but badly in exploitation, the ABC algorithm inevitably faces poor convergence. To address this drawback, researchers have developed plenty of approaches in many aspects, as shown in Table 1. Generally, the variants of ABC can be classified into three categories: ABC with modified search equations, hybrids of ABC with other techniques, and ABC with adaptive mechanisms. In terms of the modified search equations, the global best solution inspired from PSO and DE was highlighted to improve the exploitation ability such as in the GABC [50], MABC [17], ABC/best/1 [18], and EABC [20] algorithms. Under the guidance of global best solution, individuals could be pulled towards the potential regions, thus the convergence rate of the algorithm can be improved to some extent. Besides, both [12] and [16] introduced the property of Gaussian distribution system to the search equation. In [12], Gaussian distribution was used for parameter tuning to get better stability and exploitative behavior, while it was used to control the movement distribution of candidate solutions in [16]. Instead of differential position update rule in the original ABC, a distribution-based solution update rule was proposed in the distABC [4] algorithm to prevent stagnation in the whole population by using the mean and standard deviation of two selected food sources in order to obtain a new candidate solution. In the case of hybrid ABC, some particular search methods or evolution operators involved in other EAs have been incorporated in the ABC. For example, chaotic-based search widely used for initialization was designed to help enhance the global convergence and population diversity [30, 40]. Other effective techniques, such as orthogonal experimental design (OED) [19], Rosenbrocks rotational direction method [22] and Hooke-Jeeves pattern search [21], were taken to generate a more efficient candidate solution by sampling a set of well representative combinations of solutions. Finally, for the adaptation mechanism, the new selection strategy of neighborhood was designed for the updating equation to obtain a more precise search ability [25, 40]. To avoid random search direction, previous successful experience of foraging was memorized for bees to provide a more favorable search guidance [1, 28, 32]. Kiran et al. [29] integrated five solution update rules and counters into the ABC (ABCVSS) algorithm to update solutions, aiming to efficiently handle optimization problems with different characteristics. However, this study is not limited to the above mentioned methods; for a comprehensive review, classifying the methods of improvements and the applications, one can refer to [38].

However, to our best knowledge, the practical effect of onlooker bees and scouts on the performance of the ABC algorithm is rarely studied; and little attention has been paid on the neighborhood structure of the ABC algorithm for achieving superior performance. Based on the above consideration, we first analyze the actual role of the onlooker bees played in improving the convergence rate and the effect of scouts on escaping from local optima, in order to obtain a comprehensive understanding of the search mechanism of the ABC algorithm. Different from the randomly generated population and applying the search mechanism on the population as a whole, we use a cellular topology, motivated by cellular automata (CA) [46], to decentralize the population. The decentralization of the ABC can arrange the whole population in a two-dimensional lattice structure and keep individuals interacting within a particular neighborhood in order to strike a balance between exploitation and exploration. We further introduce the Gaussian distribution combined with redefined local attractor in the search equation to guarantee the convergence. Moreover, a modified probability calculation is proposed to benefit the selection of onlooker bees, thereby improving the exploitation ability. This proposed algorithm is named as CGABC (Cellular neighborhood with Gaussian distribution ABC). A theoretical analysis for the CGABC algorithm is given in a probabilistic metric space, and experimental studies are conducted based on a set of numerical benchmarks and three real-world applications to validate the performance of the proposed CGABC algorithm.

The rest of paper is structured as follows. Section 2 describes the ABC algorithm in detail. Section 3 introduces the motivation of this paper and presents the framework of the proposed approach. Experimental studies are reported and discussed in section 4. Finally, some conclusions are drawn in section 5.

ACCEPTED MANUSCRIPT

Table 1. Three types of ABC variants

Type	Algorithms	Abbreviation	Reference
modified search equations	Gbest-guided ABC	GABC	[50]
	Improved ABC	MABC	[17]
	Global Best ABC	ABC/best/1	[18]
	Enhancing ABC	EABC	[20]
	Gaussian ABC	GABC	[12]
	Bare Bones ABC	BABC	[16]
	ABC with Distribution-based Update Rule	distABC	[4]
hybrids of ABC with other techniques	Chaotic ABC based on Tent Map	STOC-ABC	[30]
	ABC with Novel Search Equation and Improved Dimension Selection Strategy	NSABC	[40]
	ABC with Modified Search Equation and Orthogonal Learning	OCABC	[19]
	Rosenbrock ABC	RABC	[22]
	HookeJeeves ABC	HABC	[21]
adaptive mechanisms	Quick ABC	qABC	[25]
	ABC with Modification Rate and Scaling Factor	ASF-MR	[1]
	Directed ABC	dABC	[28]
	ABC with Memory	ABCM	[32]
	ABC with Variable Search Strategy	ABCVSS	[29]

2. Overview of ABC Algorithm

The ABC algorithm contains three groups of bees: employed bees, onlooker bees and scouts. The number of employed and onlooker bees are equal, both of which account for half of the colony. The position of a food source represents a candidate solution of the optimization problem, and its nectar amount denotes the corresponding fitness value. It should be noted that one food source is assigned to only one employed bee. Assuming that the initial population, consisting of SN solutions with D-dimensional vector $X_i = (x_{i,1}, x_{i,2}, ..., x_{i,D})$ is randomly generated by Eq. (1),

$$x_{i,j} = x_j^{min} + rand(0,1)(x_j^{max} - x_j^{min}) \tag{1}$$

where $i \in \{1, 2, ..., SN\}$, $j \in \{1, 2, ..., D\}$. x_j^{min} and x_j^{max} respectively represent the lower and upper bounds of j_{th} dimension. rand(0,1) denotes a random value uniformly distributed in (0,1). In the employed bee phase, a dimension is randomly chosen to generate a new candidate solution V_i by the following equation:

$$v_{i,j} = x_{i,j} + \varnothing_{i,j}(x_{i,j} - x_{k,j}) \tag{2}$$

where $j \in \{1, 2, ..., D\}$ and $k \in \{1, 2, ..., SN\}$ are randomly selected indexes; $\varnothing_{i,j}$ is a uniformly random value in [-1, 1]. Then, a greedy selection based on the quality of the nectar amount is adopted to select the better one between the candidate solution and the old solution. After that, the employed bees share information of food sources with onlooker bees through dancing.

The update process of onlookers is the same as that in the employed bee phase, however, the main difference between them relies on that the onlookers select potential food sources to exploit according to the probabilities calculated by fitness values. Assuming a minimization problem, the probability p_i and the

fitness value fit_i of solution X_i is calculated as follows:

$$p_i = fit_i / \sum_{i=1}^{SN} fit_i \tag{3}$$

$$fit_i = \begin{cases} \frac{1}{1+f_i}, & if \quad f_i \geq 0 \\ f_i, & if \quad f_i < 0 \end{cases} \tag{4}$$

where f_i means the objective function value. During the scout phase, a solution would be replaced with a new generated solution Eq. (2) if it had not been improved after consecutive *Limit* iterations.

3. ABC with the Concept of Cellular Automata Model

3.1. Motivations

3.1.1. The neighborhood effect

Two contradictory aspects that influence the performance of EAs are the exploration and the exploitation abilities. Exploration is considered as the capability to search the undiscovered region in the whole search space to find potential solutions, while exploitation indicates the ability to exploit potential solutions using the previous information obtained by good individuals. Based on Eq. (2), a candidate solution is generated by moving the individual towards a randomly chosen one in the population with a random coefficient in $[-1, 1]$; therefore, the search direction is totally stochastic, which can help preserve the population diversity to some extent. However, it is difficult to ensure a high probability of successful update because the ABC algorithm ignores the beneficial information hidden in the neighborhood and does not make full use of the elite solutions to guide the search towards a potential direction.

Assuming that there are ten individuals as shown in Fig. 1, the five-pointed star highlighted in yellow is the current individual denoted as X_i, while other nine individuals are shown in blue squares and labeled as $S1$ to $S9$. The red circle is the actual optimal solution represented as X_{opt}. $L1$ and $L2$ are the search directions from the current individual to $S1$ and $S5$, respectively. According to Eq. (2), any solution in the subset $\{S1, S2, ..., S3\}$ would be randomly chosen to help generate the candidate solution. This would cause a high-level randomness, and it may even exert a negative effect on the global convergence when some distant solutions (e.g., $S1$) are selected. On the other hand, the search direction can be purposeful if a good solution in the neighborhood is considered. In Fig. 1, the neighborhood of X_i is circled by dotted line, where $S4$ and $S5$ are neighbors. X_i will move towards X_{opt} if $S5$ is considered to guide the search, and the convergence speed will be increasingly improved. Therefore, it is necessary to establish the neighborhoods for the population.

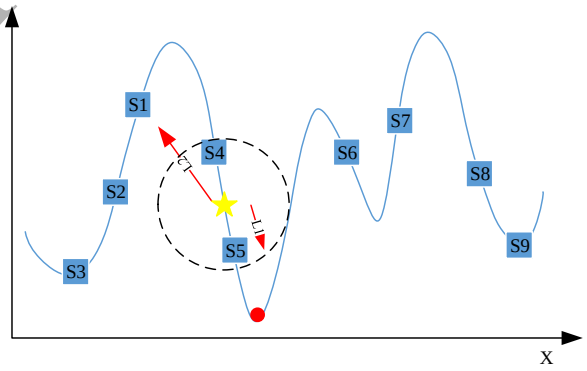


Fig. 1. The search direction of an individual

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

3.1.2. Preliminary experiments on properties of the ABC algorithm

Besides, onlookers selected by probabilities (as shown in Eq. (3)) take the responsibility for exploiting better individuals. Nevertheless, the fitness values can be approximately equal to 1 when the corresponding objective function values are positive but too small, e.g. 1e-30. This has a directly negative effect on the calculation of probabilities, on which onlookers depend to select potential individuals to exploit. Furthermore, scouts are supposed to increase diversity when the population is getting stuck in a local optimum. However, the population can drastically lose diversity when a new solution provided by a scout is surrounded and influenced by a large amount of individuals falling into stagnation. In view of the fact that the number of scouts is small and the time occurred is always late, therefore the effect of scouts is relatively feeble.

In order to validate the above discussion about the performance of onlooker bees and scouts on the convergence, an experiment is conducted to record the mean number of scouts, the mean percentage of independent individuals regarded as onlookers and the mean convergence curves along with the search process. Figs. 1-3 present the experimental results on the Ackley function with $D = 30$ in $[-32, 32]$ observed at three different values of Limit, i.e. $0.2 * SN * D$, $0.6 * SN * D$ and $1.0 * SN * D$ [50], respectively. From Figs. 2-4 , inspiring conclusions can be drawn as follows:

1) Scouts always appear in a short time just after the stagnation of the population, and its appearance may postpone due to the increasing Limit;
2) The percentage of onlooker bees oscillates temporarily along with the appearance of scouts, otherwise it maintains a high proportion (more than 60%);
3) The convergence curves keep unchangeable no matter what the values of Limit are.

With the purpose of overcoming the mentioned shortcomings, three novel operators are introduced in this paper, namely: the cellular structured neighborhood mechanism, the Gaussian-based search equation and the modified probability calculation, in order to efficiently exploit the available information concealed in the neighborhood and to find the elite individuals. Therefore a simple and effective framework containing these three operators, named as the CGABC algorithm, is proposed.

3.2. Hybridization of CA model and ABC algorithm

Based on the analysis in the above subsection, we need a well-designed communication structure and rigid information inheriting mechanism to influence the collaboration of individuals and thereby affect the performance of the ABC algorithm.

The concept of the CA model was first proposed by Von Neumann and Ulam [45], and the basic classifications of CA was described by Wolfram [46]. As a discrete dynamical system, CA can stimulate micro-behavior by micro-dynamics using the interaction of individuals (cells) connected in particular neighborhood structures. The deterministic automata (cells) in CA are homogeneously interconnected and can work synchronously at discrete steps following the same transition rule; thus, a large amount of elements working

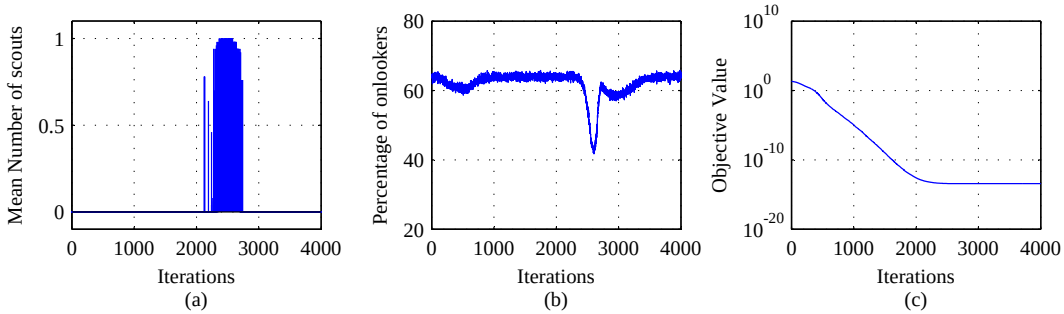


Fig. 2. Results on the Ackley function with $Limit = 0.2 * SN * D$: (a) the mean number of scouts; (b) the percentage of selected onlookers. (c) convergence curve on the Ackley function.

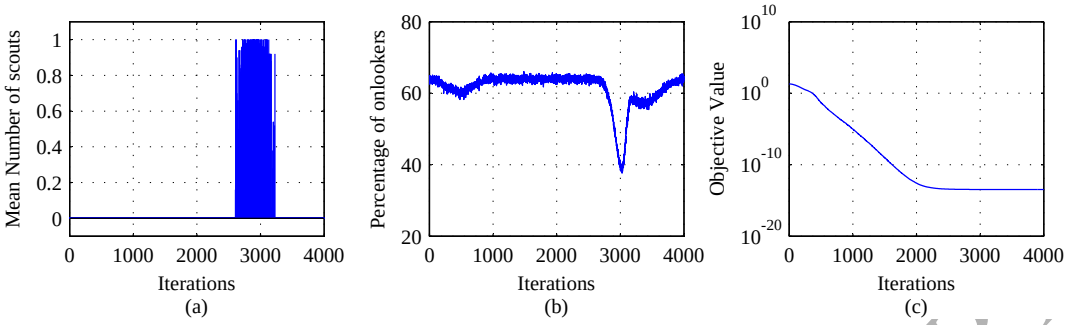


Fig. 3. Results on the Ackley function with $Limit = 0.6 * SN * D$: (a) the mean number of scouts; (b) the percentage of selected onlookers. (c) convergence curve on the Ackley function.

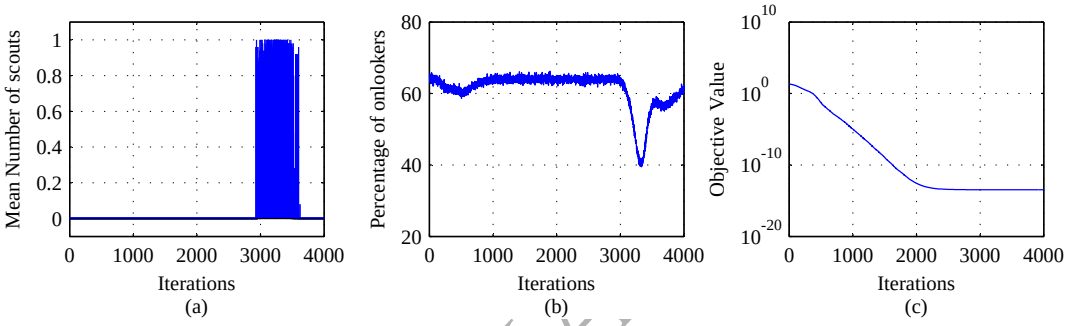


Fig. 4. Results on the Ackley function with $Limit = 1.0 * SN * D$: (a) the mean number of scouts; (b) the percentage of selected onlookers. (c) convergence curve on the Ackley function.

in parallel can be found to be quite powerful to model complex systems [31]. Therefore, the CA model is adopted to decentralize the population of the ABC.

3.2.1. Cellular automata (CA)

There are four basic components of CA: cell space, neighborhood, cell state, and transition rule. Cell space presents the connecting structure of cells, and a checkboard-like lattice structure in two dimensions is employed in this paper. A periodic boundary is used for cells on the edges to decide their neighbors by means of imaging the grids embedded in toroidal topology, which can be regarded as taping the left and right boundaries of the rectangle to form a tube, and then taping the upper and bottom boundaries of the tube to form a torus. Neighborhood is defined as the cells surrounding a given cell, and the six commonly used neighborhoods are shown in Fig. 5 where L5 and L9 are linear neighborhood and C9, C13, C21, and C25 are compact neighborhood. Cell state is considered as the intrinsic feature of a cell, denoting the different states in which a single cell can be. The transition rule is used to control the following state of a cell according to its current state and the states of its neighboring cells.

Given a cell, apart from the number of neighbors, the distribution of neighbors is another important characteristic that crucially influence the property of the neighboring structure. For instance, although the number of neighbors for neighborhood L9 and C9 are the same, the dispersion of them are totally different. Alba et al. [2] defined the radius for both the 2-D grid and the neighborhood by the dispersion of n^* points in a circle centering $(\bar{x}, \bar{y})$ based on Eq. (5), and then the grid-neighborhood relationship is able to be precisely quantified by the relative ratio between their radius according to Eq. (6). For comparison, the radius and ratio for different neighborhood on two grid shapes (i.e. $100 = 5 * 20$ and $100 = 10 * 10$) with 100 individuals are presented in Table 2. It is obviously that with the same number of neighbors, a thinner grid shape gets small ratio because of the large dispersion (e.g. C9 and L9 in Table 2).

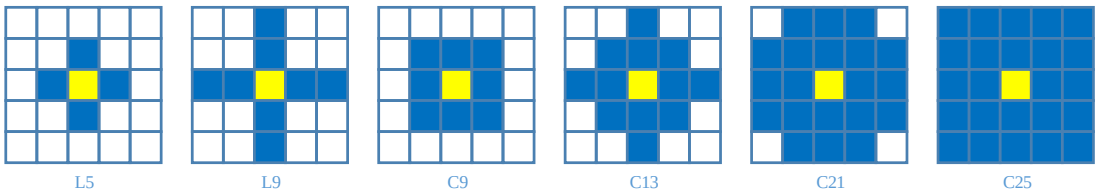


Fig. 5. Six common types of neighborhoods in CA

$$rad = \sqrt{(\sum_{i=1}^{n^*}(x_i - \bar{x})^2 + \sum_{i=1}^{n^*}(y_i - \bar{y})^2)/n^*}, \bar{x} = \sum_{i=1}^{n^*}\frac{x_i}{n^*}, \bar{y} = \sum_{i=1}^{n^*}\frac{y_i}{n^*} \tag{5}$$

$$ratio = \frac{rad_{neighborhood}}{rad_{shape}} \tag{6}$$

3.2.2. ABC algorithm with cellular structured population

From the above description of CA, it can be seen that CA are easy to implement without any strict mathematical reasoning. To enhance the cooperation of population, the concept of a CA model is utilized in the ABC algorithm to explore the neighboring structure and the diffusing mechanism of the swarm system. Some definitions for integrating the ABC and CA model are given as follows.

1) Cell space

At first, an individual is thought of as a cell, and the number of all cells in the CA is exactly the same as the number of individuals in the ABC algorithm. Then, each individual is randomly allocated to a unique cell without duplication in the initial phase, and most importantly the position of an individual in the lattice structure is fixed during the whole search process.

2) Neighborhood

Taken neighborhood C9 as an example, an explicit representation of the neighboring structure is depicted in Fig. 6 with one hundred individuals. The neighbors of yellow individuals are highlighted in blue, while the overlapping neighbors belonging to different cells are colored in purple.For example, the neighbors of individuals 5, 12 and 19 denoted as X_5 ,X_{12} and X_{19} are $\{X_1, X_2, X_3, X_4, X_6, X_9, X_{10}, X_{11}\}$, $\{X_6, X_7, X_8, X_{11}, X_{13}, X_{16}, X_{17}, X_{18}\}$,$\{X_{13}, X_{14}, X_{15}, X_{18}, X_{20}, X_{21}, X_{22}, X_{23}\}$, respectively. The overlapping neighbors between individuals X_5 and X_{12} are $\{X_6, X_{11}\}$, and those between X_{12} and X_{19} are $\{X_{13}, X_{18}\}$. Apparently, each individual in the grid has the same neighborhood overlapping with nearby individuals. Individuals locally interact with their neighbors, and such limited information transmission has advantages in improving the local search ability. Meanwhile, due to the decentralized population, the information delivered by individuals could be slowly diffused to others through overlapping neighbors. In

Table 2. Radius and ratio of six representative neighborhoods in Fig. 4

Neighborhood	$5 * 20$			$10 * 10$		
	rad_{shape}	$rad_{neighborhood}$	*Ratio*	rad_{shape}	$rad_{neighborhood}$	*Ratio*
L5	5.2783	0.8944	**0.1694**	4.0620	0.8944	**0.2202**
L9	5.2783	1.4907	**0.2824**	4.0620	1.4907	**0.3670**
C9	5.2783	1.1547	**0.2188**	4.0620	1.1547	**0.2843**
C13	5.2783	1.4676	**0.2780**	4.0620	2.0000	**0.3613**
C21	5.2783	1.7995	**0.3409**	4.0620	1.4676	**0.4430**
C25	5.2783	2.0000	**0.3789**	4.0620	1.7995	**0.4924**

ACCEPTED MANUSCRIPT

Fig. 6, It can be noted that $\{X_6, X_{11}\}$ takes responsibility for transferring information between X_5 and X_{12}, and $\{X_{13}, X_{18}\}$ is useful for delivering information between X_{12} and X_{19}.Thus, the information between X_5 and X_{19} can be slowly exchanged through $\{X_6, X_{11}, X_{13}, X_{18}\}$.

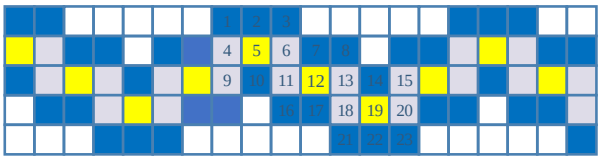


Fig. 6. Neighboring structure in the neighborhood C9

3) Cell state

According to the definition of the ABC algorithm, the cell state can be briefly defined as the employed bee state, onlooker bee state or scout bee state. Naturally, how to switch cell states is an important issue for the search process to balance the exploitation and exploration. It can be noted that the state transition between the employed bees and onlooker bees is dominated by the parameter p based on Eq. 3, and the state transition between the employed bees and scouts is determined by the parameter $Limit$. However, due to the above consideration, these two parameters are not appropriately designed for the algorithm. As we all know, the search ability of EAs could be highly enhanced by efficient parameter adaptation, especially when the control parameter is reasonably designed to accommodate with different phases of the search process. In the ABC algorithm, good individuals do not have more opportunities than other individuals to be chosen by onlookers to perform meticulous exploitation due to the improper calculation of the parameter p; therefore, the valuable information involved in good individuals cannot be fully exploited to guide the search into more potential regions.

In order to take full advantages of the beneficial information hidden in good individuals, a novel probability assignment is defined based on the ranking of individuals at the current iteration, presented as follows.

$$Pr_i = exp(-2.0 * \frac{Fr_i}{SN/2}) \tag{7}$$

where Fr means the rank of individuals from best to worst; Fr_i and Pr_i denote the rank and the probability of X_i. Here, -2.0 is used as the exponential growth coefficient to control the distribution of probabilities. Negative exponential function enlarges the probabilities of those individuals that are better than the median individual, whereas it nearly ignores the lagged individuals. As shown in Fig. 7, the

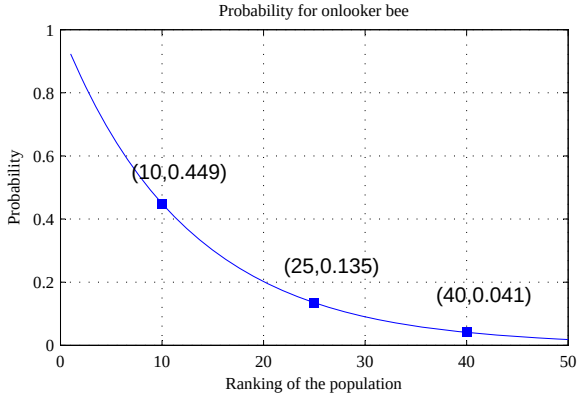


Fig. 7. The probability curve for onlooker bees with $SN = 50$

ACCEPTED MANUSCRIPT

probabilities of the individuals that are worse than the median individual are all smaller than 0.135, while the probabilities of those ranking in the top ten are over 0.449. Therefore, the transition between the employed bee state and the onlooker bee state can be strictly controlled. In other words, only a few individuals can be frequently employed as onlooker bees performing exploitative search. From the viewpoint of scouts, another advantage lies in that the transition probability between the employed state and the scout state may be increased to some extent. Given a top individual with a high probability, it may easily get trapped into local minima with little useful information after the intensive exploitation used as an onlooker bee; and then it will be replaced by a scout as soon as possible. As a consequence, the population can quickly extract beneficial information from better individuals for exploitation as well as timely replace trapped individuals with randomly generated solutions. This can highly improve the convergence rate of the algorithm while preserving the population diversity.

In order to make a clear and accurate comparison between the old and the candidate solution, the objective function values are directly applied to the selection of better solutions as in [20]. Hence, there is no need to calculate the fitness values for each solution, dramatically reducing the computational cost.

4) Transition rule

The search equation updating the positions of individuals is regarded as being the transition rule. With the aim to improve the local exploitation in the neighborhood, a novel search equation is designed, which gets inspiration from Gaussian distribution. The generalized formulation is defined as follows.

$$v_{i,j} = N(s_{i,j}, \alpha * |s_{i,j} - x_{k,j}|) \tag{8}$$

$$\alpha = 0.8 * (1 - \sqrt{\frac{FEs}{maxFEs}}) + 0.2 \tag{9}$$

where $j \in 1, 2, ..., D$ is a randomly chosen dimension; FEs denotes the index of current function evaluation; $maxFEs$ means the maximum number of function evaluations; $k \in 1, 2, ..., D, k \neq i$ and X_k is a randomly chosen solution from the whole population instead of the neighborhood; S_i is regarded as the local attractor of individual X_i , and $s_{i,j}$ is the j_{th} element of vector S_i ; α is a scaling factor to control the magnitude of variance operator. Based on previous experiments, α decreasing iteratively from 1.0 to 0.2 by Eq. (9) can perfrom best, which can improve the search stability by decreasing the search scope along with the evolutionary process. Besides, the descending preference can be explicitly considered to make the search process undergo two optimization phases: exploration at the initial phase and exploitation at the later phase.

It should be emphasized that S_i is an abstract term having different definitions for employed and onlooker bees as shown in Eqs. (10) and (11), where $x_{lbest,j}$ and $x_{gbest,j}$ are the j_{th} element of the local best position X_{lbest} and the global best position X_{gbest} , respectively. λ , which controls the degree $x_{i,j}$ depending on $x_{lbest,j}$, decreases along with the iterations; $N(\cdot)$ is used as a noise for λ , obeying a normal distribution whose mean and standard deviation both are 0.5.

$$s_{i,j} = \lambda * x_{lbest,j} + (1 - \lambda) * x_{k,j} \tag{10}$$

$$s_{i,j} = \lambda * x_{lbest,j} + (1 - \lambda) * x_{gbest,j} \tag{11}$$

$$\lambda = N(0.5, 0.5) * (1 - (\frac{FEs}{maxFEs})^2) \tag{12}$$

The main difference between Eq. (10) and (11) lies in that the former equation uses a randomly selected individual X_k while the latter one uses the global best solution X_{gbest}. The effect of X_k can be viewed as a random motion, making the individual fluctuate so that it may reach a point far from the original position, especially when the local best individual is stuck in a local optimum. Therefore, the movement of an individual is endowed with global search ability by using X_k , which can help the local best individual skip out onto a better position. Additionally, the integration of global best solution X_{gbest} can apparently accelerate the convergence rate of the search. As a whole, $s_{i,j}$ works as the local attractor inside a hyper-rectangle with $x_{lbest,j}$ and $x_{k,j}$ (or $x_{gbest,j}$) regarded as the two ends of the diagonal. As mentioned above,

ACCEPTED MANUSCRIPT

each individual concentrate on exploiting in the specialized neighborhood, so the information diffusion of better individuals is slowed down. This may exert a negative influence on the convergence rate although the population diversity is preserved. It is apparent that the search centered around S_i with Gaussian distribution can accelerate the diffusion of the local best individual through overlapping neighbors and hence drastically improve the exploitative efficiency.

In general, the neighborhood of each individual is determined by the CA model, and a Gaussian-based search equation with well-designed parameter adaptation is proposed in order to have a better balance between global and local search. The pseudocode of CGABC is described in Algorithm 1.

3.2.3. Time complexity of CGABC

The cellular structure does not increase the complexity of the CGABC, since the neighborhood of each individual is predesigned, as depicted in Fig. 5. Compared to the ABC algorithm, the selection of local best individual, the Gaussian based search equation and the population sorting are additional operations contained in the CGABC, of which the time complexity are $O(SN * 2N_{neigh})$ (N_{neigh} is the number of neighbors of each individual), $O(SN)$ and $O(SN * log(SN))$, respectively. The time complexity of the ABC algorithm is $O(SN * D)$, which has a larger scale than the three additional operations. Therefore, the time complexity of the CGABC remains $O(SN * D)$. Hence, it can be concluded that the complexity of the CGABC algorithm is not increased by the additional operations.

4. Convergence analysis of CGABC

The stochastic properties of nature-inspired algorithms make them difficult to validate global convergence in theory. We follow the global search criteria provided by Solis and Wets [35, 41] to investigate the convergence of the proposed algorithm. For clarity, some relevant preliminaries are included below.

4.1. Preliminaries

Remark 1. A random search algorithm can be regarded as global search algorithm if the successive approximations generated by the algorithm with a random initialization are guaranteed to converge to the global minimizer.

Assumption 1. *Given a continuous fitness function f in the feasible domain S, $f : R^N \to R$ and $S \in R^N$. The objective is to find a point $z \in S$ which can minimize f on S or at least generate an acceptable approximation of the infimum of f on S .*

Let $\{z_k\}_{k=0}^{\infty}$ be a sequence obtained by the random search algorithm. Global convergence requires that the sequence $\{f(z_k)\}_{k=0}^{\infty}$ should converge to the infimum of f on S. For the sake of avoiding some pathological situations which prevents the algorithm finding the global minimizer, the search objective of the algorithm is defined as the essential infimum of f on S as follows:

$$inf(m : v[z \in S \mid f(z < m) > 0]) \tag{13}$$

where $v[A]$ is the Lebesgue measure on the set A.

Remark 2. The ε-acceptable region $R_\varepsilon \in S$ is defined as the following:

$$R_\epsilon = \{z \in S \mid f(z) < \psi + \varepsilon\} \tag{14}$$

where $\varepsilon > 0$, and ε denotes the acceptable error; ψ means global minimum. $R_\epsilon \in S$ denotes the optimality region for the algorithm. The algorithm can be considered to have found an acceptable approximation to the global minimum if it discovers a point in the ε-acceptable region.

ACCEPTED MANUSCRIPT

Algorithm 1 The pseudocode of CGABC algorithm

Initialize the parameters, i.e. SN, D, $maxFEs$, $Limit$;
Generate the initial population;
Evaluate the objective function values for the population;
repeat
    Determine the neighbors of each individual by a specific topology;
    Calculate α according to Eq. (9);
    The employed bee state:
    for $i = 1 : SN$ **do**
        Calculate λ by Eq. (12);
        Determine the local best individual X_{lbest};
        Determine the local attractor S_i by Eq. (10);
        Generate a candidate solution V_i by Eq. (8);
        if $f(X_i) > f(V_i)$ **then**
            Set $X_i = V_i$, $trial_i = 0$;
            if $f(X_{gbest}) > f(V_i)$ **then**
                Set $X_{gbest} = V_i$;
            end if
        else
            Set $trial_i = trial i + 1$
        end if
    end for
    Calculate Pr according to Eq. (7), and set $t = 0$, $i = 1$;
    Determine the neighbors of each individual by a specific topology;
    The onlooker bee state:
    while $t \leq SN$ **do**
        if $rand(0, 1) < Pr_i$ **then**
            Calculate λ by Eq. (12);
            Determine the local best solution X_{lbest} and global best solution X_{gbest};
            Determine the local attractor S_i by Eq. (11);
            Generate a candidate solution V_i by Eq. (8);
            if $f(X_i) > f(V_i)$ **then**
                Set $X_i = V_i$, $trial_i = 0$;
                if $f(X_{gbest}) > f(V_i)$ **then**
                    Set $X_{gbest} = V_i$;
                end if
            else
                Set $trial_i = trial i + 1$
            end if
            $t = t + 1$, $i = i + 1$
            if $i > SN$ **then**
                $i = 1$
            end if
        end if
    end while
    The scout state:
    if $max(trial_i) > Limit$ **then**
        replace X_i with a randomly generated solution by Eq. (1)
    end if
until termination condition is met.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Lemma 1. *(Borel-Cantelli Lemmas) Let* $\{E_k\}_{k=1}$ *be a sequence of events occurring in a probability space. Let* $p_k = P(E_k)$ *. If* $\sum_{k=1}^{\infty} p_k < \infty$ *holds, then* $P\left\{\bigcap_{n=1}^{\infty}\bigcup_{k\geq n}^{\infty} E_k\right\} = 0$ *; Similarly, suppose that* $\{E_k\}_{k=1}$ *is a sequence of independent events in a probability space. If* $\sum_{k=1}^{\infty} p_k = \infty$ *holds, then* $P\left\{\bigcap_{n=1}^{\infty}\bigcup_{k\geq n}^{\infty} E_k\right\} = 1$ *.*

Thus, the lemma alleges that the set of all outcomes with the characteristic of being repeated infinitely many times must occur with probability zero if the sum of the probabilities of the events E_k is finite. Otherwise, the probability that infinitely many of them occur is 1 if the events E_k are independent and the sum of the probabilities of E_k diverges to infinity.

4.2. Global convergence criteria for random search algorithms

Hypothesis 1. $f(G(z,\xi)) \leq f(z)$ *and* $f(G(z,\xi)) \leq f(\xi)$ *if* $\xi \in S$ *.*

Here, $G(\cdot)$ function represents a random search algorithm and must satisfy condition (H1).

Hypothesis 2. *For any Borel subset* A *of* S *with its measure* $v[A] > 0$ *, we have*

$$\prod_{k=0}^{\infty}(1-\mu_k(A)) = 0 \tag{15}$$

(H2) indicates that, given any subset A of S with positive Lebesgue measure v, the probability that repeatedly missing the subset A must be zero with random sampling. Therefore, for random search algorithm, we can induce the necessary and sufficient condition of global convergence according to (H1) and (H2), as the following theorem.

Theorem 1. *(Global search) Given that* f *is a measurable function,* S *is a measurable subset of* R^N *and (H1) and (H2) are satisfied. Let* $\{z_k\}_{k=0}^{\infty}$ *be a sequence generated by the random search algorithm, then*

$$\lim_{k\to\infty} P[z_k \in R_\varepsilon] = 1 \tag{16}$$

where $P[z_k \in R_\varepsilon]$ *denotes the probability that the point* z_k *is in the* ε*-acceptable region* R_ε *at step* k*.*

Based on theorem 1, the algorithms can be proved to converge onto the global minimum when hypotheses (H1) and (H2) are satisfied.

4.3. Global convergence analysis of CGABC

This subsection aims to demonstrate that CGABC algorithm is a global convergence algorithm, namely, the CGABC satisfies the hypotheses (H1) and (H2).

Lemma 2. *CGABC satisfy (H1).*

Proof. According to the property of CGABC, the function G mentioned in (H1) for CGABC is defined as

$$G(X_{gbest}^t, X_i^{t+1}) = \begin{cases} x_{gbest}^t, & if \quad f(X_i^{t+1}) \geq f(X_{gbest}^t) \\ X_i^{t+1}, & if \quad f(X_i^{t+1}) < f(X_{gbest}^t) \end{cases} \tag{17}$$

where X_i^t means the position of individual X_i at the t_{th} iteration; and the sequence $\left\{X_{gbest}^t\right\}_{t=0}^{n}$ denotes best position found by the current population so far (from the first to the t_{th} iteration); X_i^{t+1} represents the updated position for the next iteration of individual X_i^t, that is,

$$X_i^{t+1} = \begin{cases} X_i^t, & if \quad f(X_i^t) \leq f(V_i^t) \\ V_i^t, & if \quad f(X_i^t) > f(V_i^t) \end{cases} \tag{18}$$

where V_i^t is the candidate position of individual X_i at the t_{th} iteration. Therefore, the sequence $\left\{X_{gbest}^t\right\}_{t=0}^{n}$ is monotonic and the definition of X_i^{t+1} meet the conditions of (H1). This completes the proof of Lemma 2.

ACCEPTED MANUSCRIPT

For any Borel subset A of S with $v[A] > 0$, suppose that $T_0 = [z \mid z \in A]$ and $T_1 = [z \mid z \in A \setminus A]$, then the position vectors $X(t)$ generated by CGABC in S at step t should be classified into two states:

1) It is the noted state I_0 when there is at least an individual belongs to T_0;
2) It is the noted state I_1 when all individuals belong to T_1 .

Lemma 3. *(Individual state transition) assuming that Assumption 1 holds, and let q_{ij} $(i, j = 0, 1$) denotes the state transition probability of transforming from the state I_i in $X(t)$ to state I_j in $X(t+1)$, then four cases can be obtained: $q_{01} = 0$, $q_{00} = 1$, $q_{11} \leq c \in (0, 1)$ and $q_{10} \geq 1 - c \in (0, 1)$.*

Proof. Based on Lemma 2, the best solution searched by the CGABC algorithm is monotonic, thus, we can have: $q_{01} = 0$, $q_{00} = 1$. The following content is to testify the other two cases.

According to the definition of search equation in CGABC, i.e. Eq. (8), it is obviously that individuals obey Gaussian distribution, i.e. $X_{i,j} \sim N(\mu_{i,j}, \sigma_{i,j}^2)$ where $\mu_{i,j} = s_{i,j}$ and $\sigma_{i,j}^2 = \alpha \left| s_{i,j} - x_{k,j} \right|$. For any Borel subset A of S with $v[A] > 0$, we have

$$P\left\{X_i \in A\right\} = \prod_{j=1}^{D} P\left\{x_{i,j} \in A\right\} = \prod_{j=1}^{D} \oint_A \frac{1}{\sqrt{2\pi}\sigma_{i,j}} exp(-\frac{(y-\mu_{i,j}^2)}{2\sigma_{i,j}^2}) dy \tag{19}$$

Let $P(X_i) = P(X_i \in A)$, $\forall i \in SN$. Because the positions of individuals are bounded, it can be observed that $E(X_i) = \mu_i < \infty$ where μ_i represents the mean vector of individual X_i , thus $0 < P(X_i) < 1$. Let the subset $\hat{s} \in S$ which contains SN individuals of the swarm. If there exists $l \in \hat{s}$ in finite set $\hat{s}$, and then

$$P(X_l) = min_{i \in \hat{s}} P(X_l) \tag{20}$$

From Eq. (20), it is clear that $P(X_l) \leq P(X_i) \leq q_{10}$ while $q_{10} + q_{11} = 1$. Therefore, given a constant $\gamma \in (0, 1)$, we have

$$q_{11} = 1 - q_{10} \leq 1 - P(X_l) = \gamma \tag{21}$$

This completes the proof of Lemma 3.

Lemma 4. *CGABC can satisfy (H2).*

Proof. First, let us define $\bar{P}(X_i^t) = P\left\{X_i^t \in S \setminus A\right\}$, $\forall i \in \{1, 2, .., SN\}$. By Lemma 3, it can be deduced that

$$\bar{P}(X_i^t) = \begin{cases} 0, & if \quad X_i^t \in A, \exists t \in \{0, 1, ..., t\} \\ q_{11}^t, & if \quad X_i^t \in S \setminus A, \forall t \in \{0, 1, ..., t\} \end{cases} \tag{22}$$

Therefore,

$$\sum_{t=1}^{\infty} q_{11}^t \leq \sum_{t=1}^{\infty} r^t = \frac{r}{1-r} < \infty \tag{23}$$

Note that q_{11}^t means the t_{th} power of q_{11}. Based on Lemma 1 (i.e. Borel-Cantelli Lemmas), for any $i \in \{1, 2, ..., SN\}$, we have

$$P\left\{\bigcap_{n=1}^{\infty} \bigcup_{t \geq n}^{\infty} X_i^t \in S \setminus A\right\} = 0 \tag{24}$$

This indicates that the probability of repeatedly missing the acceptable region A must be zero. Hence, Eq. (24) implies that

$$\prod_{t=0}^{\infty} (1 - \mu_t(A)) = 0 \tag{25}$$

Therefore, CGABC satisfies (H2), which completes the proof of Lemma 4.

Theorem 2. *The CGABC algorithm is a global convergence algorithm.*

Proof. Since CGABC satisfies both (H1) and (H2) and based on Theorem 1, it can be concluded that CGABC is global convergent.

ACCEPTED MANUSCRIPT

5. Experiments and analysis

5.1. Test suites

With the aim to make an overall investigation of the CGABC algorithm, experiments are conducted on 32 numerical benchmarks functions with different characteristics (unimodal, multimodal, separated, shifted, rotated, and noisy) from [16, 44]. F1-F20 are classical benchmark functions which have been frequently utilized for easy comparisons [16, 29, 32], and F21-F32 are CEC 2005 functions [44]. Besides, two real-world problems, referred to CEC 2010 [9], and a feature selection problem [3], are used to validate the performance of CGABC on real applications.

5.1.1. Benchmark functions

The function name, search range, global optimum and main properties of these 32 benchmark functions are described in Table 3 [16, 44]. Their mathematic formulas are listed in the Appendix, where $z = M * (x - o)$; x, o, M are the original variable, the shifted global optimum and the orthogonal rotation matrix, respectively. The function can be shifted when o is not the origin of coordinates. The matrix M can transform the function from separability to inseparability while keeping the properties of the original function unchangeable. Certainly the function is non-rotated if matrix M is a D-dimensional identity matrix. In the Appendix A, the Accept (column 3) [16] defined for each benchmark function is used to verify whether the current trial is successful. A trial is regarded as a success if the result obtained is within the actual global optimum (column 4 in Table 3) and the Accept value [16].

5.1.2. Real-world applications

In order to evaluate the performance of the CGABC algorithm, three real-world problems are employed here, which are the Lennard-Jones potential problem, the parameter estimation for a frequency-modulated (FM) sound wave synthesis problem and the feature selection problem. These three problems are simulated as optimization problems. The first two problems use continuous variables, while the last one is a binary problem.

1) Lennard-Jones potential problem

The Lennard-Jones potential problem, as a potential energy minimization problem, aims to find an optimal organization of atoms with minimum potential energy. This is a multimodal problems, and the local minima exponentially increase with increasing number of atoms. Therefore, it is difficult for optimization algorithms to slove this problems.

Assuming that there are N atoms in a cluster, and the Cartesian coordinates of each atom can be given as

$$P_i = \{x_i, y_i, z_i\}, i \in \{1, 2, ..., N\} \tag{26}$$

Therefore, the dimension of each individual associated with the number of atoms should be $3 * N$. The potential energy of a cluster is calculated by the summation of paired interactive actions among atoms, which is denoted as

$$f = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (\mu_{ij}^{-12} - \mu_{ij}^{-6}) \tag{27}$$

where μ_{ij} means the Euclidean distance between the i_{th} atom and the j_{th} atom.

In this subsection, experiments are conducted with different number of atoms from 7 to 15 [40], and the corresponding number of maximum fitness evaluations are 50000, 50000, 60000, 70000, 80000, 90000, 100000, 100000 and 100000.

2) Frequency-Modulated (FM) sound wave synthesis problem

The Frequency-Modulated (FM) sound wave synthesis plays a crucial role in several modern music systems, which aims at generating sounds similar to target sounds. It is a highly complex multimodal problem with optimum value being 0. The formula for the estimated sound and target sound waves are represented as:

$$y(t) = \alpha_1 \cdot sin(\omega_1 \cdot t \cdot \theta - \alpha_2 \cdot sin(\omega_2 \cdot t \cdot \theta + \alpha_3 \cdot sin(\omega_3 \cdot t \cdot \theta))) \tag{28}$$

$$y_0(t) = 1.0 \cdot sin(5.0 \cdot t \cdot \theta - 1.5 \cdot sin(4.8 \cdot t \cdot \theta + 2.0 \cdot sin(4.9 \cdot t \cdot \theta))) \tag{29}$$

where $\theta = 2\pi/100$. The objective of this problem is to minimize the summation of the squared errors between the estimated wave and target wave as follows:

$$f(\vec{X}) = \sum_{t=0}^{100} (y(t) - y_0(t))^2 \tag{30}$$

Table 3. Properties of benchmark functions

No.	Function Name	Range	Optimum	Modality	Separable	Shifted	Rotated	Noisy
F1	Sphere	$[-100, 100]^D$	0	U	Y	N	N	N
F2	Elliptic	$[-100, 100]^D$	0	U	Y	N	N	N
F3	Sumsquare	$[-10, 10]^D$	0	U	Y	N	N	N
F4	Schwefel226	$[-1, 1]^D$	0	U	Y	N	N	N
F5	Schwefel222	$[-10, 10]^D$	0	U	Y	N	N	N
F6	Schwefel221	$[-100, 100]^D$	0	U	Y	N	N	N
F7	Step	$[-100, 100]^D$	0	U	Y	N	N	N
F8	QuarticWN	$[-1.28, 1.28]^D$	0	U	Y	N	N	N
F9	Schaffer	$[-100, 100]^D$	0	M	N	N	N	N
F10	Rosenbrock	$[-10, 10]^D$	0	M	N	N	N	N
F11	Rastrigin	$[-5.12, 5.12]^D$	0	M	Y	N	N	N
F12	Ncrastrigin	$[-5.12, 5.12]^D$	0	M	Y	N	N	N
F13	Griewank	$[-600, 600]^D$	0	M	N	N	N	N
F14	Sumpower	$[-10, 10]^D$	0	M	N	N	N	N
F15	Ackley	$[-32, 32]^D$	0	M	N	N	N	N
F16	Penalized1	$[-50, 50]^D$	0	M	N	N	N	N
F17	Penalized2	$[-50, 50]^D$	0	M	N	N	N	N
F18	Alphine	$[-10, 10]^D$	0	M	Y	N	N	N
F19	Levy	$[-10, 10]^D$	0	M	N	N	N	N
F20	Weierstrass	$[-0.5, 0.5]^D$	0	M	Y	N	N	N
F21	Shifted Sphere	$[-100, 100]^D$	0	U	Y	Y	N	N
F22	Shifted Schwefel 1.2	$[-100, 100]^D$	0	U	N	Y	N	N
F23	Shifted Rotated High Conditioned Elliptic	$[-100, 100]^D$	0	U	N	Y	Y	Y
F24	Shifted Noise Schwefel 1.2	$[-100, 100]^D$	0	U	N	Y	N	N
F25	Schwefel 2.6 with Optimum on Bounds	$[-100, 100]^D$	0	U	N	N	N	N
F26	Shifted Rosenbrock	$[-100, 100]^D$	0	U	N	Y	N	N
F27	Shifted Rotated Griewank without Bounds	$[-600, 600]^D$	0	M	N	Y	Y	N
F28	Shifted Rotated Ackley with Optimum on Bounds	$[-32, 32]^D$	0	M	N	Y	Y	N
F29	Shifted Rastrigin	$[-5, 5]^D$	0	M	Y	Y	N	N
F30	Shifted Rotated Rastrigin	$[-5, 5]^D$	0	M	N	Y	Y	N
F31	Shifted Rotated Weierstrass	$[-0.5, 0.5]^D$	0	M	N	Y	Y	N
F32	Schwefel 2.13	$[-\pi, \pi]^D$	0	M	N	N	N	N

ACCEPTED MANUSCRIPT

where the parameters of $\vec{X} = \{\alpha_1, \omega_1, \alpha_2, \omega_2, \alpha_3, \omega_3\}$ are defined in the range $[-6.4, 6.35]$. Here, the maximum number of fitness evaluations are set as 6×10^6 for comparison.

3) Feature selection problem

Feature selection, as a preprocessing tool in classification tasks, can select a subset of meaningful features from a dataset and abandon the redundant ones [3]. Given a dataset, let D denotes the number of all features. Due to the fact that these features may be noisy, misleading and irrelevant, EAs are used to extract an optimal subset containing d relevant features from the original feature set in order to improve the classification accuracy.

A binary vector is employed to represent the features. 1 indicates that the corresponding feature is chosen, while 0 indicates that it is abandoned. The whole population is controlled to search in [0, 1]. Let $X_i = \{x_{i,1}, x_{i,2}, ..., x_{i,D}\}$ denotes the i_{th} individual, and the corresponding feature subset is $F_i = \{f_{i,1}, f_{i,2}, ..., f_{i,D}\}$. A straightforward strategy, as shown in Eq. (31) where j={ 1,2,...,D}, is used to constrain each element of an individual to 0 and 1.

$$f_{i,j} = \begin{cases} 0, & if \quad x_{i,j} \in (0.5, 1] \\ 1, & if \quad x_{i,j} \in [0, 0.5] \end{cases} \tag{31}$$

Feature selection is used to solve classification problems in this paper [10]. Five groups of datasets for classification problems are employed, which are taken from the UCI Repository [34]. Moreover, K-nearest neighbor (KNN) technique [3] and leave-one-out cross-validation (LOOCV) are used to evaluate the performance of each individual [40]. The classification accuracy is viewed as the objective values, which is calculated as the correctly classified samples to all samples. This is a maximization problem, because higher accuracy means better performance of selected feature subset.

Table 4. Parameter settings for all algorithms used in the comparisons

Algorithms	Parameter setting	Ref	Algorithms	Parameter setting	Ref
ABC	$Limit = 1.0 * SN * D$	[24]	DE	$DE/rand/1/bin, F = 0.8, CR = 0.8$	[43]
GABC	$c = 1.5, Limit = 1.0 * SN * D$	[50]	jDE	$\tau_1 = \tau_2 = 0.1, F_l = 0.1, F_u = 0.9$	[5]
MABC	$p = 0.7, Limit = 1.0 * SN * D$	[17]	JADE	$DE/current - to - pbest/, p = 0.05, c = 0.1$	[49]
ABC/best/1	$Limit = 0.6 * SN * D$	[18]	SaDE	$DE/rand/1/bin, F = N(0.5, 0.3), CR = N(CR_{m_k}, 0.1), CR_{m_0} = 0.5$	[37]
ASF-MR	$Limit = 200, MR = 0.4, SF = 1$	[1]	FIPS	$\chi = 0.729, \sum c_i = 4.1$	[36]
qABC	$r = 1, Limit = 0.5 * SN * D$	[25]	HPSO-TVAC	$V : V_{max} - 0.1V_{max}, c_1 = c_2 = 2$	[39]
EABC	$A = 1, \mu = \sigma = 0.3, Limit = 200$	[20]	CLPSO	$\omega : 0.9 - 0.4, c = 1.49445, m = 7$	[33]
ABCVSS	$Limit = 1.0 * SN * D$	[29]	FPSO	$\omega : 0.9 - 0.4, \sum c_i = 4$	[11]
ABCM	$M = 2, Limit = 1.0 * SN * D$	[32]	OLPSO-G	$\omega : 0.9 - 0.4, c = 2.0, G = 5; V_{maxd} = 0.2 * Range$	[48]
BABC	$Limit = 200$	[16]	PSO	$\omega : 0.9 - 0.4, c_1 = c_2 = 2.0$	[27]
distABC	$TSD = 10^{-5}, Limit = 1.0 * SN * D$	[4]	GBBPSO	–	[26]
OCABC	$M = 2 \lceil log2(D + 1) \rceil, Limit = 200$	[19]	MCS	$\rho = 3/2, p_a = 0.25$	[15]
APABC	$SN_{max} = 50, SN_{min} = 20, T = 20$	[8]	MTABC	$Limit = 200$	[42]
HABC	$Limit = 150, F_{adapt} = 50, F_{reproduce} = 5, \eta = 0.5$	[7]			

5.2. Experimental settings

For all of the compared algorithms, the population size is set to be 100, namely the colony size SN is 50. The dimension D of all tested benchmarks functions is 30, and the parameter $Limit$ of CGABC is set as 100. Each algorithm runs 50 times independently for each objective function listed in Table 3.

Moreover, five groups of experiments are designed for analysis and comparison in terms of benchmark functions. Aiming to investigate the effect of the proposed strategies (i.e. CA model, Gaussian-based search equation and modified parameter adaptation), the first group of experiments is to make comparisons among ABC and five new ABC variants, including ABC with C25 cellular structured population (cABC), ABC with adaptive parameter P_r (pABC), ABC with Gaussian-based search equation (gABC), ABC with both P_r and C25 neighborhood structure (pcABC) as well as the proposed CGABC. The next two groups of experiments are conducted to investigate the influence of different grid shapes and neighborhoods appointed on the ABC algorithm. In the fourth group, CGABC is further compared with twelve state-of-art ABC algorithms, four variants of DE and five variants of PSO to validate the efficiency of CGABC. For a fair comparison, the parameter settings and control methods of these algorithms coincide exactly with their corresponding papers as listed in Table 4. For the first four groups, the maximum number of function evaluations on F1-F20 and F21-F32 are set as 150000 and 300000 [16, 44], respectively, expecting that it is set as 200000 for the comparison among CGABC and PSO variants [20].

5.3. Experimental results on benchmark functions

5.3.1. Performance comparison among ABC, cABC, pABC, pcABC, gABC and CGABC

For cABC, pcABC, and CGABC algorithms, which are embedded with the cellular structured population, the $2D$ grid shape is set as 6×8 , and the neighborhood structure applied is the neighborhood C25. The results are listed in Table 5 regarding mean best values ($Mean$) and standard deviations (Std) of the best

Table 5. Mean and standard deviation values of six algorithms on 15 functions

Func	Metric	ABC	pABC	cABC	pcABC	gABC	CGABC
F1	*Mean*	8.81E-16	6.51E-16	2.06E-17	2.99E-21	1.16E-41	**3.88E-75**
	Std	1.59E-16	9.63E-17	1.79E-17	5.08E-21	6.91E-41	**5.69E-75**
F2	*Mean*	9.89E-09	2.41E-15	2.74E-09	5.28E-17	2.59E-37	**1.26E-71**
	Std	9.01E-09	1.77E-15	3.48E-09	6.91E-17	1.79E-36	**1.83E-71**
F3	*Mean*	5.71E-16	5.84E-16	3.90E-19	2.04E-22	9.38E-43	**5.18E-76**
	Std	8.79E-17	8.61E-17	3.04E-19	2.62E-22	3.56E-42	**8.82E-76**
F4	*Mean*	3.41E-08	1.77E-06	2.37E+00	1.13E-07	-3.02E-12	**-3.64E-12**
	Std	1.21E-07	1.04E-05	1.67E+01	7.41E-07	8.70E-13	**0.00E+00**
F5	*Mean*	2.13E-10	5.54E-11	1.20E-10	8.01E-12	2.21E-22	**1.27E-39**
	Std	6.69E-11	1.73E-11	3.51E-11	5.82E-12	8.00E-22	**9.49E-40**
F6	*Mean*	1.28E+01	1.07E+01	1.17E+01	1.07E+01	**1.25E+00**	5.46E+00
	Std	2.87E+00	2.33E+00	2.98E+00	2.22E+00	**2.21E-01**	9.00E-01
F7	*Mean*	**0**	**0**	**0**	**0**	**0**	**0**
	Std	**0**	**0**	**0**	**0**	**0**	**0**
F8	*Mean*	1.10E-01	1.08E-01	1.12E-01	1.04E-01	**1.32E-02**	1.60E-02
	Std	2.40E-02	2.11E-02	2.07E-02	2.21E-02	**3.53E-03**	4.35E-03
F9	*Mean*	3.66E-01	3.19E-01	3.59E-01	3.22E-01	**1.97E-01**	2.28E-01
	Std	**2.16E-02**	3.26E-02	2.62E-02	3.13E-02	3.81E-02	2.63E-02
F10	*Mean*	5.80E-02	**4.28E-02**	7.05E-02	4.39E-02	6.65E-01	1.65E-01
	Std	4.71E-02	**3.10E-02**	7.15E-02	3.41E-02	1.24E+00	2.52E-01
F11	*Mean*	1.44E-14	1.42E-14	1.57E-14	4.05E-15	**0**	**0**
	Std	1.43E-14	1.56E-14	1.64E-14	6.28E-15	**0**	**0**
F12	*Mean*	2.99E-13	2.48E-13	3.82E-13	3.72E-14	**0**	**0**
	Std	4.46E-13	3.23E-13	6.95E-13	4.93E-14	**0**	**0**
F13	*Mean*	6.16E-14	3.52E-15	2.17E-14	3.95E-15	**0**	**0**
	Std	1.35E-13	5.22E-15	4.95E-14	1.23E-14	**0**	**0**
F14	*Mean*	1.23E-16	9.23E-14	2.13E-21	1.33E-21	5.80E-61	**1.83E-133**
	Std	1.23E-16	1.96E-13	3.97E-21	3.23E-21	2.98E-60	**9.53E-133**
F15	*Mean*	1.51E-09	1.56E-09	9.85E-10	2.72E-10	1.56E-14	**1.06E-14**
	Std	5.59E-10	5.26E-10	3.82E-10	1.93E-10	3.11E-15	**2.74E-15**

ACCEPTED MANUSCRIPT

Table 6. AVENs and successful rates of six algorithms on 15 functions

Func	Metric	ABC	pABC	cABC	pcABC	gABC	CGABC
F1	*AVEN*	9.02E+04	7.03E+04	8.64E+04	6.70E+04	2.69E+04	**2.56E+04**
	SR	**100**	**100**	**100**	**100**	**100**	**100**
F2	*AVEN*	1.50E+05	1.01E+05	1.44E+05	9.69E+04	3.46E+04	**3.25E+04**
	SR	62	**100**	92	**100**	**100**	**100**
F3	*AVEN*	8.15E+04	6.44E+04	7.93E+04	6.15E+04	2.54E+04	**2.38E+04**
	SR	**100**	**100**	**100**	**100**	**100**	**100**
F4	*AVEN*	-	-	-	-	8.70E+04	**5.65E+04**
	SR	84	72	72	90	**100**	**100**
F5	*AVEN*	1.30E+05	1.18E+05	1.28E+05	1.10E+05	4.00E+04	**3.81E+04**
	SR	**100**	**100**	**100**	**100**	**100**	**100**
F6	*AVEN*	4.54E+04	2.84E+04	4.41E+04	2.72E+04	1.40E+04	**1.36E+04**
	SR	**100**	**100**	**100**	**100**	**100**	**100**
F7	*AVEN*	3.56E+04	2.95E+04	3.78E+04	2.88E+04	1.27E+04	**1.21E+04**
	SR	**100**	**100**	**100**	**100**	**100**	**100**
F8	*AVEN*	-	-	-	-	**2.46E+04**	2.51E+04
	SR	28	34	20	44	**100**	**100**
F9	*AVEN*	-	-	-	-	-	-
	SR	-	-	-	-	-	-
F10	*AVEN*	**4.14E+04**	5.70E+04	4.87E+04	5.63E+04	5.27E+04	6.03E+04
	SR	**100**	**100**	**100**	**100**	96	96
F11	*AVEN*	1.14E+05	1.14E+05	1.23E+05	1.03E+05	8.46E+04	**6.26E+04**
	SR	**100**	**100**	**100**	**100**	**100**	**100**
F12	*AVEN*	1.31E+05	1.27E+05	1.27E+05	1.14E+05	5.85E+04	**3.51E+04**
	SR	**100**	**100**	**100**	**100**	**100**	**100**
F13	*AVEN*	1.10E+05	9.46E+04	1.13E+05	9.25E+04	8.18E+04	**7.29E+04**
	SR	**100**	**100**	**100**	**100**	**100**	**100**
F14	*AVEN*	7.79E+04	4.70E+04	7.25E+04	4.63E+04	1.79E+04	**1.70E+04**
	SR	**100**	**100**	**100**	**100**	**100**	**100**
F15	*AVEN*	1.40E+05	1.38E+05	1.39E+05	1.29E+05	4.29E+04	**4.06E+04**
	SR	**100**	**100**	**100**	**100**	**100**	**100**

solutions obtained on fifteen selected functions. The superior algorithm for each function is in boldface. The convergence curves of different ABCs are plotted in Fig. 8. To further provide contrasts in the robustness and convergence rate, the comparison associated with successful rate (SR) [16], average function evaluation number ($AVEN$) [16] and acceleration rate (AR) [22] are conducted and reported in Table 6 and 7, respectively. In particular, SR denotes the percentage of successful runs in all independent runs on each function; $AVEN$ means the average number of function evaluations required to reach the threshold defined as Accept in the Appendix, which is only associated with those successful optimizations; the acceleration rate (AR) denotes the ratio of $AVEN$ of algorithm1 to that of algorithm2 relating to algorithm1 vs algorithm2. Further, the nonparametric Wilcoxons rank-sum test aims to estimate whether the comparisons between two algorithms (i.e. algorithm1 vs. algorithm2) are significantly different at a 5% significance level, where '+', '-', and '=' stands for that algorithm1 is significantly superior, inferior, or identical to algorithm2, respectively. The results of Wilcoxon rank-sum test are listed in Table 8. The term *'b/w/e'* denotes the number of functions that algorithm1 is significantly better than, worse than, or equal to algorithm2; and the term *gm* denotes *general merits*, which calculated by the difference between the number of significantly superior and significantly worse functions so as to provide a clear comparison between algorithm1 and algorithm2 [14, 47].

From Table 5, we can see that ABC performs worst among all the algorithms. CGABC achieves the best performance on 11 functions except F6, F8, F9 and F10, where gABC performs best on the first three functions and pABC does well on the last one. As can be inferred from Table 6 and 7, CGABC can obtain 100% successful rate and converges faster than other algorithms on most cases expect for F8, F9 and F10

ACCEPTED MANUSCRIPT

where F9 cannot be successfully solved by all algorithms and the results of CGABC on F8 and F10 are slightly worse than that of ABC. As a whole, the superior performance achieved by CGABC should be mainly attributed to the reasonable cooperation of the three proposed strategies as illustrated in section 3.2.

For the comparison between pABC and ABC, pABC converges faster than ABC because of the lower acceleration rates listed in Table 7, although the comparison is not so obvious with regard to accuracy and successful rate. It can be seen that the performance of cABC is the same as that of pABC as confirmed in Table 8, indicating that the cellular structured population and adaptive probability calculation proposed in this paper have advantages for the convergence rate. From Tables 7-8, an interesting result is that pcABC performs much better than both pABC and cABC, which informs us that combination of cellular neighborhood and adaptive parameter is an effective way to enhance the performance of ABC. Moreover, in terms of mean values and acceleration rate shown in Tables 5 and 7, gABC performs better than other four algorithms in general, although gABC cannot surpass CGABC on most functions except F6, F8 and F9. Therefore, it can be observed that the performance of gABC dramatically benefits from the effectiveness of Gaussian-based search equation with the newly defined local attractor.

From the convergence curves plotted in Fig. 8, it is evident that the CGABC converges fastest, followed by gABC, and ABC converges the slowest. The convergence rate of other three algorithms are between ABC and gABC, which supports the earlier discussions in section 3.1 and 3.2.

Table 7. Acceleration rate comparisons among six algorithms on 15 functions

AR	F1	F2	F3	F4	F5	F6	F7	F8
pABC vs. ABC	0.78	0.67	0.79	-	0.9	0.62	0.83	-
cABC vs. ABC	0.96	0.96	0.97	-	0.98	0.97	1.06	-
pABC vs. cABC	0.81	0.7	0.81	-	0.92	0.64	0.78	-
pcABC vs. ABC	0.74	0.65	0.75	-	0.84	0.6	0.81	-
pcABC vs. pABC	0.95	0.96	0.96	-	0.93	0.96	0.98	-
pcABC vs. cABC	0.78	0.67	0.78	-	0.86	0.62	0.76	-
gABC vs. ABC	0.3	0.23	0.31	-	0.31	0.31	0.36	-
gABC vs. pcABC	0.4	0.36	0.41	-	0.36	0.52	0.44	-
CGABC vs. ABC	0.28	0.22	0.29	-	0.29	0.3	0.34	-
CGABC vs. pABC	0.36	0.32	0.37	-	0.32	0.48	0.41	-
CGABC vs. cABC	0.3	0.23	0.3	-	0.3	0.31	0.32	-
CGABC vs. pcABC	0.38	0.34	0.39	-	0.35	0.5	0.42	-
CGABC vs. gABC	0.95	0.94	0.94	0.65	0.95	0.97	0.95	1.02
AR	F9	F10	F11	F12	F13	F14	F15	
pABC vs. ABC	-	1.38	1	0.97	0.86	0.6	0.99	
cABC vs. ABC	-	1.18	1.08	0.97	1.03	0.93	0.99	
pABC vs. cABC	-	1.17	0.93	0.99	0.84	0.65	1	
pcABC vs. ABC	-	1.36	0.9	0.87	0.84	0.59	0.92	
pcABC vs. pABC	-	0.99	0.91	0.9	0.98	0.99	0.93	
pcABC vs. cABC	-	1.16	0.84	0.9	0.82	0.64	0.93	
gABC vs. ABC	-	1.27	0.74	0.45	0.74	0.23	0.31	
gABC vs. pcABC	-	0.94	0.82	0.51	0.88	0.39	0.33	
CGABC vs. ABC	-	1.46	0.55	0.27	0.66	0.22	0.29	
CGABC vs. pABC	-	1.06	0.55	0.28	0.77	0.36	0.29	
CGABC vs. cABC	-	1.24	0.51	0.28	0.65	0.23	0.29	
CGABC vs. pcABC	-	1.07	0.61	0.31	0.79	0.37	0.32	
CGABC vs. gABC	-	1.14	0.74	0.6	0.89	0.95	0.95	

ACCEPTED MANUSCRIPT

Table 8. Wilcoxon rank-sum test results of comparisons among six algorithms on 15 functions

Wilcoxon rank-sum test	F1	F2	F3	F4	F5	F6	F7	F8
pABC vs. ABC	+	+	=	-	+	+	=	=
cABC vs. ABC	+	+	+	=	+	=	=	=
pABC vs. cABC	-	+	-	=	+	=	=	=
pcABC vs. ABC	+	+	+	=	+	+	=	=
pcABC vs. pABC	+	+	+	+	+	=	=	=
pcABC vs. cABC	+	+	+	=	+	=	=	+
gABC vs .ABC	+	+	+	+	+	+	=	+
gABC vs. pcABC	+	+	+	+	+	+	=	+
CGABC vs. ABC	+	+	+	+	+	+	=	+
CGABC vs. pABC	+	+	+	+	+	+	=	+
CGABC vs. cABC	+	+	+	+	+	+	=	+
CGABC vs. pcABC	+	+	+	+	+	+	=	+
CGABC vs. gABC	+	+	+	+	+	-	=	-
Wilcoxon rank-sum test	F9	F10	F11	F12	F13	F14	F15	*b/w/e/gm*
pABC vs. ABC	+	=	=	=	+	-	=	6/2/7/4
cABC vs. ABC	=	=	=	=	=	+	+	6/0/9/6
pABC vs. cABC	+	=	=	=	+	-	-	4/4/7/0
pcABC vs. ABC	+	=	+	+	+	+	+	11/0/4/11
pcABC vs. pABC	=	=	+	+	-	+	+	9/1/5/8
pcABC vs. cABC	+	=	+	+	+	+	+	11/0/4/11
gABC vs. ABC	+	-	+	+	+	+	+	13/1/1/12
gABC vs. pcABC	+	-	+	+	+	+	+	13/1/1/12
CGABC vs. ABC	+	=	+	+	+	+	+	13/0/2/13
CGABC vs. pABC	+	=	+	+	+	+	+	13/0/2/13
CGABC vs. cABC	+	=	+	+	+	+	+	13/0/2/13
CGABC vs. pcABC	+	=	+	+	+	+	+	13/0/2/13
CGABC vs. gABC	-	+	=	=	=	+	+	8/3/4/5

Table 9. Radiuses and ratios of four gird shapes

Neighborhood	4×12		5×10		6×8		7×7	
	rad_{shape}	*Ratio*	rad_{shape}	*Ratio*	rad_{shape}	*Ratio*	rad_{shape}	*Ratio*
L5	3.6429	0.2455	3.2879	0.272	3.116	0.287	2.8284	0.3162
L9	3.6429	0.4092	3.2879	0.4534	3.116	0.4784	2.8284	0.527
C9	3.6429	0.317	3.2879	0.3512	3.116	0.3706	2.8284	0.4082
C13	3.6429	0.4029	3.2879	0.4464	3.116	0.471	2.8284	0.5189
C21	3.6429	0.494	3.2879	0.5473	3.116	0.5775	2.8284	0.6362
C25	3.6429	0.549	3.2879	0.6083	3.116	0.6418	2.8284	0.7071

5.3.2. Performance comparison of CGABC with four different grid shapes

In this group of experiments, the performance of CGABC with different grid shapes and fixed number of neighbors is studied. With a constant or approximated size of colony (i.e.50), four $2D$ grid shapes used here are: (a) $4 \times 12 = 48$; (b) $5 \times 10 = 50$; (c) $6 \times 8 = 48$; (d) $7 \times 7 = 49$, and the corresponding CGABC with these grid shapes are denoted as CGABC-G1, CGABC-G2, CGABC-G3, CGABC-G4. For clarity, the ratios for different grid shapes are listed in Table 9. Neighborhood structure C25 is employed for this group of experiments.

The results are given in Table 10 with respect to mean best values and standard deviations out of 50 runs. Based on Table 10, we can see that all the four algorithms can reach the global optima on functions F7, F12, F13, and F14, and CGABC-G3 performs the best on most unimodal functions, involving F1, F2, F3 and F5. However,

the comparisons on multimodal functions are unclear due to the statistical results in Table 11, except for functions F8 and F14 where CGABC-G1 and CGABC-G3 achieve the best results, respectively. From Fig. 9, CGABC-G3 outperforms all other algorithms, while CGABC-G2 performs the worst. Meanwhile, the performance of the other two algorithms is between CGABC-G3 and CGABC-G2. It seems that the relationship between the ratios and the results are not evident.

Table 10. Performance comparisons among the CGABC with four grid shapes (best results in bold)

F	Metric	CGABC-G1	CGABC-G2	CGABC-G3	CGABC-G4
F1	*Mean*	1.12E-77	3.88E-75	**7.72E-78**	2.53E-76
	Std	1.47E-77	5.69E-75	**1.23E-77**	4.13E-76
F2	*Mean*	4.59E-74	1.26E-71	**1.46E-74**	3.81E-73
	Std	6.52E-74	1.83E-71	**1.95E-74**	6.05E-73
F3	*Mean*	**1.75E-78**	5.18E-76	2.73E-78	1.96E-77
	Std	**2.76E-78**	8.82E-76	4.18E-78	2.57E-77
F4	*Mean*	**-3.64E-12**	**-3.64E-12**	**-3.64E-12**	**-3.64E-12**
	Std	**0**	**0**	**0**	**0**
F5	*Mean*	1.02E-40	1.27E-39	**5.13E-41**	1.65E-40
	Std	8.21E-41	9.49E-40	**4.20E-41**	1.36E-40
F6	*Mean*	5.42E+00	5.46E+00	**5.24E+00**	5.55E+00
	Std	9.80E-01	**9.00E-01**	9.69E-01	9.14E-01
F7	*Mean*	**0**	**0**	**0**	**0**
	Std	**0**	**0**	**0**	**0**
F8	*Mean*	1.48E-02	1.60E-02	**1.46E-02**	1.52E-02
	Std	**3.77E-03**	4.35E-03	4.30E-03	4.11E-03
F9	*Mean*	2.30E-01	**2.28E-01**	2.34E-01	2.30E-01
	Std	3.37E-02	**2.63E-02**	2.84E-02	3.07E-02
F10	*Mean*	4.07E-01	**1.65E-01**	2.78E-01	2.36E-01
	Std	1.12E+00	**2.52E-01**	5.72E-01	4.08E-01
F11	*Mean*	**0**	**0**	**0**	**0**
	Std	**0**	**0**	**0**	**0**
F12	*Mean*	**0**	**0**	**0**	**0**
	Std	**0**	**0**	**0**	**0**
F13	*Mean*	**0**	**0**	**0**	**0**
	Std	**0**	**0**	**0**	**0**
F14	*Mean*	4.99E-138	1.83E-133	**1.59E-139**	6.93E-137
	Std	3.29E-137	9.53E-133	**5.38E-139**	3.28E-136
F15	*Mean*	1.09E-14	1.06E-14	1.08E-14	**1.05E-14**
	Std	**2.73E-15**	2.74E-15	2.87E-15	2.96E-15

Table 11. The Wilcoxon rank-sum test results when comparing CGABC with four grid shapes

Wilcoxon Rank-sum test	F1	F2	F3	F4	F5	F6	F7	F8
CGABC-G3 vs. CGABC-G1	=	+	=	=	+	=	=	=
CGABC-G3 vs. CGABC-G2	+	+	+	=	+	=	=	=
CGABC-G3 vs. CGABC-G4	+	+	+	=	+	=	=	=
Wilcoxon Rank-sum test	**F9**	**F10**	**F11**	**F12**	**F13**	**F14**	**F15**	***b/w/e/gm***
CGABC-G3 vs. CGABC-G1	=	=	=	=	=	+	=	3/0/12/3
CGABC-G3 vs. CGABC-G2	=	=	=	=	=	+	=	5/0/10/5
CGABC-G3 vs. CGABC-G4	=	=	=	=	=	+	=	5/0/10/5

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

5.3.3. Performance comparisons of CGABC with six different neighborhood structures

In this subsection, six neighborhoods in Fig. 5 are used in CGABC to find out the impact of these neighborhoods on the performance of CGABC, and the algorithms are represented as CGABC-L5, CGABC-L9, CGABC-C9, CGABC-C13, CGABC-C21, and CGABC-C25 respectively. The fixed $2D$ grid shape for all algorithms is $6 \times 8 = 48$ according to the results obtained in the former subsection. Mean best values and standard deviations over 50 runs are given in Table 12 and the rank-sum test results are given in Table 13.

From Table 12, it can be noted that all of the algorithms can reach the global optima on four functions, including F7, F11, F12 and F13. According to the statistical results in Table 13, both CGABC-C25 and CGABC-21 show the best performance and the difference between them is not obvious. CGABC-L5 performs the worst, while the performance of the other three algorithms are between CGABC-C25 and CGABC-L5. According to the mean best curves presented in Fig. 10, it is easy to conclude that the performance of these six algorithms shows a promising trend, which has relation to the ratios of the neighborhood as presented in Table 2. The neighborhood C25 has the largest ratio, while L5 holds the lowest ratio. It reveals that there is an inherent relationship between the ratios of neighborhood and the performance of algorithms associated.

5.3.4. Performance comparison among CGABC and the state-of-art ABC variants

To study the efficiency of the CGABC algorithm, this group of experiments are designed to compare CGABC with 15 state-of-art ABC variants. These ABC variants are ABC, GABC, MABC, ABC/best/1, ASF-MR, qABC, EABC, ABCVSS, ABCM, BABC, distABC, OCABC, APABC, HABC, and MTABC. The mean best values and standard deviation on 50 runs of each algorithm on each function are reported in Tables 14-17, and the statistical

Table 12. Performance comparisons among CGABC with six neighborhoods (best results in bold)

Func	Metric	CGABC-L5	CGABC-L9	CGABC-C9	CGABC-C13	CGABC-C21	CGABC-C25
F1	*Mean*	1.92E-72	2.41E-76	9.98E-76	2.86E-77	1.34E-77	**7.72E-78**
	SD	2.70E-72	4.04E-76	1.26E-75	4.26E-77	2.09E-77	**1.23E-77**
F2	*Mean*	1.21E-69	4.90E-73	2.01E-72	8.80E-74	**1.42E-74**	1.46E-74
	SD	1.11E-69	6.48E-73	2.36E-72	1.62E-73	2.09E-74	**1.95E-74**
F3	*Mean*	2.01E-73	2.45E-77	1.11E-76	3.62E-78	**9.35E-79**	2.73E-78
	SD	2.48E-73	3.28E-77	1.37E-76	5.67E-78	**1.78E-78**	4.18E-78
F4	*Mean*	**-3.64E-12**	**-3.64E-12**	**-3.64E-12**	**-3.64E-12**	**-3.64E-12**	**-3.64E-12**
	SD	**0**	**0**	**0**	**0**	**0**	**0**
F5	*Mean*	6.46E-38	4.04E-40	1.06E-39	1.17E-40	5.58E-41	**5.13E-41**
	SD	4.65E-38	2.38E-40	8.63E-40	6.88E-41	**3.79E-41**	4.20E-41
F6	*Mean*	7.18E+00	6.17E+00	6.39E+00	5.95E+00	5.28E+00	**5.24E+00**
	SD	9.88E-01	8.99E-01	1.21E+00	9.52E-01	**8.78E-01**	9.69E-01
F7	*Mean*	**0**	**0**	**0**	**0**	**0**	**0**
	SD	**0**	**0**	**0**	**0**	**0**	**0**
F8	*Mean*	1.99E-02	1.61E-02	1.61E-02	1.57E-02	**1.44E-02**	1.46E-02
	SD	4.73E-03	4.15E-03	3.92E-03	3.94E-03	**3.62E-03**	4.30E-03
F9	*Mean*	2.51E-01	2.39E-01	2.43E-01	2.37E-01	**2.29E-01**	2.34E-01
	SD	3.11E-02	2.73E-02	**2.43E-02**	3.00E-02	3.14E-02	2.84E-02
F10	*Mean*	9.87E-02	1.28E-01	8.28E-02	**5.68E-02**	2.80E-01	2.78E-01
	SD	1.49E-01	2.25E-01	1.15E-01	**8.58E-02**	4.74E-01	5.72E-01
F11	*Mean*	**0**	**0**	**0**	**0**	**0**	**0**
	SD	**0**	**0**	**0**	**0**	**0**	**0**
F12	*Mean*	**0**	**0**	**0**	**0**	**0**	**0**
	SD	**0**	**0**	**0**	**0**	**0**	**0**
F13	*Mean*	**0**	**0**	**0**	**0**	**0**	**0**
	SD	**0**	**0**	**0**	**0**	**0**	**0**
F14	*Mean*	4.81E-112	4.53E-126	9.30E-129	3.95E-134	8.51E-138	**1.59E-139**
	SD	2.52E-111	3.15E-125	5.68E-128	1.68E-133	3.13E-137	**5.38E-139**
F15	*Mean*	1.26E-14	1.20E-14	1.15E-14	1.13E-14	**1.02E-14**	1.08E-14
	SD	**1.60E-15**	2.12E-15	2.06E-15	2.79E-15	2.93E-15	2.87E-15

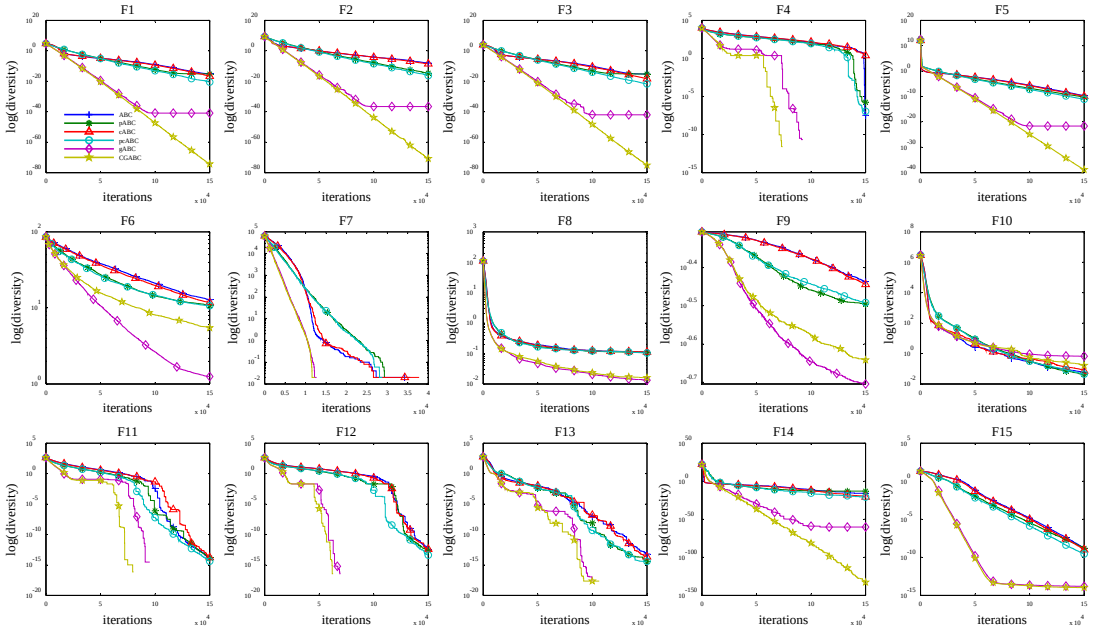


Fig. 8. Convergence curves of six algorithms on 15 benchmark functions

Table 13. The Wilcoxon rank-sum test results when comparing CGABC with six neighborhoods

Wilcoxon Rank-sum test	F1	F2	F3	F4	F5	F6	F7	F8
CGABC-C25 vs. CGABC-L5	+	+	+	=	+	+	=	+
CGABC-C25 vs. CGABC-L9	+	+	+	=	+	+	=	=
CGABC- C25vs. CGABC-C9	+	+	+	=	+	+	=	=
CGABC- C25vs. CGABC-C13	+	+	=	=	+	+	=	=
CGABC- C25vs. CGABC-C21	=	=	-	=	=	=	=	=
Wilcoxon Rank-sum test	F9	F10	F11	F12	F13	F14	F15	*b/w/e/gm*
CGABC-C25 vs. CGABC-L5	+	=	=	=	=	+	+	9/0/6/9
CGABC-C25 vs. CGABC-L9	=	=	=	=	=	+	+	7/0/8/7
CGABC- C25vs. CGABC-C9	=	=	=	=	=	+	=	6/0/9/6
CGABC- C25vs. CGABC-C13	=	=	=	=	=	+	=	5/0/10/5
CGABC- C25vs. CGABC-C21	=	=	=	=	=	+	=	1/1/13/0

results are given in Tables 18 and 19.

As seen from Table 14-17, all algorithms have achieved the global optima on F7 and F20, and CGABC performs the best on 15 functions, which are F1, F3, F4, F5, F7, F11, F12, F13, F14, F16, F17, F19, F20, F21, F29. The algorithms OCABC, MABC, EABC, ABC/best/1, ASF-MR, distABC, GABC, BABC, ABCVSS, ATABC and MTABC yield better results on nineteen, ten, ten, nine, seven, seven, six, six, five, eight, and seven functions, and the remaining algorithms obtain the best results on less than five functions. Although the number of functions on which OCABC performs better is larger than that of CGABC, OCABC performs worse than CGABC due to the statistical results in Table 18-19. Generally, CGABC shows the best performance among the fifteen algorithms for solving the 32 benchmark functions, based on the results in Tables 18-19.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Table 14. The performance comparison on F1-F8 among CGABC and other 15 ABC variants (best in bold)

Algorithms	F1	F2	F3	F4	F5	F6	F7	F8
ABC	8.72E-16	1.27E-09	5.60E-16	2.22E-01	2.04E-10	1.00E+01	**0**	6.20E-02
	1.59E-16	1.13E-09	6.72E-17	1.57E+00	6.38E-11	2.47E+00	**0**	1.20E-02
BABC	6.95E-13	3.32E-05	5.64E-40	1.90E-11	2.07E-22	**1.55E-02**	**0**	5.93E-02
	3.09E-12	1.17E-04	1.40E-39	6.46E-11	1.43E-22	**1.20E-02**	**0**	1.51E-02
ASFMR	1.11E-23	1.89E-20	1.09E-24	3.85E+03	5.36E-17	4.48E-02	**0**	1.17E-01
	9.34E-24	1.85E-20	8.36E-25	3.44E+02	2.22E-17	2.28E-02	**0**	3.68E-02
MABC	3.31E-40	6.73E-37	3.01E-41	-1.82E-12	1.40E-21	4.44E+00	**0**	2.85E-02
	2.47E-40	4.49E-37	1.62E-41	**0**	4.07E-22	5.62E-01	**0**	6.12E-03
ABCVSS	2.51E-17	7.19E-10	4.88E-19	2.35E-09	1.47E-10	3.60E+00	**0**	4.87E-02
	2.90E-17	7.99E-10	6.31E-19	1.13E-08	8.05E-11	2.12E+00	**0**	1.86E-02
EABC	1.73E-54	7.10E-51	1.34E-55	-3.09E-12	1.37E-28	1.67E+00	**0**	2.05E-02
	1.31E-54	4.99E-51	8.62E-56	8.42E-13	5.16E-29	2.71E-01	**0**	4.51E-03
OCABC	1.24E-62	3.17E-58	1.79E-63	**-3.64E-12**	1.40E-32	7.84E-01	**0**	**2.18E-04**
	1.40E-62	3.91E-58	2.23E-63	**0**	8.30E-33	3.79E-01	**0**	**9.24E-05**
qABC	1.62E-09	4.28E-07	2.30E-10	1.75E+02	6.23E-06	3.71E+01	**0**	1.28E-01
	2.61E-09	4.32E-07	3.28E-10	1.44E+02	2.46E-06	3.38E+00	**0**	3.29E-02
ABCM	2.05E-05	5.68E-02	2.81E-06	3.23E+02	1.87E-03	1.36E+01	**0**	6.48E-02
	4.91E-05	1.56E-01	5.97E-06	1.30E+02	2.23E-03	3.71E+00	**0**	1.25E-02
distABC	3.77E-49	**1.03E-197**	7.91E-48	-1.91E+06	2.73E-26	3.85E+00	**0**	2.87E-02
	2.10E-48	**0**	1.58E-47	8.30E+06	2.21E-26	5.90E-01	**0**	5.49E-03
ABC/best/1	7.17E-50	9.32E-15	2.80E-50	-2.69E-12	1.23E-25	5.22E+00	**0**	2.03E-02
	1.42E-49	6.25E-14	3.41E-50	9.18E-13	5.90E-26	2.53E+00	**0**	4.23E-03
GABC	3.81E-16	3.31E-16	4.09E-16	-2.73E-12	1.20E-15	2.83E+00	**0**	2.71E-02
	5.46E-17	6.17E-17	3.46E-17	9.19E-13	1.04E-16	4.85E-01	**0**	4.98E-03
APABC	2.70E-49	3.13E-47	5.88E-57	2.37E+00	2.53E-30	1.61E-01	**0**	1.14E-02
	1.91E-48	2.21E-46	4.16E-56	1.67E+01	1.79E-29	3.98E-02	**0**	2.93E-03
HABC	1.16E-16	2.58E-08	2.12E-18	7.57E+00	2.22E-10	2.23E+01	**0**	1.41E-01
	8.84E-17	4.01E-08	2.70E-18	2.85E+01	9.52E-11	4.65E+00	**0**	3.48E-02
MTABC	6.33E-52	3.33E-47	4.15E-53	2.37E+00	2.41E-31	6.31E-01	**0**	1.50E-02
	6.08E-52	4.39E-47	3.01E-53	1.67E+01	1.01E-31	3.26E-01	**0**	3.85E-03
CGABC	**3.88E-75**	1.26E-71	**5.18E-76**	**-3.64E-12**	**1.27E-39**	5.46E+00	**0**	1.60E-02
	5.69E-75	1.83E-71	**8.82E-76**	**0**	**9.49E-40**	9.00E-01	**0**	4.35E-03

ACCEPTED MANUSCRIPT

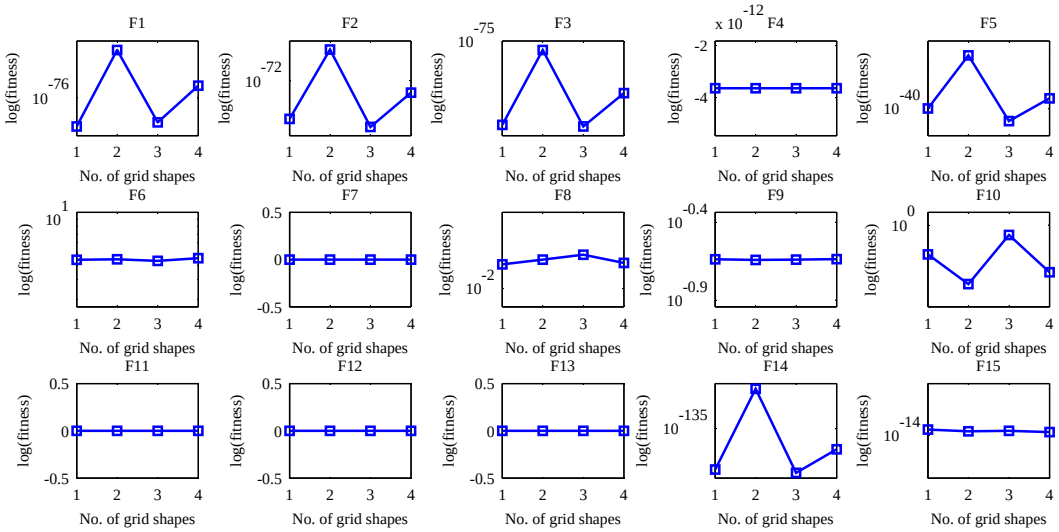


Fig. 9. The results of CGABC with four different gird shapes (the numbers from 1 to 4 on the X-axis represent: (1) CGABC-G1; (2) CGABC-G2; (3) CGABC-G3; (4) CGABC-G4)

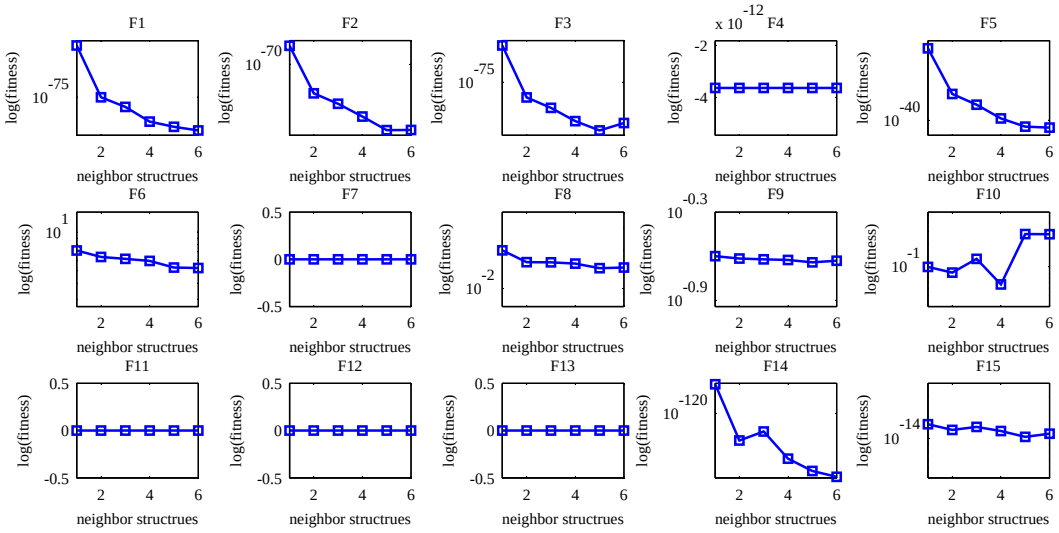


Fig. 10. The results of CGABC with six different neighborhoods (the numbers from 1 to 6 on the X-axis represent: (1) CGABC-L5; (2) CGABC-C9; (3) CGABC-L9; (4) CGABC-C13; (5) CGABC-C21; (6) CGABC-C25)

Table 15. The performance comparison on F9-F16 among CGABC and other 15 ABC variants (best in bold)

Algorithms	F9	F10	F11	F12	F13	F14	F15	F16
ABC	3.53E-01	**4.10E-02**	2.16E-14	4.99E-13	1.09E-13	1.33E-17	1.61E-09	5.20E-16
	2.29E-02	**3.97E-02**	2.35E-14	6.91E-13	2.15E-13	5.74E-18	5.92E-10	7.96E-17
BABC	3.33E-01	1.22E-01	**0**	**0**	3.66E-16	8.28E-55	2.83E-14	9.07E-13
	2.49E-02	1.42E-01	**0**	**0**	2.18E-15	5.62E-54	2.62E-15	3.74E-12

Continued on next page

Table 15 – continued from previous page

Algorithms	F9	F10	F11	F12	F13	F14	F15	F16
ASFMR	4.83E-01	2.14E+01	2.60E+01	4.17E+01	8.11E-13	9.46E-56	7.31E-01	1.33E-04
	2.82E-03	3.62E+00	3.79E+00	8.47E+00	5.72E-12	3.58E-55	9.71E-01	6.73E-04
MABC	2.28E-01	4.00E-01	**0**	**0**	**0**	1.74E-69	2.85E-14	**1.57E-32**
	2.03E-02	3.92E-01	**0**	**0**	**0**	4.50E-69	2.47E-15	**5.53E-48**
ABCVSS	3.36E-01	6.09E-02	1.57E-14	8.85E-14	1.10E-14	3.20E-23	8.42E-10	2.14E-19
	3.91E-02	5.29E-02	2.03E-14	1.38E-13	2.48E-14	6.73E-23	6.34E-10	2.37E-19
EABC	2.11E-01	8.55E-01	**0**	**0**	**0**	3.01E-76	1.91E-14	**1.57E-32**
	3.24E-02	1.86E+00	**0**	**0**	**0**	1.54E-75	2.24E-15	**5.53E-48**
OCABC	**3.85E-02**	1.27E+01	**0**	**0**	**0**	3.59E-79	**5.72E-15**	**1.57E-32**
	1.54E-02	2.64E+01	**0**	**0**	**0**	1.26E-78	**1.25E-15**	**5.53E-48**
qABC	4.52E-01	4.99E-01	7.95E-08	7.68E-04	7.79E-08	1.61E-14	7.04E-05	9.56E-11
	1.03E-02	4.90E-01	2.24E-07	5.30E-03	1.29E-07	2.89E-14	3.29E-05	1.39E-10
ABCM	3.51E-01	3.85E+00	4.80E-01	2.05E-06	8.35E-03	7.95E-14	9.27E-05	1.38E-08
	2.41E-02	4.42E+00	5.10E-01	4.35E-06	9.56E-03	4.24E-13	1.10E-04	3.41E-08
distABC	2.44E-01	1.56E-01	**0**	**0**	**0**	1.26E-113	2.94E-14	5.30E-10
	2.31E-02	1.19E-01	**0**	**0**	**0**	7.84E-113	2.51E-15	5.80E-11
ABC/best/12	2.02E-01	9.11E+00	**0**	**0**	2.22E-17	7.70E-91	1.04E-14	**1.57E-32**
	2.66E-02	1.89E+01	**0**	**0**	1.05E-16	2.66E-90	3.42E-15	**5.53E-48**
GABC	2.28E-01	9.28E-01	**0**	**0**	**0**	5.89E-18	3.08E-14	3.75E-16
	3.20E-02	2.20E+00	**0**	**0**	**0**	2.57E-18	2.86E-15	5.19E-17
APABC	2.09E-01	2.13E-01	**0**	**0**	6.37E-14	1.26E-80	3.10E-14	**1.57E-32**
	3.57E-02	2.38E-01	**0**	**0**	2.29E-13	8.75E-80	4.09E-15	**5.53E-48**
HABC	3.82E-01	5.62E-02	8.90E-13	1.05E-12	6.64E-07	3.37E-19	1.77E-09	1.10E-18
	2.46E-02	5.71E-02	5.98E-12	4.24E-12	4.69E-06	1.89E-18	9.56E-10	1.66E-18
MTABC	1.92E-01	8.50E+00	**0**	**0**	1.61E-13	7.56E-63	1.75E-14	**1.57E-32**
	3.50E-02	2.15E+01	**0**	**0**	1.14E-12	5.32E-62	4.29E-15	**5.53E-48**
CGABC	2.28E-01	1.65E-01	**0**	**0**	**0**	**1.83E-133**	1.06E-14	**1.57E-32**
	2.63E-02	2.52E-01	**0**	**0**	**0**	**9.53E-133**	2.74E-15	**5.53E-48**

Table 16. The performance comparison on F17-F24 among CGABC and other 15 ABC variants (best in bold)

Algorithms	F17	F18	F19	F20	F21	F22	F23	F24
ABC	8.28E-16	4.98E-06	2.00E-13	**0**	3.90E-16	5.16E+03	6.97E+06	3.49E+04
	1.81E-16	2.81E-06	2.24E-13	**0**	5.42E-17	1.79E+03	2.18E+06	5.87E+03
BABC	1.53E-12	7.44E-17	1.35E-31	**0**	6.10E-11	4.43E+03	7.51E+06	3.90E+04
	8.07E-12	1.88E-16	1.77E-46	**0**	2.37E-10	1.53E+03	2.43E+06	5.40E+03
ASFMR	4.34E-25	1.38E-01	8.21E-26	**0**	**0**	**2.19E+00**	**1.43E+06**	5.14E+04
	3.51E-25	3.27E-01	8.52E-26	**0**	**0**	**8.01E-01**	**3.00E+05**	8.61E+03
MABC	**1.50E-33**	4.47E-21	**1.35E-31**	**0**	**0**	1.03E+04	1.21E+07	3.63E+04
	0	3.23E-21	**1.77E-46**	**0**	**0**	2.08E+03	3.39E+06	4.52E+03
ABCVSS	3.29E-17	3.63E-06	1.12E-13	**0**	**0**	5.41E+03	5.78E+06	3.34E+04
	4.25E-17	3.21E-06	1.51E-13	**0**	**0**	1.53E+03	1.95E+06	5.50E+03
EABC	**1.50E-33**	1.63E-28	**1.35E-31**	**0**	**0**	9.65E+03	1.55E+07	3.36E+04
	0	1.45E-28	**1.77E-46**	**0**	**0**	2.22E+03	4.01E+06	4.42E+03
OCABC	**1.50E-33**	**1.76E-33**	**1.35E-31**	**0**	**0**	5.48E+02	8.77E+06	**2.22E+03**
	0	**1.46E-33**	**1.77E-46**	**0**	**0**	1.60E+02	2.13E+06	**5.12E+02**
qABC	9.73E-10	3.79E-04	4.87E-08	**0**	4.67E-16	7.30E+03	9.74E+06	4.85E+04
	1.20E-09	3.08E-04	3.57E-08	**0**	6.15E-17	3.04E+03	3.26E+06	7.96E+03
ABCM	1.77E-09	2.53E-04	6.98E-07	**0**	3.40E-08	4.38E+03	5.79E+06	3.53E+04
	4.66E-09	2.16E-04	1.08E-06	**0**	1.64E-07	1.44E+03	1.77E+06	5.05E+03
distABC	7.91E-09	3.70E-05	1.48E-06	**0**	4.11E-08	3.96E+03	5.44E+06	2.74E+04

Continued on next page

ACCEPTED MANUSCRIPT

Table 16 – continued from previous page

Algorithms	F17	F18	F19	F20	F21	F22	F23	F24
	1.05E-09	7.96E-06	8.02E-07	**0**	4.99E-09	1.17E+03	1.85E+06	4.56E+03
ABC/best/1	**1.50E-33**	5.44E-17	**1.35E-31**	**0**	**0**	5.24E+03	7.54E+06	3.18E+04
	0	1.54E-16	**1.77E-46**	**0**	**0**	2.43E+03	2.89E+06	6.06E+03
GABC	3.94E-16	6.19E-07	3.13E-16	**0**	2.90E-16	5.53E+03	6.78E+06	3.09E+04
	5.00E-17	9.02E-07	6.40E-17	**0**	5.95E-17	2.20E+03	2.39E+06	6.08E+03
APABC	**1.50E-33**	2.08E-33	**1.35E-31**	**0**	**0**	1.14E+04	1.55E+07	3.09E+04
	0	7.42E-33	**1.77E-46**	**0**	**0**	3.00E+03	3.35E+06	5.32E+03
HABC	1.32E-16	6.07E-06	2.95E-13	**0**	2.60E+04	2.47E+04	2.48E+04	2.42E+04
	1.04E-16	5.69E-06	4.03E-13	**0**	5.55E+03	5.35E+03	5.39E+03	4.98E+03
MTABC	**1.50E-33**	3.59E-31	**1.35E-31**	**0**	9.79E+03	1.00E+04	**9.62E+03**	1.05E+04
	0	1.36E-30	**1.77E-46**	**0**	3.59E+03	3.64E+03	**3.24E+03**	3.33E+03
CGABC	**1.50E-33**	5.73E-16	**1.35E-31**	**0**	**0**	1.07E+04	1.57E+07	3.55E+04
	0	7.80E-16	**1.77E-46**	**0**	**0**	4.48E+03	5.10E+06	4.39E+03

Table 17. The performance comparison on F25-F32 among CGABC and other 15 ABC variants (best in bold)

Algorithms	F25	F26	F27	F28	F29	F30	F31	F32
ABC	1.03E+04	1.49E+00	4.70E+03	2.08E+01	**0**	3.25E+02	2.79E+01	8.66E+03
	1.36E+03	1.81E+00	2.63E-12	5.29E-02	**0**	4.68E+01	1.89E+00	3.08E+03
BABC	6.38E+03	1.56E+01	4.70E+03	2.10E+01	**0**	1.33E+02	2.84E+01	2.10E+04
	1.07E+03	1.29E+01	2.61E-12	5.67E-02	**0**	2.58E+01	1.73E+00	4.58E+03
ASFMR	5.90E+03	6.12E+01	4.74E+03	**2.01E+01**	5.25E+01	9.30E+01	3.52E+01	**9.30E+02**
	7.04E+02	2.74E+01	1.54E+01	**2.39E-02**	1.04E+01	1.38E+01	1.72E+00	**9.46E+02**
MABC	8.50E+03	1.61E+00	4.70E+03	2.08E+01	**0**	1.64E+02	2.72E+01	1.25E+04
	1.31E+03	1.99E+00	4.96E-12	5.31E-02	**0**	2.19E+01	1.89E+00	3.47E+03
ABCVSS	1.04E+04	**1.24E+00**	4.70E+03	2.08E+01	**0**	2.56E+02	2.72E+01	8.68E+03
	1.32E+03	**1.72E+00**	2.57E-12	4.74E-02	**0**	8.10E+01	1.85E+00	3.27E+03
EABC	5.93E+03	1.70E+01	4.70E+03	2.09E+01	**0**	1.15E+02	2.68E+01	1.75E+04
	7.55E+02	2.01E+01	2.55E-12	6.31E-02	**0**	1.90E+01	1.66E+00	3.73E+03
OCABC	**3.02E+03**	6.28E+01	4.70E+03	2.09E+01	**0**	**4.12E+01**	**2.64E+01**	1.95E+04
	3.33E+02	3.98E+01	4.81E-12	4.01E-02	**0**	**7.75E+00**	1.77E+00	3.81E+03
qABC	1.07E+04	2.22E+00	4.70E+03	2.09E+01	6.36E-13	3.22E+02	2.84E+01	1.42E+04
	1.12E+03	2.58E+00	8.79E-12	4.18E-02	1.30E-12	4.40E+01	2.08E+00	5.58E+03
ABCM	1.12E+04	7.95E+01	4.70E+03	2.07E+01	3.17E-10	3.52E+02	2.79E+01	9.02E+03
	1.57E+03	4.95E+01	2.50E-04	5.66E-02	1.25E-09	5.55E+01	1.86E+00	4.07E+03
distABC	8.35E+03	3.65E+00	**3.61E+00**	2.08E+01	8.03E-06	2.16E+02	2.79E+01	9.74E+03
	1.32E+03	4.52E+00	1.48E+00	5.16E-02	8.30E-07	3.47E+01	1.51E+00	2.98E+03
ABC/best/1	8.35E+03	1.38E+01	4.70E+03	2.08E+01	**0**	1.37E+02	2.70E+01	6.89E+03
	1.08E+03	1.86E+01	4.81E-12	6.33E-02	**0**	2.10E+01	1.72E+00	2.97E+03
GABC	8.44E+03	4.21E+00	4.70E+03	2.08E+01	**0**	1.63E+02	2.67E+01	8.19E+03
	1.34E+03	6.74E+00	4.21E-12	7.02E-02	**0**	2.42E+01	1.55E+00	3.79E+03
APABC	5.63E+03	2.48E+01	4.70E+03	2.09E+01	**0**	1.11E+02	2.73E+01	1.95E+04
	7.40E+02	2.58E+01	3.12E+00	4.42E-02	**0**	1.48E+01	1.96E+00	4.30E+03
HABC	2.37E+04	2.47E+04	2.60E+04	2.07E+04	1.00E+04	8.69E+03	9.66E+05	2.55E+04
	4.91E+03	5.50E+03	5.50E+03	4.39E+03	3.11E+03	3.20E+03	**6.44E-10**	5.42E+03
MTABC	9.72E+03	1.00E+04	9.47E+03	9.15E+03	9.82E+03	9.96E+03	9.66E+05	9.89E+03
	3.58E+03	2.74E+03	3.00E+03	3.55E+03	3.46E+03	3.06E+03	7.89E-10	3.40E+03
CGABC	5.69E+03	1.07E+01	4.70E+03	2.09E+01	**0**	1.10E+02	2.82E+01	2.56E+04
	7.29E+02	1.42E+01	**2.53E-12**	4.53E-02	**0**	1.98E+01	**1.46E+00**	5.88E+03

ACCEPTED MANUSCRIPT

Table 18. Wilcoxon rank sum test among CGABC and 15 ABC variants on F1-F17 (best in bold)

Algorithms	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17
CGABC vs. ABC	+	+	+	+	+	+	=	+	+	-	+	+	+	+	+	+	+
CGABC vs. BABC	+	+	+	+	+	-	=	+	+	=	=	=	+	+	+	+	+
CGABC vs. ASFMR	+	+	+	+	+	-	=	+	+	+	+	+	=	+	+	+	+
CGABC vs. MABC	+	+	+	+	+	-	=	+	=	+	=	=	=	+	+	=	=
CGABC vs. ABCVSS	+	+	+	+	+	-	=	+	+	=	+	+	+	+	+	+	+
CGABC vs. EABC	+	+	+	+	+	-	=	+	=	+	=	=	=	+	+	=	=
CGABC vs. OCABC	+	+	+	=	+	-	=	-	-	+	=	=	=	+	+	=	=
CGABC vs. qABC	+	+	+	+	+	+	=	+	+	+	+	+	+	+	+	+	+
CGABC vs. ABCM	+	+	+	+	+	+	=	+	+	+	+	+	+	+	+	+	+
CGABC vs. distABC	+	-	+	+	+	-	=	+	=	-	=	=	=	+	+	+	+
CGABC vs. ABC/best/1	+	+	+	+	+	=	=	+	-	+	=	=	=	+	=	=	=
CGABC vs. GABC	+	+	+	+	+	-	=	+	=	=	=	=	=	+	+	+	+
CGABC vs. ATABC	+	+	+	+	+	-	=	-	-	+	=	=	+	+	+	=	=
CGABC vs. HABC	+	+	+	+	+	+	=	+	+	=	+	+	+	+	+	+	+
CGABC vs. MTABC	+	+	+	=	+	-	=	=	-	+	=	=	=	+	+	=	=

Table 19. Wilcoxon rank sum test among CGABC and 15 ABC variants on F18-F32 (best in bold)

Algorithms	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30	F31	F32	*b/w/e/gm*
CGABC vs. ABC	+	+	=	+	-	-	+	+	+	=	+	=	+	=	-	23/4/5/19
CGABC vs. BABC	=	=	=	=	-	-	+	=	+	+	-	+	-	+	-	17/6/9/11
CGABC vs. ASFMR	+	+	=	=	=	-	=	+	-	-	-	=	+	-	-	18/7/7/11
CGABC vs. MABC	=	=	=	=	-	-	=	+	-	=	-	=	+	-	-	11/7/14/4
CGABC vs. ABCVSS	+	+	=	=	=	=	-	=	+	=	=	=	=	-	-	17/4/11/13
CGABC vs. EABC	=	=	=	=	-	-	-	-	+	-	=	=	-	-	-	10/9/13/1
CGABC vs. OCABC	=	=	=	+	-	-	+	+	-	+	-	+	+	=	-	12/9/11/3
CGABC vs. qABC	+	+	=	+	-	-	-	+	=	-	-	+	+	=	-	22/6/4/16
CGABC vs. ABCM	+	+	=	=	-	-	-	+	=	-	-	=	+	-	-	20/7/5/13
CGABC vs. distABC	+	+	=	+	-	-	=	+	+	+	-	+	+	=	-	17/8/8/9
CGABC vs. ABC/best/1	=	=	=	+	-	-	-	+	-	=	-	=	+	-	-	11/8/13/3
CGABC vs. GABC	+	+	=	+	-	-	=	+	-	=	-	=	+	=	-	15/6/11/9
CGABC vs. APABC	=	=	=	=	=	=	-	=	+	+	=	=	=	-	-	11/6/15/5
CGABC vs. HABC	+	+	=	+	+	-	-	+	+	+	+	+	+	+	=	26/2/4/24
CGABC vs. MTABC	=	=	=	+	=	-	-	+	+	+	+	+	+	+	-	15/5/12/10

5.3.5. Performance comparison among CGABC and the other EAs

In order to further validate the effectiveness of the CGABC algorithm, comparison with four DE variants and five PSO variants on twelve selected functions are conducted. These algorithms contains DE, jDE, JADE, SaDE, FIPS, HPSO-TVAC, CLPSO, FPSO, OLPSO-G. Note that the maximal number of function evaluations (maxFEs) for DE variants on different functions are listed in Table 20 (column 2), and the maxFEs for PSO variants is set as 200000 [20].The results listed in Tables 20 and 21 regarding the mean best values and standard deviations are taken directly from [40], which has been frequently cited [16, 19, 29].

As seen from Table 20, the comparison shows that CGABC outperforms DEs with the exception of the Quartic function on which JADE performs the best. CGABC is the only one that can reach the global optimum on function Rastrigin among the compared algorithms. In Table 21, there is a huge amount of difference between the performance of CGABC and that of other algorithms on most cases, especially on those unimodal functions where CGABC shows great performance. Besides, CGABC can obtain the optimal values on four functions, including Step, Rastrigin, NCRastrigin and Griewank functions. FPSO and OLPSO-G perform better than other algorithms on the Quartic and Ackley functions, respectively. From Table 20 and 21, it can be seen that CGABC can significantly outperform the other algorithms on almost all cases.

ACCEPTED MANUSCRIPT

Table 20. Comparison among CGABC and DE variants on twelve functions (best in bolds)

F	maxFEs	Matric	DE	jDE	JADE	SaDE	CGABC
F1	150,000	Mean	9.80E-14	1.46e28	1.32e54	3.28e20	**4.09E-75**
		Std	8.40E-14	1.78e28	9.22e54	3.63e20	**5.67E-75**
F4	100,000	Mean	5.90E+03	1.70e10	2.62e04	1.13e08	**-3.60E-12**
		Std	1.10E+03	1.70e10	3.59e04	1.08e08	**2.57E-13**
F5	200,000	Mean	1.60E-09	9.02e24	3.18e25	3.51e25	**3.25E-53**
		Std	1.10E-09	6.01e24	2.05e24	2.74e25	**2.33E-53**
F7	10,000	Mean	4.70E+03	6.13E+02	5.62E+00	5.07E+01	**2.20E-01**
		Std	1.10E+03	1.72E+02	1.87E+00	1.34E+01	**4.18E-01**
F8	300,000	Mean	4.70E-03	3.35e03	**6.14e04**	4.86e03	1.06E-02
		Std	1.20E-03	8.68e04	**2.55e04**	5.21e04	3.11E-03
F10	500,000	Mean	2.10E+00	1.04e03	1.59e01	7.98e02	**2.23E-02**
		Std	1.50E+00	1.37e03	7.89e01	5.64e01	**4.42E-02**
F11	100,000	Mean	1.80E+02	3.32e04	1.33e01	2.43E+00	**0**
		Std	1.30E+01	6.39e04	9.74e02	1.60E+00	**0**
F13	50,000	Mean	2.00E-01	7.29e06	1.57e08	2.52e09	**1.72E-14**
		Std	1.10E-01	1.05e05	1.09e07	1.24e08	**7.70E-14**
F15	50,000	Mean	1.10E-01	2.37e04	3.35e09	3.81e06	**1.89E-11**
		Std	3.90E-02	7.10e05	2.84e09	8.26e07	**8.78E-12**
F16	50,000	Mean	1.20E-02	7.03e08	1.67e15	8.25e12	**3.15E-24**
		Std	1.00E-02	5.74e08	1.02e14	5.12e12	**2.80E-24**
F17	50,000	Mean	7.50E-02	1.80e05	1.87e10	1.93e09	**1.45E-22**
		Std	3.80E-02	1.42e05	1.09e09	1.53e09	**1.36E-22**
F18	300,000	Mean	2.30E-04	6.08e10	2.78e05	2.94e06	**3.55E-16**
		Std	1.70E-04	8.36e10	8.43e06	3.47e06	**4.74E-16**

Table 21. Comparison among CGABC and PSO variants on twelve functions (best in bolds)

F	metric	FIPS	HPSO-TVAC	CLPSO	FPSO	OLPSO-G	CGABC
F1	Mean	2.42E-13	2.83E-33	1.58E-12	2.40E-16	4.12E-54	**1.62E-101**
	Std	1.73E-13	3.19E-33	7.70E-13	2.00E-31	6.34E-54	**2.14E-101**
F4	Mean	9.93E+02	1.59E+03	3.82E-04	1.34E+03	3.84E+02	**-3.64E-12**
	Std	5.09E+02	3.26E+02	1.28E-05	2.77E+02	2.17E+02	**0**
F5	Mean	2.76E-08	9.03E-20	2.51E-08	1.58E-11	9.85E-30	**2.72E-53**
	Std	9.04E-09	9.58E-20	5.84E-09	1.03E-22	1.01E-29	**1.91E-53**
F7	Mean	**0**	**0**	**0**	**0**	**0**	**0**
	Std	**0**	**0**	**0**	**0**	**0**	**0**
F8	Mean	4.24E-03	9.82E-02	5.85E-03	**4.16E-03**	1.16E-02	1.31E-02
	Std	1.28E-04	3.26E-02	1.11E-03	**2.40E-06**	4.10E-03	3.69E-03
F10	Mean	2.51E+01	2.39E+01	1.13E+01	2.81E+01	2.15E+01	**1.31E-01**
	Std	5.10E-01	2.65E+01	9.85E+00	2.31E+02	2.99E+01	**3.24E-01**
F11	Mean	6.51E+01	9.43E+00	9.09E-05	7.38E+01	1.07E+00	**0**
	Std	1.33E+01	3.48E+00	1.25E-04	3.70E+02	9.92E-01	**0**
F12	Mean	7.01E+01	1.03E+01	1.54E+00	7.03E+01	2.18E+00	**0**
	Std	1.47E+01	8.24E+00	2.75E+00	2.96E+02	6.31E-01	**0**
F13	Mean	9.01E-12	9.75E-03	9.02E-09	1.47E-03	4.83E-03	**0**
	Std	1.84E-11	8.33E-03	8.57E-09	1.28E-05	8.63E-03	**0**
F15	Mean	2.33E-07	7.29E-14	3.66E-07	2.17E-09	**7.98E-15**	8.56E-15
	Std	7.19E-08	3.00E-14	7.57E-08	**1.71E-18**	2.03E-15	2.74E-15
F16	Mean	1.96E-15	2.71E-29	6.45E-14	5.51E-18	1.59E-32	**1.57E-32**
	Std	1.11E-15	1.88E-28	3.70E-14	1.45E-34	1.03E-33	**5.53E-48**
F17	Mean	2.70E-14	2.79E-28	1.25E-12	1.37E-17	4.39E-04	**1.50E-33**
	Std	1.57E-14	2.18E-28	9.45E-12	3.42E-32	2.20E-03	**0**

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

5.4. Experimental results on three real-world problems

5.4.1. Lennard-Jones potential problem

Eight algorithms mentioned above as well as CGABC are run on the three real-world problems, including ABC, MABC, ABCVSS, DE, JADE, PSO, GBBPSO, MCS and CGABC. The mean best values and standard deviations out of 50 independent runs are presented in Table 22, and the statistical results are shown in Table 23. From Table 22, it can be easily observed that CGABC can obtain the best solution among the tested algorithms. As evident from the results in Table 23, CGABC is statistically superior when compared to all other algorithms.

For clarity, the convergence curves of these nine algorithms with N being 7 and 8 are given in Fig. 11 to illustrate the search process. As seen from curves, the proposed CGABC exhibits the best performance in the whole process, demonstrating that the CGABC algorithm can be considered as an effective tool for settling the Lennard-Jones potential problem.

Table 22. Comparison among 9 algorithms of different number of atoms on Lennard-Jones potential problem (best in bolds)

N	7	8	9	10	11	12	13	14	15
ABC	-15.2738	-18.1408	-21.2695	-24.8877	-28.6383	-32.0716	-35.7046	-38.9081	-42.3555
	0.419304	0.456627	0.494486	0.689789	0.585263	0.97964	0.937652	0.922704	1.167892
MABC	-15.2422	-18.2234	-21.617	-24.9195	-28.6559	-32.2131	-36.0323	-39.111	-42.8684
	0.317558	0.350828	0.53618	0.552199	0.708897	0.792669	0.799918	0.879837	1.218653
ABCVSS	-15.0911	-18.1128	-21.7329	-24.8341	-28.3378	-32.1409	-35.9725	-39.1301	-42.3857
	0.39007	0.495017	0.66661	0.693408	0.772914	0.767914	0.982806	0.919408	1.024813
DE	-7.57219	-8.38575	-9.3637	-10.3937	-11.4195	-12.6442	-13.9937	-14.7873	-15.4556
	0.546621	0.435862	0.67343	0.659369	0.68307	0.650433	1.074177	1.007694	0.851148
JADE	-8.13482	-9.04101	-9.5323	-10.7329	-12.4104	-13.2542	-13.8191	-14.4238	-16.2652
	2.086477	2.377023	1.777437	1.648833	3.511142	2.942856	2.401755	1.452774	3.650645
PSO	-11.5894	-12.1248	-14.5438	-17.3442	-19.4519	-21.0005	-23.4152	-25.9687	-25.5145
	1.995733	2.150686	1.938375	2.969171	3.028371	3.10588	3.156312	4.351524	4.309333
GBBPSO	-14.4621	-17.5342	-20.2161	-22.8031	-25.2476	-27.5723	-28.9203	-30.8626	-33.4785
	1.844479	2.255024	2.52454	2.703047	3.093529	3.575741	3.785422	4.247284	4.27287
MCS	-10.2308	-11.5335	-13.6292	-15.6778	-17.7477	-20.2611	-22.6459	-24.0073	-25.7011
	0.589026	0.627761	0.752724	0.783488	1.036291	1.07056	1.338691	1.129113	1.260751
CGABC	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**
	15.4219	**18.6796**	**21.8944**	**25.4084**	**28.8143**	**32.5034**	**36.1997**	**39.6407**	**42.9935**
	0.542658	0.456531	0.692997	0.860641	1.123522	1.380463	1.470512	1.710728	1.890147

Table 23. Wilcoxon rank sum test on the Lennard-Jones potential problem

N	7	8	9	10	11	12	13	14	15	*b/w/e/gm*
CGABC vs. ABC	=	+	+	+	=	=	=	+	=	4/0/5/4
CGABC vs. MABC	=	+	=	+	=	=	=	=	=	2/0/7/2
CGABC vs. ABCVSS	+	+	=	+	+	=	=	=	=	4/0/5/4
CGABC vs. DE	+	+	+	+	+	+	+	+	+	9/0/0/9
CGABC vs. JADE	+	+	+	+	+	+	+	+	+	9/0/0/9
CGABC vs. PSO	+	+	+	+	+	+	+	+	+	9/0/0/9
CGABC vs. GBBPSO	=	=	+	+	+	+	+	+	+	7/0/2/7
CGABC vs. MCS	+	+	+	+	+	+	+	+	+	9/0/0/9

5.4.2. Frequency-Modulated (FM) sound wave synthesis problem

To validate the effectiveness of our proposed algorithm, CGABC is further compared with ABC, MABC, ABCVSS, DE, JADE, PSO, GBBPSO, MCS and CGABC The best fitness values, worst fitness values, mean fitness values and their standard deviations obtained by the nine algorithms over 50 independent runs are presented in Table

24. The statistical results are shown in Table 25, including the sum of rank, z-value and P-value. The results in Tables 24 and 25 reveal that CGABC statistically outperforms the other algorithms. Both CGABC and GBBPSO can even achieve the global optima on some appropriate cases according to the best fitness values obtained, however, CGABC has great advantages in terms of the mean best value. The convergence curves plotted in Fig. 12 show that CGABC converges much faster than all the other algorithms, especially in the later search process. Here, the optimal values of $\bar{X}$ obtained by these compared algorithms are presented in Table 26.

Table 24. Best, worst, median, mean and standard deviation values obtained by nine algorithms on FM sound wave synthesis problem

Algorithms	Best	Worst	Median	Mean	Std
ABC	5.71E-03	1.07E+01	2.28E-01	1.50E+00	3.09E+00
OCABC	1.45E-27	6.87E-02	1.18E-02	1.97E-02	2.04E-02
ABCVSS	6.63E-03	9.05E+00	2.11E-01	9.67E-01	2.28E+00
distABC	2.34E-03	8.42E+00	8.21E-02	5.89E-01	1.71E+00
JADE	1.36E+01	2.13E+01	1.92E+01	1.87E+01	1.89E+00
PSO	8.50E+00	1.16E+01	1.03E+01	9.90E+00	9.79E-01
GBBPSO	**0**	1.95E+01	1.18E+01	1.12E+01	5.97E+00
MCS	6.56E-03	4.46E+00	4.05E-01	1.00E+00	1.24E+00
CGABC	**0**	**7.56E-06**	**1.24E-23**	**3.35E-07**	**1.30E-06**

Table 25. Wilcoxon rank sum test on the FM sound wave synthesis problem

Wilcoxon rank sum test	ranksum	z-value	P-value	Significance
CGABC vs. ABC	1275	-8.61621	< 0.0001	Extremely significant
CGABC vs. OCABC	1436	-7.50602	< 0.0001	Extremely significant
CGABC vs. ABCVSS	1275	-8.61621	< 0.0001	Extremely significant
CGABC vs. distABC	1275	-8.61621	< 0.0001	Extremely significant
CGABC vs. JADE	1275	-8.61621	< 0.0001	Extremely significant
CGABC vs. PSO	1275	-8.61623	< 0.0001	Extremely significant
CGABC vs. GBBPSO	1671	-5.89618	< 0.0001	Extremely significant
CGABC vs. MCS	1275	-8.61621	< 0.0001	Extremely significant

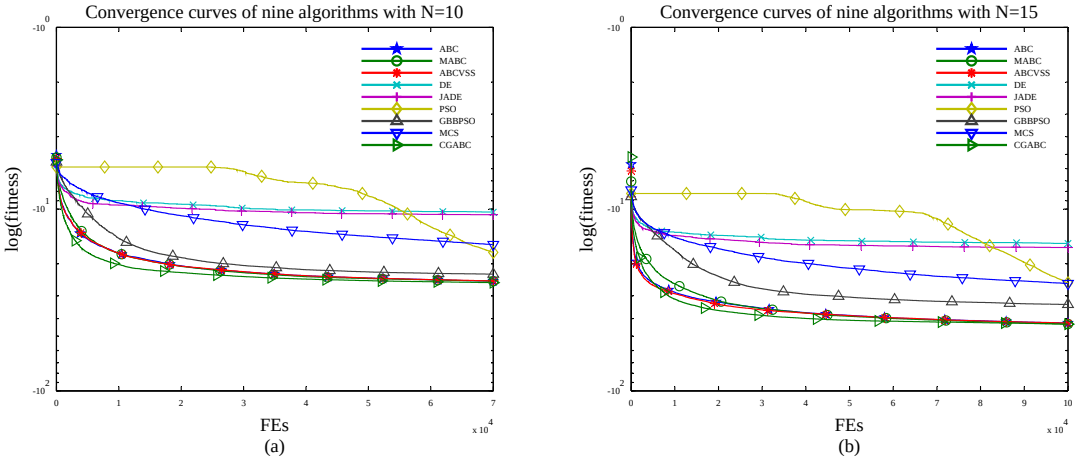


Fig. 11. Comparison among nine algorithms on Lennard Jones potential problem with 10 and 15 atoms

5.4.3. Feature selection problem

A detailed description of five datasets from UCI Repository can be found in Table 27. The CGABC is further compared with ABC, MABC, ABCVSS, distABC, JADE, PSO and MCS. The maximum number of fitness evaluations is 10000, and the population size is 80, i.e. the colony size of the CGABC is 40. The grid shape of the CGABC is 5×8, and the number of the nearest neighbors, i.e. k, in KNN classification is set as 5. The mean accuracy and mean number of selected features obtained by the eight algorithms on five UCI dataset over 50 independent runs are given in Table 28. It can be seen that all algorithms can achieve the best performance on the Parkinsons dataset, and the performance on the Glass dataset is the same. The CGABC performs best on Wine and Sonar datasets. For the Ionosphere dataset, the best-performing algorithm is PSO. Combined with the statistical results presented in Table 29, it can be concluded that the CGABC have competitive performance when solving feature selection problems. However, there is still room for improvement with respect to the number of selected features.

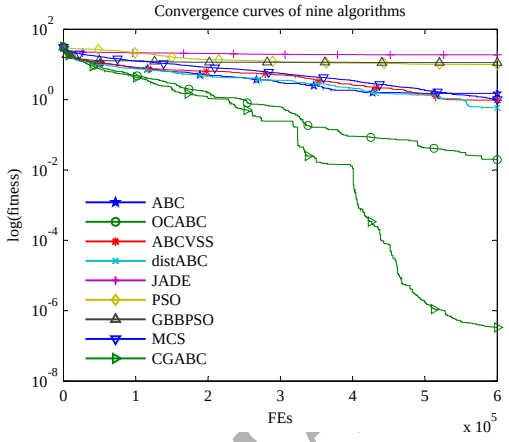


Fig. 12. Comparisons among nine algorithms on FM sound wave synthesis problem

Table 26. The optimal solutions obtained by the nine compared algorithms

Algorithm	Optimal solution
ABC	[-1.00251156,-4.99819208,-1.48660393,-4.80082690,2.00104913,-4.90103221]
OCABC	[0.99999999,5.00000000,1.50000000,-4.80000000,2.00000000,-4.90000000]
ABCVSS	[-0.99466610,-5.00464350,1.50968105,4.79665330,-2.00082423,-4.90105142]
distABC	[0.998031996,5.00067718,-1.49960400,4.79843105,2.00308671,4.89951371]
JADE	[-0.618234900,-0.0239568283,4.75758643,4.90521381,-0.171399954,0.180965386]
PSO	[-0.689434977,114.686097,0.774640062,-90.2565538,-1.05060820,-194.876992]
GBBPSO	[-1,-5,1.50000000,4.80000000,2,4.90000000]
MCS	[0.994984030,5.00367789,1.49800669,-4.79867017,2.00500683,-4.90171657]
CGABC	[1,5,-1.50000000,4.80000000,2,4.90000000]

Table 27. Descriptions of five datasets for feature selection problem

Dataset	Glass	Wine	Ionosphere	Parkinsons	Sonar
Number of samples	10	13	34	23	60
No. of Classes	7	3	2	2	2
No. of attributes	214	178	351	195	208

Table 28. Mean accuracy and mean number of selected features obtained by eight algorithms on feature selection problems

Datasets	Glass		Wine		Ionosphere		Parkinsons		Sonar	
	fn	*Accuracy*	*fn*	*Accuracy*	*fn*	*Accuracy*	*fn*	*Accuracy*	*fn*	*Accuracy*
ABC	5.5	**0.9953**	7.4	0.9618	11.1	0.9003	10.6	**1**	27.5	0.8808
MABC	**4.7**	**0.9953**	7.7	0.964	9.6	0.9123	9	**1**	28.7	0.8856
ABCVSS	5.2	**0.9953**	8.1	0.9652	**8.5**	0.9157	9.3	**1**	26.9	**0.899**
distABC	5.2	**0.9953**	7.3	0.9674	10.7	0.9085	9.5	**1**	29.3	0.887
JADE	5.4	**0.9953**	**6**	**0.9719**	9.8	0.9031	10.1	**1**	28.5	0.887
PSO	**4.7**	**0.9953**	7.9	0.964	8.9	**0.9225**	10.3	**1**	28.5	0.8963
MCS	5.4	**0.9953**	6.6	0.9697	8.7	0.9094	**8.9**	**1**	**26.4**	0.8952
CGABC	5.2	**0.9953**	**6**	**0.9719**	8.8	0.9191	9.2	**1**	27	**0.899**

Table 29. Wilcoxons rank sum test on the feature selection problem

Wilcoxon rank sum test	Glass	Wine	Ionosphere	Parkinsons	Sonar	*b/w/e/g*
CGABC vs. ABC	=	+	+	=	+	3/0/2/3
CGABC vs. MABC	=	+	+	=	+	3/0/2/3
CGABC vs. ABCVSS	=	+	+	=	=	2/0/3/2
CGABC vs. distABC	=	+	+	=	+	3/0/2/3
CGABC vs. JADE	=	=	+	=	+	2/0/3/2
CGABC vs. PSO	=	+	-	=	+	2/1/2/1
CGABC vs. MCS	=	+	+	=	+	3/0/2/3

6. Conclusion

In this paper, the ABC algorithm has been enhanced to the so-called CGABC algorithm, after elaborately analyzing the effect of the onlooker bees and scout bees with well-designed experiments. The cellular structured neighborhood was introduced to the CGABC algorithm to make individuals only interact with their neighbors while preserving the population diversity. In addition, the proposed Gaussian-based search equation with redefined local attractor can help improve the local search ability. Besides, a more intelligent and robust probability calculation method based on rank ordering was developed to determine the qualified solutions regarded as onlooker bees. We have further exploited the global convergence property of the CGABC algorithm according to the theory of probabilistic metric spaces. The experimental results conducted over 32 benchmark functions and three real-world applicatons indicate that the proposed CGABC algorithm has demonstrated superior capabilities related to accuracy, robustness and efficiency. Our future work will involve studying how to apply the CGABC algorithm to some optimization problems, such as flexible job shop problem and traveling salesman problem, etc.

Acknowledgements

This project is supported by the National Science Foundation of China (61572238), the Provincial Outstanding Youth Foundation of Jiangsu Province (BK20160001).

ACCEPTED MANUSCRIPT

Appendix A. Benchmark Functions

Function Name	Mathematical formulation	Accept
Sphere	$f(X) = \sum_{i=1}^{D} z_i^2$	1×10^{-8}
Elliptic	$f(X) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} z_i^2$	1×10^{-8}
Sumsquare	$f(X) = \sum_{i=1}^{D} i z_i^2$	1×10^{-8}
Schwefel226	$f(X) = 418.982887 * n - \sum_{i=1}^{D} i z_i sin(\sqrt{\lvert z_i \rvert})$	1×10^{-8}
Schwefel222	$f(X) = \sum_{i=1}^{D} \lvert z_i \rvert + \prod_{i=1}^{D} \lvert z_i \rvert$	1×10^{-8}
Schwefel221	$f(X) = max(\lvert z_i \rvert, 1 \leq i \leq D)$	4×10^{1}
Step	$f(X) = \sum_{i=1}^{D} (\lvert z_i + 0.5 \rvert)^2$	1×10^{-8}
QuarticWN	$f(X) = \sum_{i=1}^{D} z_i^4 + random(0, 1)$	1×10^{-1}
Schaffer	$f(X) = 0.5 + (sin^2(\sqrt{\sum_{i=1}^{D} z_i^2}) - 0.5)/(1 + 0.001 * [\sum_{i=1}^{D} z_i^2])^2$	1×10^{-8}
Rosenbrock	$f(X) = \sum_{i=1}^{D-1} [100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2]$	5×10^{0}
Rastrigin	$f(X) = \sum_{i=1}^{D-1} [z_i^2 - 10cos(2\pi z_i) + 10]$	1×10^{-8}
Ncrastrigin	$f(X) = \sum_{i=1}^{D-1} [y_i^2 - 10cos(2\pi y_i) + 10], y_i = \begin{cases} z_i, & if \quad \lvert z_i \rvert < 0.5 \\ round(2z_i)/2, & if \quad \lvert z_i \rvert \geq 0.5 \end{cases}$	1×10^{-8}
Griewank	$f(X) = 1/4000 \sum_{i=1}^{D} z_i^2 - \prod_{i=1}^{D} cos(z_i/\sqrt{i}) + 1$	1×10^{-8}
Sumpower	$f(X) = \sum_{i=1}^{D} \left\lvert z_i^{i+1} \right\rvert$	1×10^{-8}
Sumpower	$f(X) = \sum_{i=1}^{D} \left\lvert z_i^{i+1} \right\rvert$	1×10^{-8}
Ackley	$f(X) = -20exp(-0.2\sqrt{1/D * \sum_{i=1}^{D} z_i^2}) - exp(1/D * \sum_{i=1}^{D} cos(2\pi z_i)) + 20 + e$	1×10^{-8}
Penalized1	$f(X) = \frac{\pi}{D} \left\{ 10sin^2(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10sin^2(\pi y_{i+1})] + (y_D - 1)^2 \right\} + \sum_{i=1}^{D} \mu(z_i, 10, 100, 4); y_i = 1 + \frac{1}{4}(z_i + 1), \mu_{x_i,a,k,m} = \begin{cases} k(z_i - a)^m, & if \quad z_i > a \\ 0, & if \quad -a \leq z_i \leq a \\ k(-z_i - a)^m, & if \quad z_i < -a \end{cases}$	1×10^{-8}
Penalized2	$f(X) = \frac{1}{10} \left\{ sin^2(\pi z_1) + \sum_{i=1}^{D-1} (z_i - 1)^2 [1 + sin^2(3\pi z_{i+1})] + (z_D - 1)^2 [1 + sin^2(2\pi z_{i+1})] \right\} + \sum_{i=1}^{D} \mu(z_i, 5, 100, 4)$	1×10^{-8}
Alphine	$f(X) = \sum_{i=1}^{D} \lvert z_i sin(z_i) + 0.1 z_i \rvert$	1×10^{-8}
Levy	$f(X) = \sum_{i=1}^{D-1} (z_i - 1)^2 [1 + sin^2(3\pi z_{i+1})] + sin^2(3\pi z_1) + \lvert x_D - 1 \rvert [1 + sin^2(3\pi z_D)]$	1×10^{-8}
Weierstrass	$f(X) = \sum_{i=1}^{D} (\sum_{k=1}^{kmax} [a^k cos(2\pi b^k (z_i + 0.5))]) + D \sum_{k=1}^{kmax} [a^k cos(2\pi b^k)], a = 0.5, b = 3, kmax = 20$	1×10^{-8}
Schwefel 1.2	$f(X) = \sum_{i=1}^{D} (\sum_{j=1}^{i} z_j)^2$	1×10^{-8}
Noise Schwefel 1.2	$f(X) = \sum_{i=1}^{D} (\sum_{j=1}^{i} z_j)^2 * (1 + 0.4 \lvert N(0, 1) \rvert)$	1×10^{-8}
Schwefel 2.6	$f(X) = max(A_i x - A_i o), A \in R^{D \times D}$, the element of A is in [-500,500], $det(A) \neq 0; B_i = A_i * o, o \in R^{D \times 1}$, the element of o is in [-100, 100]	1×10^{-8}
Schwefel 2.13	$f(X) = \sum_{i=1}^{D} (A_i - B_i(z))^2, A_i = \sum_{j=1}^{D} (a_{ij} sin\alpha_j + b_{ij} cos\alpha_j), B_i(z) = \sum_{j=1}^{D} (a_{ij} sin z_j + b_{ij} cos z_j), A \times B \in R^{D \times D}, a_{ij} \in [-100, 100], b_{ij} \in [-100, 100]; \alpha = [\alpha_1, \alpha_2, \alpha_D], \alpha_j \in [-\pi, \pi]$	1×10^{-8}

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

References

References

[1] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Information Sciences 192 (2012) 120–142.

[2] E. Alba, J. Troya, Cellular evolutionary algorithms: Evaluating the influence of ratio, in: Parallel Problem Solving from Nature (PPSN VI), Springer, 29–38, 2000.

[3] M. G. Asl, M. R. Mobasheri, B. Mojaradi, Unsupervised feature selection using geometrical measures in prototype space for hyperspectral imagery, IEEE transactions on geoscience and remote sensing 52 (7) (2014) 3774–3787.

[4] I. Babaoglu, Artificial bee colony algorithm with distribution-based update rule, Applied Soft Computing 34 (C) (2015) 851–861.

[5] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, IEEE transactions on evolutionary computation 10 (6) (2006) 646–657.

[6] R. Chelouah, S. Patrick, A continuous genetic algorithm designed for the global optimization of multimodal functions, Journal of Heuristics 6 (2) (2000) 191–213.

[7] H. Chen, L. Ma, M. He, X. Wang, X. Liang, L. Sun, M. Huang, Artificial bee colony optimizer based on bee life-cycle for stationary and dynamic optimization, IEEE Transactions on Systems, Man, and Cybernetics: Systems 47 (2) (2017) 327–346.

[8] L. Cui, G. Li, Z. Zhu, Q. Lin, Z. Wen, N. Lu, K.-C. Wong, J. Chen, A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization, Information Sciences 414 (2017) 53–67.

[9] S. Das, P. N. Suganthan, Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems, Tech. Rep., Jadavpur University, Nanyang Technological University, Kolkata, India, 2010.

[10] M. Dash, H. Liu, Feature selection for classification, Intelligent data analysis 1 (1-4) (1997) 131–156.

[11] M. A. M. De Oca, T. Stutzle, M. Birattari, M. Dorigo, Frankenstein's PSO: a composite particle swarm optimization algorithm, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 1120–1132.

[12] L. Dos Santos Coelho, P. Alotto, Gaussian artificial bee colony algorithm approach applied to Loney's solenoid benchmark problem, IEEE Transactions on Magnetics 47 (5) (2011) 1326–1329.

[13] A. E. Eiben, J. E. Smith, Introduction to evolutionary computing. 2nd edition, Springer, 2003.

[14] W. Fang, J. Sun, H. H. Chen, et al, A decentralized quantum-inspired particle swarm optimization algorithm with cellular structured population, Information Sciences 330 (2016) 19–48.

[15] A. H. Gandomi, X.-S. Yang, A. H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Engineering with computers 29 (1) (2013) 17–35.

[16] W. F. Gao, F. T. Chan, L. L. Huang, et al, Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood, Information Sciences 316 (2015) 180–200.

[17] W. F. Gao, S. Y. Liu, Improved artificial bee colony algorithm for global optimization, Information Processing Letters 111 (17) (2011) 871–882.

[18] W. F. Gao, S. Y. Liu, L. L. Huang, A global best artificial bee colony algorithm for global optimization, Journal of Computational and Applied Mathematics 236 (11) (2012) 2741–2753.

[19] W. F. Gao, S. Y. Liu, L. L. Huang, A novel artificial bee colony algorithm based on modified search equation and orthogonal learning, IEEE Transactions on Cybernetics 43 (3) (2013) 1011–1024.

[20] W. F. Gao, S. Y. Liu, L. L. Huang, Enhancing artificial bee colony algorithm using more information-based search equations, Information Sciences 270 (2014) 112–133.

[21] F. Kang, J. J. Li, H. J. Li, Artificial bee colony algorithm and pattern search hybridized for global optimization, Applied Soft Computing 13 (4) (2013) 1781–1791.

[22] F. Kang, J. J. Li, Z. Y. Ma, Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions, Information Sciences 181 (16) (2011) 3508–3531.

[23] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Tech. Rep., Erciyes university, Melikgazi, Turkey, 2005.

[24] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of global optimization 39 (3) (2007) 459–471.

[25] D. Karaboga, B. Gorkemli, A quick artificial bee colony-qABC-algorithm for optimization problems, in: Innovations in Intelligent Systems and Applications (INISTA), 2012 International Symposium on, IEEE, 1–5, 2012.

[26] J. Kennedy, Bare bones particle swarms, in: Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE, IEEE, 80–87, 2003.

[27] J. Kennedy, Particle swarm optimization, in: Encyclopedia of machine learning, Springer, 760–766, 2011.

[28] M. S. Kıran, O. Fındık, A directed artificial bee colony algorithm, Applied Soft Computing 26 (C) (2015) 454–462.

[29] M. S. Kiran, H. Hakli, M. Gunduz, et al, Artificial bee colony algorithm with variable search strategy for continuous optimization, Information Sciences 300 (2015) 140–157.

[30] F. J. Kuang, Z. Jin, W. H. Xu, et al, A novel chaotic artificial bee colony algorithm based on tent map, in: Evolutionary Computation (CEC), 2014 IEEE Congress on, IEEE, 235–241, 2014.

[31] M. Kutrib, Cellular automata-a computational point of view, New developments in formal languages and applications 113 (2008) 183–227.

ACCEPTED MANUSCRIPT

[32] X. N. Li, G. F. Yang, Artificial bee colony algorithm with memory, Applied Soft Computing 41 (C) (2016) 362–372.
[33] J. J. Liang, A. K. Qin, P. N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE transactions on evolutionary computation 10 (3) (2006) 281–295.
[34] M. Lichman, UCI Machine Learning Repository, URL http://archive.ics.uci.edu/ml, 2013.
[35] H. B. Liu, A. Abraham, V. Snášel, Convergence analysis of swarm algorithm, in: Nature & Biologically Inspired Computing, 2009 (NaBIC 2009), World Congress on, IEEE, 1714–1719, 2009.
[36] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE transactions on evolutionary computation 8 (3) (2004) 204–210.
[37] A. K. Qin, V. L. Huang, P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE transactions on Evolutionary Computation 13 (2) (2009) 398–417.
[38] A. Rajasekhar, N. Lynn, S. Das, P. N. Suganthan, Computing with the collective intelligence of honey bees–A survey, Swarm and Evolutionary Computation 32 (2017) 25–48.
[39] A. Ratnaweera, S. K. Halgamuge, H. C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Transactions on evolutionary computation 8 (3) (2004) 240–255.
[40] Y. J. Shi, C. M. Pun, H. D. Hu, et al, An improved artificial bee colony and its application, Knowledge-Based Systems 107 (C) (2016) 14–31.
[41] F. J. Solis, R. J. Wets, Minimization by random search techniques, Mathematics of operations research 6 (1) (1981) 19–30.
[42] X. Song, Q. Yan, M. Zhao, An adaptive artificial bee colony algorithm based on objective function value information, Applied Soft Computing 55 (2017) 384–401.
[43] R. Storn, K. Price, Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization 11 (4) (1997) 341–359.
[44] P. N. Suganthan, N. Hansen, J. J. Liang, et al, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Tech. Rep., Nanyang Technological University, Singapore, 2005.
[45] J. Von Neumann, A. W. Burks, Theory of self-reproducing automata, IEEE Transactions on Neural Networks 5 (1) (1966) 3–14.
[46] S. Wolfram, A new kind of science, Wolfram media Champaign, 2002.
[47] Z. H. Zhan, J. Zhang, Y. Li, et al, Adaptive particle swarm optimization, IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics 39 (6) (2009) 1362–1381.
[48] Z.-H. Zhan, J. Zhang, Y. Li, Y.-H. Shi, Orthogonal learning particle swarm optimization, IEEE transactions on evolutionary computation 15 (6) (2011) 832–847.
[49] J. Zhang, A. C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Transactions on evolutionary computation 13 (5) (2009) 945–958.
[50] G. P. Zhu, S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, Applied Mathematics and Computation 217 (7) (2010) 3166–3173.

ACCEPTED MANUSCRIPT