Soft Comput (2018) 22:2935–2952
https://doi.org/10.1007/s00500-017-2547-1

METHODOLOGIES AND APPLICATION

# A self-adaptive artificial bee colony algorithm based on global best for global optimization

**Yu Xue**[1,2] · **Jiongming Jiang**[1] · **Binping Zhao**[1] · **Tinghuai Ma**[1]

Published online: 22 March 2017


**Abstract** Intelligent optimization algorithms based on evolutionary and swarm principles have been widely researched in recent years. The artificial bee colony (ABC) algorithm is an intelligent swarm algorithm for global optimization problems. Previous studies have shown that the ABC algorithm is an efficient, effective, and robust optimization method. However, the solution search equation used in ABC is insufficient, and the strategy for generating candidate solutions results in good exploration ability but poor exploitation performance. Although some complex strategies for generating candidate solutions have recently been developed, the universality and robustness of these new algorithms are still insufficient. This is mainly because only one strategy is adopted in the modified ABC algorithm. In this paper, we propose a self-adaptive ABC algorithm based on the global best candidate (SABC-GB) for global optimization. Experiments are conducted on a set of 25 benchmark functions. To ensure a fair comparison with other algorithms, we employ the same initial population for all algorithms on each benchmark function. Besides, to validate the feasibility of SABC-GB in real-world application, we demonstrate its application to a real clustering problem based on the $K$-means technique. The results demonstrate that SABC-GB is superior to the other algorithms for solving complex optimization problems. It means that it is a new technique to improve the ABC by introducing self-adaptive mechanism.

**Keywords** Artificial bee colony (ABC) · Global optimization · Search strategy · Self-adaptive

Communicated by V. Loia.

✉ Yu Xue
xueyu@nuist.edu.cn

Jiongming Jiang
jiongmingjiang@163.com

Binping Zhao
binpingzhao@126.com

Tinghuai Ma
thma@nuist.edu.cn

[1] School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China

[2] Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology, Nanjing University of Information Science and Technology, Nanjing 210044, China

## 1 Introduction

Optimization problems are frequently encountered in numerous science and engineering fields. Traditional problems characterized by being continues, unimodal, differentiable, and linear were widely researched prior to the 1960s. However, real-world optimization problems tend to be nonlinear, discontinuous, non-differentiable, and multimodal. Hence, traditional optimization methods, such as Newton's method (Roy and Sevick-Muraca 1999) and quasi-Newton (Setiono and Hui 1995) methods, cannot be used. The need for effective optimization algorithms with the ability to solve complex real-world optimization problems led to the development of many evolutionary algorithms (EAs), such as genetic algorithm (GAs) (Holland 1975), particle swarm optimization (PSO) (Kennedy and Eberhart 1995), ant colony optimization (ACO) (Dorigo and Gambardella 1997), differential evolution (DE) (Storn and Price 1997), artificial bee colony (ABC) (Karaboga and Basturk 2007) algorithm. EAs can avoid becoming trapped in local optima in solving many real-world optimization problems, which traditional problems cannot solve. However, EAs also may be poor at exploitation in some complex and multi-dimensional optimization problems. That

is why we want to optimize EAs.These EAs have been used to solve many practical problems (Li and Pan 2015; Yi et al. 2016).

EAs inspired by biogenetics, natural phenomena, physical phenomena, and social phenomena have been proposed. These are known as global optimization algorithms, as they avoid becoming trapped in local optima in order to find globally optimal solutions. Generally speaking, an EA starts with an initial population of candidate solutions. New solutions are then generated from solutions in the previous population. Next, the quality of each new solution is evaluated by a fitness function. Finally, a selection process is used to produce a new population of solutions. This iterative process is repeated until the optimum or near-optimum solution is reached.

The ABC algorithm is a relatively new approach that was proposed by Karaboga in 2005 (Karaboga and Basturk 2007). Since then, the interest in ABC algorithms has increased rapidly. Experimental results show that the performance of ABC is better than that of other EAs in many problems, because it has fewer control parameters and is easier to apply (Karaboga and Basturk 2007). ABC has been widely used to solve many real-world problems. For example, ABC has been applied to a loudspeaker design problem (Zhang et al. 2014), and for the design of two-channel quadrature mirror filter banks (Agrawal and Sahu 2015). ABC was also used to minimize the makespan for single machine batch processing with nonidentical job sizes (Al-Salamah 2015). Horng adopted ABC for a stochastic economic lot scheduling problem (Horng 2015), and Pan solved the large-scale hybrid flow shop scheduling problem with ABC (Li and Pan 2015). ABC has also been employed to solve an interest-based forwarding problem (Xia et al. 2015).

The ABC algorithm is a simple, efficient, effective, and robust evolutionary optimization method. As a result, ABC has emerged as a potential tool for solving local and global optimization problems (Gu and Sheng 2013; Wen et al. 2015). An increasing number of numerical benchmark functions are being employed to evaluate the performance of ABC. However, there is no specific algorithm that can achieve the best solution for all optimization problems. The ABC algorithm also has some disadvantages. For instance, it may occasionally stop proceeding toward the global optimum even though the population has not converged to a local optimum, it can struggle with certain classes of optimization problems and suffers from long computation times because of its stochastic nature.

It is well known that EAs include both exploration and exploitation strategies. However, as exploration and exploitation are inherently contradictory, they should be well balanced to ensure good performance. Thus, many new and self-adaptive ABC algorithms have been proposed (Babaoglu 2015; Bansal et al. 2013; Gao et al. 2013, 2014; He et al. 2013; Kang et al. 2013; Li and Yin 2014; Rajasekhar and Pant 2014; Liu et al. 2015). The variation equations were inspired by DE/rand/1 and DE/current-to-rand/1 (Epitropakis et al. 2011). To further enhance the convergence rate of the proposed algorithm, a self-adaptive modification rate (MR) based on a successful update probability was also proposed to generate suitable parameters. However, Liu has not conducted further research on this issue. In order to use the parameters of ABC during the evolutionary process, Bansal et al. (2013) proposed an adaptive version of ABC where the step size in solution modification and the ABC parameter limit are determined adaptively based on the current fitness values. These self-adaptive algorithms have mainly focused on determining the self-adaptive parameters, whereas the focus of this article is on the adaptive search strategies themselves. Usually, there is only one candidate solution generating strategy (CSGS) in each algorithm. The CSGS tends to be either good at exploration or good at exploitation. Thus, it is difficult to simultaneously achieve the two goals of exploration and exploitation using one CSGS. Hence, there is a need to search for an improved optimization method. In this paper, we propose a self-adaptive artificial bee colony algorithm based on the global best (SABC-GB) for global optimization. In SABC-GB, several different CSGSs are employed simultaneously. This modification allows us to tune the balance between the convergence rate and the robustness of the algorithm. As a result, it is expected that the convergence speed of SABC-GB can be accelerated to enable better solutions to be obtained within an acceptable convergence time.

The $K$-means algorithm, proposed by Macqueen (1967), is an important clustering technique (Macqueen 1967). Its goal is to divide data sets into several clusters, where the data within the same cluster are similar and those in different clusters are as dissimilar as possible. Because of its simple description and high efficiency, $K$-means has been widely used since the 1970s. However, $K$-means also has some shortcomings: It is difficult to determine the value of $K$ in advance, it can become stuck around local optimal, and it is sensitive to the initial centers. Thus, SABC-GB is applied to this real problem to overcome these shortcomings.

In this paper, the SABC-GB, which is based on self-adaptive strategies, is proposed. We modify the employed bee phase to improve the global optimal capability of the SABC-GB algorithm and use a novel probabilistic method to enhance the search ability of the onlooker bee phase. Furthermore, we change the initialization phase to avoid local minima, i.e., SABC-GB adopts chaotic systems and opposition-based learning method to initialize the population. The remainder of this paper is organized as follows. Section 2 summarizes the conventional ABC algorithm and $K$-means algorithms. The SABC-GB algorithm is present and analyzed in Sect. 3. In Sect. 4, we present and discuss

the results of a series of experiments. The paper concludes with a discussion in Sect. 5.

## 2 Related work

### 2.1 Artificial bee colony algorithm

In the ABC algorithm, the location of a food source represents a feasible solution of the optimization problem, and the quality of solution is referred to as the fitness. There are three kinds of bees in the ABC algorithm: employed bees, onlooker bees, and scout bees. In the algorithmic model, the number of employed bees is always equal to the number of onlooker bees. It is important to note that the three types of bees can transmute into each other. In the initial stage, the population $P$ is produced according to Eq. (1).

$$P = Lbound + rand_1 * (Ubound - Lbound) \tag{1}$$

where $Lbound$ and $Ubound$ are the lower and upper bounds, respectively, and $rand_1$ is a random number from 0 to 1.

Each solution of $P$ is a $D$-dimensional vector. The employed bees search for all the food sources for a maximum of MaxFES iterations. The search equation is shown as (2).

$$V_{i,j} = X_{i,j} + rand_2 * (X_{i,j} - X_{k,j}) \tag{2}$$

where $i$ represents the current individual, $k, j \in \{1, 2, \ldots, SN\}, k \neq j \neq i, rand_2 \in (-1, 1)$, and $V_{i,j}$ is the new solution in the next generation.

After gathering honey, the employed bees compare the quality of the former food source with that of the new food. If the quality of the new food is higher than that of the previous one, the bees will memorize the location of the new food source; otherwise, the old one is remained. When the search stage is finished, the employed bees go back to a dance area and transmit information about the food sources to the onlooker bees. According to this information, the onlooker bees choose good food sources from which to gather honey. The richer the source is, the higher the probability it will be selected. The computational formula is shown as (3).

$$P_i = fit_i / \sum_{k=1}^{SN} fit_k \tag{3}$$

where $fit_i$ is the fitness value of solution $I$, $SN$ is the number of individuals, and $P_i$ is the selection probability of the current solution.

Each individual corresponds to one trial counter. A food source that could not be improved through s set number of trials, referred to as the limit, is abandoned. If the trial counter of a solution exceeds the limit, the scout bee will abandon it and generate a new one.

### 2.2 Parameter optimization in using $K$-means algorithm

The $K$-means clustering algorithm has become one of the most frequently used clustering algorithms (Ji et al. 2015; Macqueen 1967). However, $K$-means is strongly affected by the initial centers. Thus, we use the proposed algorithm to determine the clustering centers.

The idea of the $K$-means algorithm is as follows: Data are classified into clusters according to the Euclidean distance to the center of the cluster. Assume the number of data objects is $n$. The sample set is a set of numeric objects $\mathrm{X} = \{x_1, x_2, \ldots, x_n\}$, and the number of clusters is $k(< n)$. The aim of the $K$-means algorithm is to search for a partition of X into $k$ clusters such that objects belonging to the same cluster are as similar to each other as possible, whereas objects belonging to different clusters are as dissimilar as possible. The process of the $K$-means algorithm is as follows:

*Step 1* Choose $k$ data objects randomly from the original dataset X as initial cluster centers

$$C = \{c_1, c_2, \ldots, c_k\} \tag{4}$$

*Step 2* According to the distance calculation Eq. (5), calculate the distances between each data object $x_j$ $(j = 1, 2, \ldots, n)$ and the selected $k$ cluster centers, defined as $\mathrm{d}\left(x_j, \mathrm{c}_i\right)$ $(i = 1, 2, \ldots, k)$. Each data object is thus classified into a cluster with the least Euclidean distance to the center of the cluster;

*Step 3* Recalculate the average value of the data objects in each cluster as a new cluster center;

*Step 4* Repeat Steps 2 and 3 until the center points can not be changed or the objective function converges.

$$\mathrm{d}\,(j, i) = \sqrt{\left(x_{j1} - x_{i1}\right)^2 + \left(x_{j2} - x_{i2}\right)^2 + \cdots + \left(x_{js} - x_{is}\right)^2} \tag{5}$$

where both $j = \left(x_{j1}, x_{j2}, \ldots, x_{js}\right)$ and

$i = (x_{i1}, x_{i2}, \ldots, x_{is})$ are $s$-dimensional data objects. The aim of the $K$-means algorithm is to minimize the following objective function:

$$\mathrm{J} = \sum_{i=1}^{k} \sum_{x_j \in c_i} \left\| x_j - c_i \right\|^2 \tag{6}$$

where $J$ represents the sum of squares of the distances between all objects in a cluster and the cluster center. $c_i$ represents the value of the $i$ th cluster center, and $x_j$ represents the $j$ th data object belonging to $c_i$.

**Table 1** Test functions

| Num | Problems | Objection function | Property |
|---|---|---|---|
| Fun1 | Shifted sphere | $\mathrm{F}_1(X) = \sum_{i=1}^{D} Z_i^2 + f\_bias_1$ | $Z = X - O$<br>Unimodal, shifted<br>Separable, scalable<br>$X_i \in [-100, 100]^D$<br>Global optimum: $X^* = O$<br>$F(X^*) = -450$ |
| Fun2 | Shifted Schwefels Problem 1.2 | $F_2(X) = \sum_{i=1}^{D} \left( \sum_{j=1}^{i} Z_j \right)^2 + f\_bias_2$ | $Z = X - O$<br>Unimodal, shifted<br>Non-separable, scalable<br>$X_i \in [-100, 100]^D$<br>Global optimum: $X^* = O$<br>$F(X^*) = -450$ |
| Fun3 | Shifted rotated High conditioned Elliptic | $F_3(X) = \sum_{i=1}^{\mathrm{D}} (10^6)^{\frac{i-1}{D-1}} Z_i^2 + f\_bias_3$ | $Z = (X - O) * M$<br>Unimodal, shifted<br>Rotated, non-separable,<br>Scalable<br>$X_i \in [-100, 100]^D$<br>Global optimum: $X^* = O$<br>$F(X^*) = -450$ |
| Fun4 | Shifted Schwefels Problem 1.2 with Noise in fitness | $F_4(X) = \left( \sum_{i=1}^{D} \left( \sum_{j=1}^{i} Z_j \right)^2 \right) * (1 + 0.4 \lvert N(0, 1) \rvert) + f\_bias_4$ | $Z = (X - O) * M$<br>Unimodal, shifted<br>Rotated, non-separable,<br>Scalable<br>Noise in fitness<br>$X_i \in [-100, 100]^D$<br>Global optimum: $X^* = O$<br>$F(X^*) = -450$ |
| Fun5 | Problem 2.6 with Global optimum on Bounds | $F_5(X) = \max \{ \lvert A_i x - B_i \rvert \} + f\_bias_5$ | Unimodal<br>Non-separable<br>Scalable<br>$X_i \in [-100, 100]^D$<br>Global optimum: $X^* = O$<br>$F(X^*) = -310$ |
| Fun6 | Shifted Rosenbrock | $F_6(X) = \sum_{i=1}^{D-1} (100(Z_i^2 - Z_{i+1})^2 + (Z_i - 1)^2) + f\_bias_6$ | $Z = X - O + 1$<br>Basic multimodal<br>Shifted, non-separable<br>Scalable<br>$X_i \in [-100, 100]^D$<br>Global optimum: $X^* = O$<br>$F(X^*) = -390$ |
| Fun7 | Shifted rotated Griewanks Function without Bounds | $F_7(X) = \sum_{i=1}^{D-1} \frac{Z_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{Z_i}{\sqrt{i}}\right) + 1 + f\_bias_7$ | $Z = (X - O) * M$<br>Basic multimodal<br>Rotated shifted<br>Non-separable scalable<br>No bounds for variables $x$<br>Initialize population in$[0, 600]^D$<br>Global optimum: $X^* = O$ is outside of the initialization<br>$F(X^*) = -180$ |

**Table 1** continued

| Num | Problems | Objection function | Property |
|---|---|---|---|
| Fun8 | Shifted rotated Ackleys function with Global optimum On bounds | $F_8(X) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} Z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi Z_i)\right) + 20 + e + f\_bias_8$ | $Z = (X - O) * M$ Basic multimodal Shifted, rotated Non-separable, scalable $X_i \in [-32, 32]^D$ Global optimum:$X^* = O$ $F(X^*) = -140$ |
| Fun9 | Shifted rastrigin | $F_9(X) = \sum_{i=1}^{D}(Z_i^2 - 10\cos(2\pi Z_i) + 10) + f\_bias_9$ | $Z = X - O$ Basic multimodal Shifted, separable Scalable $X_i \in [-5, 5]^D$ Global optimum:$X^* = O$ $F(X^*) = -330$ |

**Table 2** Test functions

| Num | Problems | Objection function | Property |
|---|---|---|---|
| Fun10 | Shifted rotated Rastrigin | $F_{10}(x) = \sum_{i=1}^{D}(Z_i^2 - 10\cos(2\pi Z_i) + 10) + f\_bias_{10}$ | $Z = (X - O) * M$ Basic multimodal Shifted, rotated Non-separable, scalable $X_i \in [-5, 5]^D$ Global optimum:$X^* = O$ $F(X^*) = -330$ |
| Fun11 | Shifted rotated Weierstrass | $F_{11}(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{kmax}[a^k\cos(2\pi b^k(Z_i + 0.5))]\right) - D\sum_{k=0}^{kmax}[a^k\cos(2\pi b^k \cdot 0.5)] + f\_bias_{11}$ | $Z = (X - O) * M$ Basic multimodal Shifted, rotated Non-separable, scalable $X_i \in [-0.5, 0.5]^D$ Global optimum:$X^* = O$ $F(X^*) = -90$ |
| Fun12 | Schwefels Problem 2.13 | $F_{12}(x) = \sum_{i=1}^{D}(A_i - B_i(x))^2 + f\_bias_{12}$ | Basic multimodal Shifted, non-separable Scalable $\chi_i \in [-\pi, \pi]^D$ Global optimum:$X^* = \alpha$ $F(X^*) = -460$ |
| Fun13 | Shifted expanded Griewanksplus Rosenbrocks Function(F8F2) | $F_{13}(x) = F8(F2(Z_1, Z_2)) + F8(F2(Z_2, Z_3)) + \cdots + F8(F2(Z_D, Z_1)) + f\_bias_{13}$ | $Z = X - O + 1$ Basic multimodal $X = [x_1, x_2, \ldots, x_D]$ Multimodal, Shifted Non-separable, scalable $X_i \in [-3, 1]^D$ Global optimum:$X^* = O$ $F(X^*) = -130$ |
| Fun14 | Shifted rotated Expanded Scaffer's F6 Function | $F_{14}(X) = EF(Z_1, Z_2, \ldots, Z_D) = F(Z_1, Z_2) + F(Z_2, Z_3) + \cdots + F(Z_{D-1}, Z_D) + F(Z_D, Z_1) + f\_bias_{14}$ | $Z = (X - O) * M$ Basic multimodal $X = [x_1, x_2, \ldots, x_D]$ Multimodal, shifted Non-separable, scalable $X_i \in [-100, 100]^D$ Global optimum:$X^* = O$ $F(X^*) = -300$ |

**Table 2** continued

| Num | Problems | Objection function | Property |
|---|---|---|---|
| Fun15 | Hybrid Composition | $f_{1-2}(x)$ : Rastrigin′sFunction<br>$f_{3-4}(x)$ : Weierstrassfunction<br>$f_{5-6}(x)$ : Griewank′sfunction<br>$f_{7-8}(x)$ : Ackley′sfunction<br>$f_{9-10}(x)$ : Spherefunction | Composition, multimodal<br>Separable nearthe<br>global optimum<br>(Rastrigin), scalable<br>A huge number of local optima<br>Different functions properties<br>are mixed together<br>Sphere functions give two<br>flat areas for the function<br>$X_i \in [-5, 5]^D$<br>$F(X^*) = 120$ |
| Fun16 | Rotated version of Hybrid composition Function F15 | Except Mi are different linear transformation matrixes with condition number of 2 all other settings are the same as F15. | Composition, multimodal<br>Rotated, non-separable<br>Scalable<br>A huge number of local optima<br>Different functions<br>Properties are mixed together<br>Sphere functions give<br>Two flat areas for<br>The function<br>$X_i \in [-5, 5]^D$<br>$F(X^*) = 120$ |
| Fun17 | F16 with Noise in Fitness | $F_{17}(X) = G(x) * (1 + 0.2\|N(0, 1)\|) + f\_bias_{17}$ | Composition, all settings<br>are the same as F16. Multimodal<br>Rotated, non-separable, scalable<br>A huge number of local optima<br>Different functions properties<br>are mixed together, sphere<br>Functions give two flat<br>areas for the function, with<br>Gaussian noise in fitness<br>$X_i \in [-5, 5]^D$<br>$F_{17}(X^*) = f\_bias_{17}(17) = 120$ |

## 3 Self-adaptive artificial bee colony algorithm

### 3.1 Population initialization

Population initialization is an essential phase, as it influences the convergence of the final solution. To reduce the deviation of different initial populations, this paper proposes an initial fixed population. In the first iteration of the algorithm, we use Eq. (7) to generate a population. This population is saved for use in later operations.

$$population = X_{\min} + rand * (X_{\max} - X_{\min}) \tag{7}$$

where $X_{\min}$ and $X_{\max}$ are the upper and lower bounds of the search space, respectively, and $rand$ is a random number in the range $(-1, 1)$.

Inspired by Ref. Gao and Liu (2012), we also designed a novel initialization algorithm that uses chaotic systems (Alatas 2010) and opposition-based learning method (Rahnamayan et al. 2008). Different from the method proposed by Gao and Liu (2012), cosine function is replaced by sine function and the range of values become a half of original values in the novel algorithm. This is because it can avoid repeated initialization and get better results. As shown in Algorithm 1, the detail steps are as follows. First, the chaotic method is used to generate the initial population. Next, another population is produced by the opposition-based learning method. The size of the two populations is the same. Then, we evaluate them and figure out the fitness values. We compare them and the better one is reserved. Finally, the final initial population is obtained.

### 3.2 Self-adaptive mechanism

During different stages of evolution, different CSGSs may be more effective. We develop the SABC-GB algorithm by introducing a self-adaptive mechanism into the ABC algorithm. We maintain a candidate strategy pool that includes several effective CSGSs with diverse characteristics. These CSGSs are used adaptively according to their previous performance in generating promising solutions. The core idea of the self-adaptive mechanism is as follows: During the evolu-

**Table 3** Test functions

| Num | Problems | Objection function | Property |
|---|---|---|---|
| Fun18 | Rotated Hybrid Composition | $f_{1-2}(x) : Ackley's$Function<br>$f_{3-4}(x) : Rastrigin's$Function<br>$f_{5-6}(x) : Sphere$Function<br>$f_{7-8}(x) : Weierstrass$Function<br>$f_{9-10}(x) : Griewank's$Function | Composition, Multimodal<br>Rotated, non-separable, scalable<br>A huge number of local optima<br>Different functions properties are mixed together, Sphere Functions give two flat areas for the function, A local optimum is set on the origin<br>$X_i \in [-5, 5]^D$<br>$F(X^*) = 10$ |
| Fun19 | Rotated Hybrid Composition Function with narrow basin global optimum | All settings are the same as F18 except $\sigma$ and $\lambda$ | Composition<br>Multimodal<br>Non-separable<br>Scalable<br>$X_i \in [-5, 5]^D$<br>$X^* = O_1$<br>$F(X^*) = 10$ |
| Fun20 | Rotated Hybrid Composition Function with Global Optimum on the Bounds | All settings are the same as F18 except after load the data file | Composition<br>Multimodal<br>Non-separable<br>Scalable<br>$X_i \in [-5, 5]^D$<br>$X^* = O_1$<br>$F(X^*) = 10$ |
| Fun21 | Rotated Hybrid Composition Function | $f_{1-2}(x) : RotatedExpanded$Scaffer'sF6Function<br>$f_{3-4}(x) : Rastrigin's$Function<br>$f_{5-6}(x) : F8F2$Function<br>$f_{7-8}(x) : Weierstrass$Function<br>$f_{9-10}(x) : Griewank's$Function | Composition, Multimodal<br>Rotated, Non-separable<br>Scalable<br>$X_i \in [-5, 5]^D$<br>$X^* = O_1$<br>$F(X^*) = 360$ |
| Fun22 | Rotated Hybrid Composition Function with High Condition Number Matrix | All settings are the same as F21 except $M_i$ | Composition , Multimodal<br>Non-separable<br>Scalable<br>$X_i \in [-5, 5]^D$<br>$X^* = O_1$<br>$F(X^*) = 360$ |
| Fun23 | Non-Continuous Rotated Hybrid Composition Function | All settings are the same as F21 except $x_j$ and round | Composition ,Multimodal<br>Non-separable<br>Scalable<br>$X_i \in [-5, 5]^D$<br>$X^* = O_1$<br>$F(X^*) = 360$ |
| Fun24 | Rotated Hybrid Composition Function | $f_1(x) : Weierstrass$Function<br>$f_2(x) :$ RotatedExpandedScaffer$'s$F6Function<br>$f_3(x) :$ F8F2Function, $f_4(x) :$ Ackley$'s$Function<br>$f_5(x) :$ Rastrigin$'s$Function, $f_6(x) :$ Griewank$'s$Function<br>$f_7(x) :$ Non $-$ ContinuousExpandedScaffer$'s$F6Function<br>$[f_8(x) :$ Non $-$ ContinuousRastrigin$'s$Function<br>$f_9(x) :$ HighConditionedEllipticFunction<br>$f_{10}(x) :$ SphereFunctionwithNoiseinFitness | Composition, Multimodal<br>Rotated, Non-separable<br>Scalable<br>$X_i \in [-5, 5]^D$<br>$X^* = O_1$<br>$F(X^*) = 260$ |
| Fun25 | Rotated Hybrid Composition Function without bounds | All settings are the same as F24 except no exact search range set for this test function | Composition, Multimodal<br>Non-separable<br>Scalable<br>$X_i \in [2, 5]^D$<br>$X^* = O_1$<br>$F(X^*) = 260$ |

**Table 4** Statistical results obtained by SABC-GB algorithm on 25 independent runs with 30-D

| Functions | Optimum | Best | Worst | Mean | Std |
|---|---|---|---|---|---|
| Fun1 | 5.68E−14 | 5.68E−14 | 5.68E−14 | 5.68E−14 | 0.00E+00 |
| Fun2 | 9.37E+03 | 9.37E+03 | 1.09E+04 | 1.01E+04 | 6.19E+02 |
| Fun3 | 1.04E+07 | 1.04E+07 | 1.11E+07 | 1.08E+07 | 3.23E+05 |
| Fun4 | 3.13E+04 | 3.13E+04 | 3.24E+04 | 3.19E+04 | 4.67E+02 |
| Fun5 | 6.49E+03 | 6.49E+03 | 8.43E+03 | 7.41E+03 | 7.94E+02 |
| Fun6 | 1.31E−01 | 1.31E−01 | 3.83E−01 | 2.52E−01 | 1.03E−01 |
| Fun7 | 4.70E+03 | 4.70E+03 | 4.70E+03 | 4.70E+03 | 0.00E+00 |
| Fun8 | 2.08E+01 | 2.08E+01 | 2.09E+01 | 2.09E+01 | 5.40E−02 |
| Fun9 | 5.68E−14 | 5.68E−14 | 5.68E−14 | 5.68E−14 | 0.00E+00 |
| Fun10 | 1.27E+02 | 1.27E+02 | 1.38E+02 | 1.34E+02 | 5.34E+00 |
| Fun11 | 2.47E+01 | 2.47E+01 | 2.52E+01 | 2.50E+01 | 2.66E−01 |
| Fun12 | 8.60E+03 | 8.60E+03 | 1.34E+04 | 1.05E+04 | 2.05E+03 |
| Fun13 | 4.64E−01 | 4.64E−01 | 8.69E−01 | 7.39E−01 | 1.94E−01 |
| Fun14 | 1.26E+01 | 1.26E+01 | 1.27E+01 | 1.27E+01 | 4.65E−02 |
| Fun15 | 2.74E−05 | 2.74E−05 | 4.43E−03 | 2.39E−03 | 1.81E−03 |
| Fun16 | 1.23E+02 | 1.23E+02 | 1.76E+02 | 1.58E+02 | 2.51E+01 |
| Fun17 | 2.31E+02 | 2.31E+02 | 2.39E+02 | 2.35E+02 | 3.37E+00 |
| Fun18 | 9.08E+02 | 9.08E+02 | 9.11E+02 | 9.09E+02 | 1.32E+00 |
| Fun19 | 9.08E+02 | 9.08E+02 | 9.11E+02 | 9.10E+02 | 1.13E+00 |
| Fun20 | 9.08E+02 | 9.08E+02 | 9.11E+02 | 9.09E+02 | 1.47E+00 |
| Fun21 | 4.93E+02 | 4.93E+02 | 5.05E+02 | 5.00E+02 | 0.00E+00 |
| Fun22 | 9.35E+02 | 9.35E+02 | 9.41E+02 | 9.38E+02 | 2.68E+00 |
| Fun23 | 5.34E+02 | 5.34E+02 | 5.34E+02 | 5.34E+02 | 0.00E+00 |
| Fun24 | 2.00E+02 | 2.00E+02 | 9.62E+02 | 4.54E+02 | 3.59E+02 |
| Fun25 | 1.64E+03 | 1.64E+03 | 1.64E+03 | 1.64E+03 | 0.00E+00 |

Mean and std denote the average and standard deviation of the corresponding function values obtained in 25 runs

tion process, each CSGS is assigned a probability value. Each CSGS is selected according to the probability for each solution through roulette wheel selection. The new individuals are then generated by the selected CSGS.

In the initialization stage, each CSGS is given an equal selection probability. The selection probability is the reciprocal of the number of CSGSs. Flag matrices for successful and failed evolutions are denoted as $nsFlagSABC$ and $nfFlagSABC$, respectively. $LP$ is defined as the fixed number of previous generations. The total success and total failure flag matrices of each CSGS in $LP$ are termed $S_{kg}$ and $F_{kg}$, respectively. These two matrices are initially null.

First, for each individual, one strategy is selected from the CSGS pool through roulette wheel selection. Next, with the selected CSGS, a new individual is generated and its fitness value is calculated. If the new fitness value is better than the previous one, the matrix $nsFlagSABC$ is updated. Otherwise, the matrix $nfFlagSABC$ is updated. After the evolution of all individuals, statistical information about the flag matrices $nsFlagSABC$ and $nfFlagSABC$ is recorded in $S_{kg}$ and $F_{kg}$. Once the iteration number reaches $LP$, new selection probabilities for all strategies, $P'_q$, are calculated through $S_{kg}$ and $F_{kg}$. This process is described in Eq. (8), and the probabilities are normalized according to Eq. (9).

**Algorithm 1** : COL initialization algorithm

Step1) Initialization: iteration Iter = 300, population size PS = 30 or 50, and dimension D = 60 or 100
Step2) For i = 1 to PS
  For j = 1 to D
  Randomly generate a variable R ∈ (0, 0.5)
  R = COS(ΠR)
  End
  $P_{i,j} = X_{\min,j} R(X_{\max,j} - X_{\min,j})$
  End
End
Step3) For i = 1 to PS
  For j = 1 to D
  $Q_{i,j} = X_{\min,j} + X_{\max,j} - P_{i,j}$
  End
End
Step4) Selecting SN fittest individuals from set the$\{P(SN) \cup Q(SN)\}$as initial population.

**Table 5** Statistical results obtained by SABC-GB algorithm on 25 independent runs with 50-D

| Functions | Optimum | Best | Worst | Mean | Std |
|---|---|---|---|---|---|
| Fun1 | 1.71E−13 | 1.71E−13 | 2.84E−13 | 2.27E−13 | 4.64E−14 |
| Fun2 | 5.67E+04 | 5.67E+04 | 6.55E+04 | 6.02E+04 | 3.79E+03 |
| Fun3 | 4.45E+07 | 4.45E+07 | 6.70E+07 | 5.85E+07 | 9.96E+06 |
| Fun4 | 1.25E+05 | 1.25E+05 | 1.38E+05 | 1.30E+05 | 6.23E+03 |
| Fun5 | 2.20E+04 | 2.20E+04 | 2.48E+04 | 2.37E+04 | 1.26E+03 |
| Fun6 | 1.18E+00 | 1.18E+00 | 5.74E+00 | 3.77E+00 | 1.91E+00 |
| Fun7 | 6.20E+03 | 6.20E+03 | 6.20E+03 | 6.20E+03 | 0.00E+00 |
| Fun8 | 2.10E+01 | 2.10E+01 | 2.10E+01 | 2.10E+01 | 0.00E+00 |
| Fun9 | 1.71E−13 | 1.71E−13 | 1.71E−13 | 1.71E−13 | 0.00E+00 |
| Fun10 | 3.70E+02 | 3.70E+02 | 4.64E+02 | 4.24E+02 | 3.92E+01 |
| Fun11 | 5.38E+01 | 5.38E+01 | 5.77E+01 | 5.60E+01 | 1.61E+00 |
| Fun12 | 4.56E+04 | 4.56E+04 | 7.55E+04 | 5.78E+04 | 1.28E+04 |
| Fun13 | 1.23E+00 | 1.23E+00 | 1.85E+00 | 1.58E+00 | 2.61E−01 |
| Fun14 | 2.23E+01 | 2.23E+01 | 2.28E+01 | 2.26E+01 | 1.98E−01 |
| Fun15 | 1.24E+01 | 1.24E+01 | 3.58E+01 | 2.27E+01 | 9.77E+00 |
| Fun16 | 3.21E+02 | 3.21E+02 | 3.47E+02 | 3.10E+02 | 3.58E+01 |
| Fun17 | 4.07E+02 | 4.07E+02 | 4.52E+02 | 4.33E+02 | 1.93E+01 |
| Fun18 | 9.35E+02 | 9.35E+02 | 9.44E+02 | 9.39E+02 | 4.18E+00 |
| Fun19 | 9.29E+02 | 9.29E+02 | 9.45E+02 | 9.37E+02 | 6.34E+00 |
| Fun20 | 9.32E+02 | 9.32E+02 | 9.40E+02 | 9.35E+02 | 3.88E+00 |
| Fun21 | 1.02E+03 | 1.02E+03 | 1.02E+03 | 1.02E+03 | 5.59E−01 |
| Fun22 | 9.46E+02 | 9.46E+02 | 1.03E+03 | 9.91E+02 | 3.46E+01 |
| Fun23 | 1.02E+03 | 1.02E+03 | 1.02E+03 | 1.02E+03 | 0.00E+00 |
| Fun24 | 1.06E+03 | 1.06E+03 | 1.09E+03 | 1.08E+03 | 1.37E+01 |
| Fun25 | 1.68E+03 | 1.68E+03 | 1.68E+03 | 1.69E+03 | 5.95E+00 |

Mean and std denote the average and standard deviation of the corresponding function values obtained in 25 runs

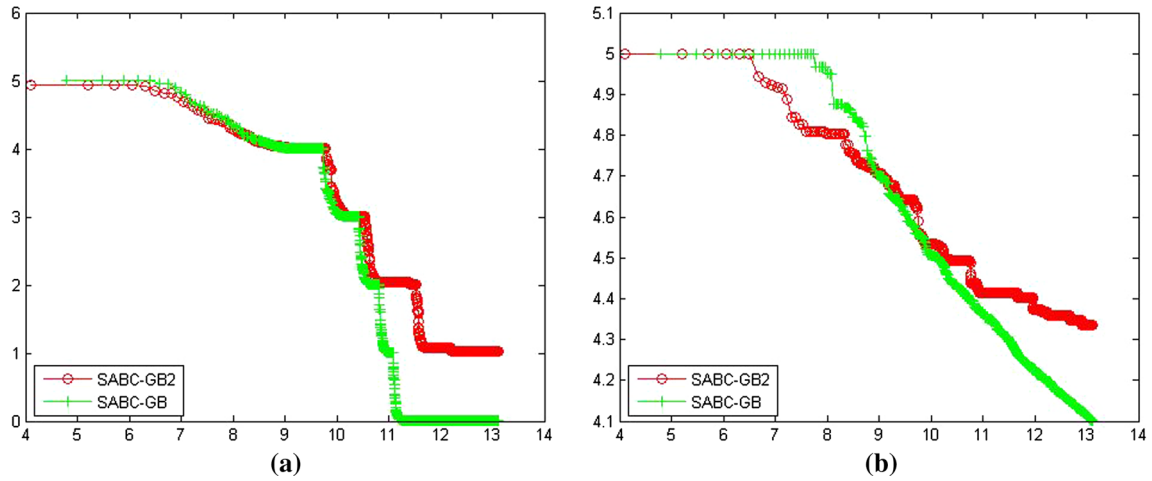


**Fig. 1** Convergence characteristics of SABC-GB and SABC-GB2 on fun1 and fun2. *Note*: because we do not know the range of the possible fitness value ($y$-axis) in advance, in this paper, we suppose the fitness value range is big enough, in order to make the figure seems clearly, we convert the fitness to the correspond fitness

**Table 6** Optimization results of GABC, ABC/best/1, ABC/best/2 and SABC-GB on 25 test functions with 30-D

| Functions | GABC | | ABC/best/1 | | ABC/best/2 | | SABC-GB | |
|---|---|---|---|---|---|---|---|---|
| | Mean | Std | Mean | Std | Mean | Std | Mean | Std |
| Fun1 | 1.09E−02 | 6.33E−03 | 1.97E−12 | 1.93E−13 | 3.00E+00 | 1.12E−01 | **5.68E−14** | **0.00E+00** |
| Fun2 | 2.99E+04 | 6.02E+03 | 3.55E+04 | 1.96E+03 | 4.04E+04 | 1.09E+03 | **1.01E+04** | **6.19E+02** |
| Fun3 | 4.88E+07 | 7.10E+06 | 5.28E+07 | 9.41E+06 | 9.14E+07 | 9.41E+06 | **1.08E+07** | **3.23E+05** |
| Fun4 | 4.60E+04 | 1.95E+03 | 5.06E+04 | 5.02E+03 | 4.74E+04 | 3.49E+03 | **3.19E+04** | **4.67E+02** |
| Fun5 | **1.09E+04** | **2.90E+02** | 9.34E+03 | 5.34E+02 | **1.09E+04** | 6.53E+02 | 7.41E+03 | 7.94E+02 |
| Fun6 | 1.44E+03 | 7.37E+02 | 1.86E+02 | 4.11E+01 | 4.74E+04 | 4.06E+04 | **2.52E−01** | **1.03E−01** |
| Fun7 | 4.70E+03 | 8.97E−01 | 4.70E+03 | 5.15E−02 | 4.70E+03 | 4.89E−01 | 4.70E+03 | **0.00E+00** |
| Fun8 | 2.09E+01 | 9.03E−02 | 2.09E+01 | 1.10E−02 | 2.09E+01 | **2.28E−03** | 2.09E+01 | 5.40E−02 |
| Fun9 | 3.64E+01 | 4.88E+00 | 2.03E−12 | 2.02E−12 | 4.69E+01 | 3.96E+00 | **5.68E−14** | **0.00E+00** |
| Fun10 | 3.11E+02 | 9.17E+00 | 2.82E+02 | 1.14E+01 | 3.08E+02 | 1.21E+01 | **1.34E+02** | **5.34E+00** |
| Fun11 | 3.36E+01 | 3.98E−01 | 3.32E+01 | 1.45E+00 | 3.46E+01 | 4.65E−01 | **2.50E+01** | **2.66E−01** |
| Fun12 | 8.10E+04 | 2.49E+04 | 8.82E+04 | 1.61E+04 | 1.64E+05 | 2.07E+04 | **1.05E+04** | **2.05E+03** |
| Fun13 | 7.58E+00 | 4.75E−01 | 7.59E+00 | 7.45E−01 | 1.03E+01 | 5.58E−01 | **7.39E−01** | **1.94E−01** |
| Fun14 | 1.32E+01 | 1.04E−01 | 1.33E+01 | **3.39E−02** | 1.33E+01 | 3.55E−02 | **1.27E+01** | 4.65E−02 |
| Fun15 | 3.36E+02 | 1.19E+02 | 3.29E+02 | 1.45E+02 | 4.87E+02 | 2.94E+00 | **2.39E−03** | **1.81E−03** |
| Fun16 | 3.66E+02 | 4.10E+01 | 3.17E+02 | **1.11E+01** | 3.72E+02 | 2.98E+01 | **1.58E+02** | 2.51E+01 |
| Fun17 | 4.65E+02 | 2.36E+01 | 4.25E+02 | 8.83E+00 | 4.33E+02 | 6.07E+00 | **2.35E+02** | **3.37E+00** |
| Fun18 | 9.24E+02 | 2.32E+00 | 9.21E+02 | 2.16E+00 | 9.29E+02 | 1.93E+00 | **9.09E+02** | **1.32E+00** |
| Fun19 | 9.32E+02 | 1.76E+00 | 9.21E+02 | 1.73E+00 | 9.25E+02 | 4.28E+00 | **9.10E+02** | **1.13E+00** |
| Fun20 | 9.27E+02 | **9.21E−01** | 9.23E+02 | 3.07E+00 | 9.32E+02 | 1.86E+00 | **9.09E+02** | 1.47E+00 |
| Fun21 | 5.14E+02 | 1.62E+00 | 5.07E+02 | 5.87E−01 | 7.49E+02 | 3.81E+01 | **5.00E+02** | **0.00E+00** |
| Fun22 | 1.07E+03 | 1.04E+01 | 1.04E+03 | 1.60E+01 | 1.05E+03 | 1.19E+01 | **9.38E+02** | **2.68E+00** |
| Fun23 | 6.99E+02 | 1.49E+00 | 6.12E+02 | 2.13E+01 | 8.78E+02 | 3.53E+01 | **5.34E+02** | **0.00E+00** |
| Fun24 | 1.18E+03 | 1.79E+01 | 1.07E+03 | 3.25E+01 | 1.10E+03 | **8.10E+00** | **4.54E+02** | 3.59E+02 |
| Fun25 | 1.71E+03 | 3.65E+00 | 1.68E+03 | 5.07E+00 | 1.68E+03 | 1.54E+01 | **1.64E+03** | **0.00E+00** |

Mean and std denote the average and standard deviation of the corresponding function values obtained in 25 runs, and the best results in terms of mean values are in bold

$$P'_q = \begin{cases} \sum_{k=1}^{LP} S_{\mathrm{kg}} / \left( \sum_{k=1}^{LP} S_{kg} + \sum_{k=1}^{LP} F_{kg} \right), & \sum_{k=1}^{LP} S_{kg} \neq 0 \\ \sum_{k=1}^{LP} S_{kg} / \left( \varepsilon + \sum_{k=1}^{LP} F_{kg} \right), & \text{Otherwise} \end{cases} \tag{8}$$

$$P_q = P'_q / \sum_{q=1}^{Q} P'_q \tag{9}$$

where LP is a fixed integer. In this paper, $LP$ is set to 10. $P_q$ represents the success rate of candidate solutions generated by one strategy successfully entering the next generation within the previous LP generations. The small constant value $\varepsilon$ is used to avoid division by zero.

### 3.3 Self-adaptive candidate strategies

We now investigate several effective CSGSs from the relevant literature and choose from among them to construct the strategy candidate pool. Theoretical studies of the optimal pool size and the selection of strategies used in the pool are attractive research issues and deserve further investigation.

The best solutions in the current population are very useful sources that can be used to improve the convergence performance. Gao proposed the ABC/best/1 and ABC/best/2 strategies (Gao et al. 2012), whereby the best solutions which have been explored are used to direct the movement of the current population. The corresponding strategies are devised as follows:

$$V_{i,j} = X_{best,j} + \phi_{i,j}(X_{r1,j} - X_{r2,j}) \tag{10}$$

$$V_{i,j} = X_{best,j} + \phi_{i,j}(X_{r1,j} - X_{r2,j}) + \phi_{i,j}(X_{r3,j} - X_{r4,j}) \tag{11}$$

where the indices $i$, $r_1$, $r_2$, $r_3$, and $r_4$ are different integers chosen at random from $\{1, 2, \ldots, SN\}$. $X_{best}$ is the individual vector with the best fitness in the current population, and $j$ is a random integer chosen from $\{1, 2, \ldots, D\}$.$\phi_{i,j}$ is a random number in the range $[-1, 1]$.

**Table 7** Optimization results of GABC, ABC/best/1, ABC/best/2 and SABC-GB on 25 test functions with 50-D

| Functions | GABC | | ABC/best/1 | | ABC/best/2 | | SABC-GB | |
|---|---|---|---|---|---|---|---|---|
| | Mean | Std | Mean | Std | Mean | Std | Mean | Std |
| Fun1 | 1.08E+00 | 7.42E−01 | 6.33E−11 | 1.10E−11 | 5.77E+02 | 1.09E+02 | **2.27E–13** | **4.64E–14** |
| Fun2 | 9.92E+04 | **3.74E+03** | 1.06E+05 | 1.22E+04 | 1.13E+05 | 3.76E+03 | **6.02E+04** | 3.79E+03 |
| Fun3 | 2.20E+08 | 1.68E+07 | 1.84E+08 | 4.77E+07 | 2.81E+08 | 1.64E+07 | **5.85E+07** | **9.96E+06** |
| Fun4 | 1.48E+05 | 1.68E+04 | 1.31E+05 | 7.70E+03 | 1.44E+05 | 7.06E+03 | **1.30E+05** | **6.23E+03** |
| Fun5 | 2.64E+04 | **6.50E+02** | 2.47E+04 | 1.20E+03 | 2.82E+04 | 1.17E+03 | **2.37E+04** | 1.26E+03 |
| Fun6 | 2.19E+04 | 2.09E+03 | 3.24E+03 | 1.33E+03 | 3.53E+06 | 1.84E+06 | **3.77E+00** | **1.91E+00** |
| Fun7 | 6.24E+03 | 1.39E+01 | 6.22E+03 | 1.85E+00 | 6.23E+03 | 1.19E+01 | **6.20E+03** | **0.00E+00** |
| Fun8 | 2.11E+01 | **9.00E-03** | 2.11E+01 | 1.82E−02 | 2.11E+01 | 4.27E−02 | **2.10E+01** | 0.00E+00 |
| Fun9 | 1.10E+02 | 9.04E+00 | 2.97E−12 | 6.18E−13 | 1.58E+02 | 7.00E+00 | **1.71E–13** | **0.00E+00** |
| Fun10 | 8.28E+02 | **1.14E+01** | 6.61E+02 | 1.17E+01 | 7.68E+02 | 2.39E+01 | **4.24E+02** | 3.92E+01 |
| Fun11 | 6.39E+01 | **6.58E–01** | 6.46E+01 | 6.63E−01 | 6.61E+01 | 6.76E−01 | **5.60E+01** | 1.61E+00 |
| Fun12 | 4.65E+05 | 8.09E+04 | 4.69E+05 | **4.79E+03** | 6.87E+05 | 1.19E+05 | **5.78E+04** | 1.28E+04 |
| Fun13 | 2.12E+01 | 1.75E+00 | 1.82E+01 | 1.90E+00 | 2.86E+01 | 1.11E+00 | **1.58E+00** | **2.61E–01** |
| Fun14 | 2.30E+01 | 2.00E−01 | 2.31E+01 | 2.34E−01 | 2.32E+01 | **7.20E–02** | **2.26E+01** | 1.98E−01 |
| Fun15 | 3.93E+02 | 6.88E+01 | 3.17E+02 | 8.96E+01 | 4.74E+02 | **3.22E+00** | **2.27E+01** | 9.77E+00 |
| Fun16 | 4.26E+02 | **3.91E–01** | 4.27E+02 | 1.37E+00 | 4.58E+02 | 4.60E+00 | **3.10E+02** | 3.58E+01 |
| Fun17 | 8.52E+02 | 3.51E+01 | 7.56E+02 | 3.98E+01 | 7.75E+02 | **1.26E+01** | **4.33E+02** | 1.93E+01 |
| Fun18 | 9.89E+02 | 2.07E+01 | 9.71E+02 | 9.33E+00 | 1.00E+03 | 8.80E+00 | **9.39E+02** | **4.18E+00** |
| Fun19 | 9.91E+02 | 8.55E+00 | 9.68E+02 | **1.30E+00** | 1.00E+03 | 1.04E+01 | **9.37E+02** | 6.34E+00 |
| Fun20 | 9.91E+02 | 7.74E+00 | 9.75E+02 | **3.36E+00** | 9.94E+02 | 6.69E+00 | **9.35E+02** | 3.88E+00 |
| Fun21 | 1.04E+03 | 1.86E+00 | 1.04E+03 | 1.16E+00 | 1.04E+03 | 2.88E+00 | **1.02E+03** | **5.59E–01** |
| Fun22 | 1.20E+03 | **5.75E+00** | 1.13E+03 | 2.46E+01 | 1.15E+03 | 1.92E+01 | **9.91E+02** | 3.46E+01 |
| Fun23 | 1.06E+03 | 3.30E+00 | 1.04E+03 | 3.69E+00 | 1.05E+03 | 3.43E+00 | **1.02E+03** | **0.00E+00** |
| Fun24 | 1.40E+03 | 1.46E+01 | 1.36E+03 | **5.29E+00** | 1.38E+03 | 1.36E+01 | 1.08E+03 | 1.37E+01 |
| Fun25 | 1.90E+03 | **4.63E+00** | 1.85E+03 | 9.93E+00 | 1.85E+03 | 1.59E+01 | 1.69E+03 | 5.95E+00 |

Mean and std denote the average and standard deviation of the corresponding function values obtained in 25 runs, and the best results in terms of mean values are in bold

Besides, Zhu proposed the GABC algorithm (Zhu and Kwong 2010), which can be written as follows:

$$V_{i,j} = X_{\mathrm{i},j} + \phi_{i,j}(X_{i,j} - X_{k,j}) + \varphi_{i,j}(x_{best,j} - x_{i,j}) \tag{12}$$

where $X_{k,j}$ is a random individual in the population and $X_{best,j}$ is the individual with the best fitness in the current population. The indices $i, r_1, r_2, r_3$, and $r_4$ are mutually exclusive integers chosen at random from $\{1, 2, \ldots, SN\}$, and $j$ is a random integer chosen from $\{1, 2, \ldots, D\}$. $\phi_{i,j}$ is a random number in the range $[-1, 1]$, and $\phi_{i,j}$ is a uniform random number in the range $[0, C]$, where $C$ is a nonnegative constant. In this paper, $C$ is set to 1.5.

We use GABC, ABC/best/1, and ABC/best/2 as the CSGSs of the SABC-GB algorithm. For a certain optimization problem, the selection probability of a good CSGS should be higher than that of the others. As the generation number increases, the probabilities will evolve until the best CSGS for the problem has been determined. A description of this framework is present in Algorithm 2.

## 4 Experiments

### 4.1 Benchmark functions and parameter settings

We employed 25 benchmark functions to test the performance of the SABC-GB algorithm. A detailed description of these test functions can be found in Suganthan et al. (2005). The functions are numbered from $f_1$ to $f_{25}$. The specific functions are listed in Tables 1, 2 and 3.

The number of food sources is equal to the number of employed or onlooker bees ($SN$). In the experiments, we considered solution dimensions of 30 and 50. To ensure a fair comparison, the populations for these dimensions were initialized using the same random seeds, and the number of

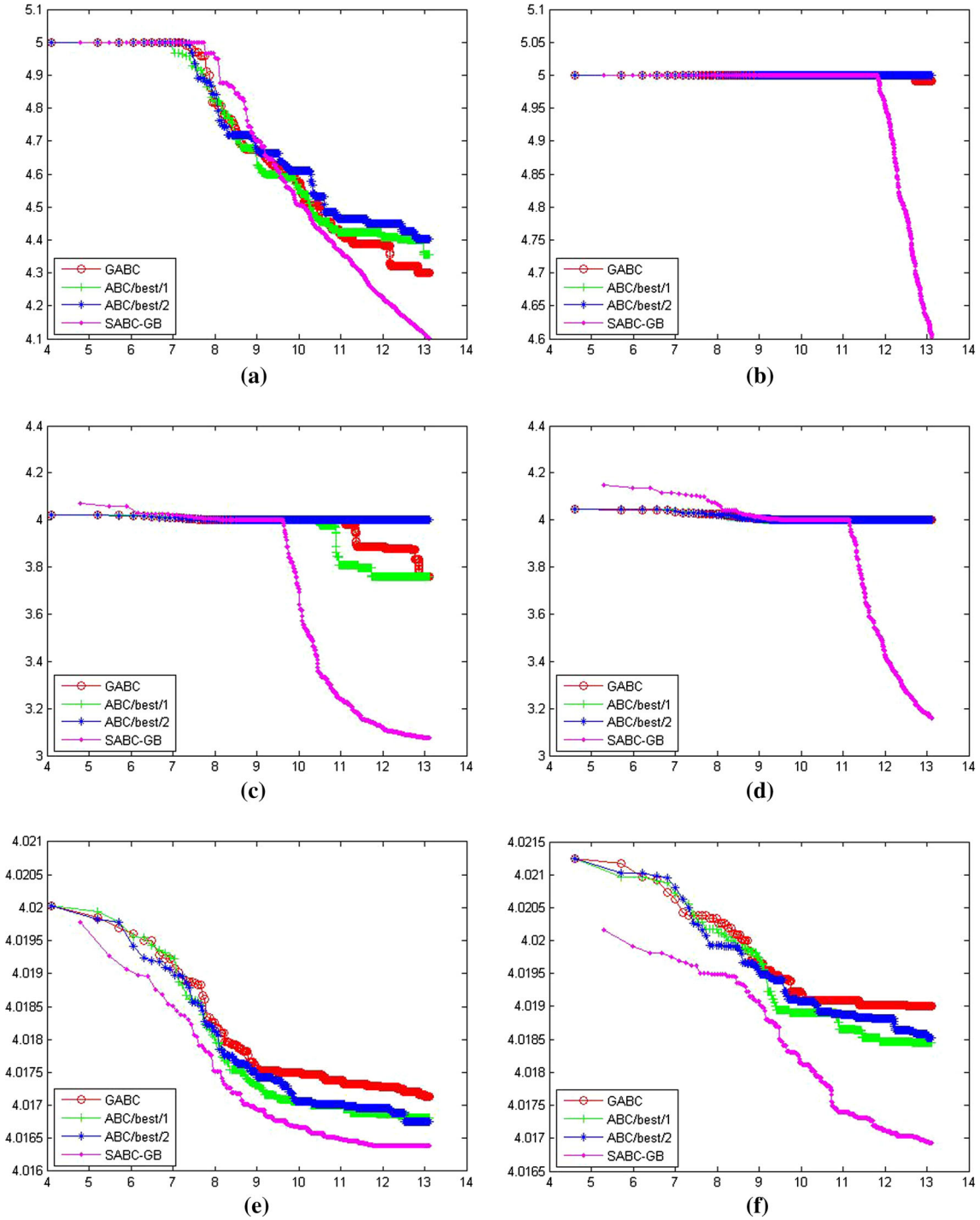


**Fig. 2** Convergence performance of SABC-GB, GABC, ABC/best/1 and ABC/best/2. **a** Fun2 with $D=30$. **b** Fun2 with $D=50$. **c** (c) Fun13 with $D=30$. **d** Fun13 with $D=50$

employed bees was set to half the population size, respectively. The maximum number of cycles for the algorithm is related to the dimension $D$ and number of individuals $ps$, $\text{limit} = 0.6 * (ps/2) * D$. The fixed number of previous generations $LP = 10$. The number of decision variables was set to the same for all 25 test functions. For each algorithm on each function, 25 independent runs were conducted with 500,000 number of function evaluations (FEs) as the termination criterion.

**Algorithm 2** : SABC-GB algorithm

Step1) Initialization
  Step1.1) Use Algorithm 1 to produce the initial population P ($P = \{x_1, x_2, \ldots, x_{SN}\}$)
  Step1.2) Initialize strategy selection probability $P_q(q = 1, 2, \ldots, Q)$, Q is the number of available strategies, $LP$, $nsFlagSABC$, $nfFlagSABC$, $Skg$ and $Fkg$
  Step1.3) Calculate the fitness values of $P$
Step2) Employed Bee Phase
  Step2.1) For (fitCount<MaxFES fitCounti++), SN,CONTINUE
    Step2.1.1) The current strategy can be selected by selection algorithm, such as roulette wheel method
    Step2.1.2) Using the current strategy from the candidate pool to generate a new solution and calculate fitness value of the new solution
  Step2.2) If find the better value, update the solution, else $trial = trial + 1$
  Step2.3) Update the memory flag matrixes $nsFlagSABC$ and $nfFlagSABC$
Step3) Calculate probability values
Step4) Onlooker Bee Phase
  Step4.1) Compare probability values
    Step4.1.1) Using the current strategy from the candidate pool to generate a new solution and calculate fitness value of the new solution
    Step4.1.2) If find the better value, update the solution, else trial=trial+1
    Step4.1.3) Update the memory flag matrixes $nsFlagSABC$ and $nfFlagSABC$
  Step4.2) If circulation achieve LP, update $Skg$ and $Fkg$
  End for
Step5) Scout Bee Phase
  If max(trial)>limit, produce one individual randomly
Step6) If FES>= MaxFES, output the optimal solution, else go to step2

### 4.2 Experimental results

We compared the SABC-GB algorithm to GABC (Zhu and Kwong 2010), ABC/best (Gao et al. 2012), ABC1 (Gao et al. 2012), and ABC2 (Gao et al. 2012). The experimental results are present in the following tables. Through the experiment, we found that the performance of SABC-GB, which consists of three self-adaptive candidate strategies, is better than that of the other algorithms. This section examines the performance of each of the three strategies employed in SABC-GB in comparison with a number of previous algorithms. The statistical results of SABC-GB are listed in Tables 4 and 5, which present the optimum, best, worst, mean, and standard deviation of each benchmark function. As shown in Tables 4 and 5, SABC-GB found the optimum solution with both 30 and 50 dimensions. From Table 4, it is apparent that SABC-GB reached the same values for Fun1, Fun7, Fun9, Fun21, Fun23, and Fun25 in the 30-D case, and similarly for Fun7, Fun8, Fun9, and Fun23 in the 50-D case. SABC-GB can discover close results in most functions. Overall, the efficiency of SABC-GB became lower when the number of dimensions increased.

#### *4.2.1 Performance comparison between SABC-GB and SABC-GB2*

As all of the test functions are minimization problems, smaller fitness values correspond to better solutions. For example, Fig. 1 illustrates the convergence characteristics of Fun1 and Fun2 in the 25 independent runs using SABC-GB and SABC-GB2 (SABC-GB2 is SABC-GB without Algorithm 1). It can be seen that, in the later stages of evolution, SABC-GB2 became trapped earlier than SABC-GB. Moreover, the convergence rate of SABC-GB2 was slower. The primary reason is that the distribution of the initial population has a significant influence on the exploitation ability of this algorithm, and good exploration directly affects the convergence rate and quality of the solution, especially in the later phases of evolution. Obviously, the novel initialization approach employing chaotic systems and opposition-based learning reduces the effects of these deficiencies. We research the conclusion that initialization with Algorithm 1 is better than the random initialization for almost all of the test functions.

#### *4.2.2 Performance comparison between different strategies*

The performance of SABC-GB was compared with that of GABC (Zhu and Kwong 2010), ABC/best/1 (Gao et al. 2012), and ABC/best/2 (Gao et al. 2012). The results present in Tables 6 and 7 in terms of the mean and standard deviation of solutions are obtained in the 25 independent runs by each algorithm. Figure 2 presents a comparison in terms of the convergence characteristics of the evolutionary processes in solving the three different kinds of problems.

Interestingly, SABC-GB found the exact minimum of Fun1 and Fun9 in all dimensions. We can also observe that GABC and ABC/best/2 do not find the precise solutions. Hence, these kinds of problems are not easy, but can generally be solved by SABC-GB with a high degree of accuracy. The solutions and the convergence rates of the different algorithms are shown in Tables 6, 7 and Fig. 2. From these results, it is clear that SABC-GB finds the most accurate solutions for most test functions and has the fastest convergence rate. In particular, SABC-GB gives highly accurate results for most test functions except Fun5 with $D = 30$, Fun7 with $D = 30$, Fun24 with $D = 50$ and Fun25 with $D = 50$. Moreover, the experimental results demonstrate that the efficiency of the four algorithms is similar on Fun8. The standard deviation of the solutions given by SABC-GB is worse in the 50-D case than for 30-D. Nonetheless, SABC-GB outperforms GABC, ABC/best/1, and ABC/best-/2 significantly, with better mean values in all dimensions. In short,

**Table 8** Optimization results of SABC-GB, SACABC, ISABC, QAABC and SSABC on 25 test functions with 30-D

| Functions | SACABC | | ISABC | | QAABC | | SSABC | | SABC-GB | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Mean | Std | Mean | Std | Mean | Std | Mean | Std | Mean | Std |
| Fun1 | 1.33E−13 | 2.68E−14 | 2.40E+04 | 1.35E+03 | 1.14E−13 | 0.00E+00 | 1.67E+00 | 2.08E−01 | 5.68E−14 | 0.00E+00 |
| Fun2 | 2.50E+04 | 3.53E+03 | 3.80E+04 | 2.00E+03 | 1.80E+04 | 9.60E+02 | 1.04E+04 | 1.25E+03 | 1.01E+04 | 6.19E+02 |
| Fun3 | 3.71E+07 | 5.08E+06 | 5.16E+07 | 7.68E+06 | 1.72E+07 | 1.21E+06 | 1.10E+07 | 2.33E+06 | 1.08E+07 | 3.23E+05 |
| Fun4 | 4.56E+04 | 2.13E+03 | 6.47E+04 | 8.59E+03 | 4.06E+04 | 5.56E+03 | 2.68E+04 | 7.80E+03 | 3.19E+04 | 4.67E+02 |
| Fun5 | 6.74E+03 | 7.52E+02 | 1.70E+04 | 1.25E+03 | 8.91E+03 | 9.53E+02 | 6.77E+03 | 1.17E+03 | 7.41E+03 | 7.94E+02 |
| Fun6 | 1.06E+01 | 1.28E+00 | 2.46E+10 | 3.21E+09 | 2.27E+00 | 2.17E+00 | 8.24E+02 | 7.71E+01 | 2.52E−01 | 1.03E−01 |
| Fun7 | 4.70E+03 | 6.14E−05 | 5.28E+03 | 1.97E+02 | 3.07E+03 | 5.74E+01 | 4.70E+03 | 6.14E−05 | 4.70E+03 | 0.00E+00 |
| Fun8 | 2.10E+01 | 4.79E−02 | 2.09E+01 | 7.28E−02 | 2.09E+01 | 1.83E−02 | 2.08E+01 | 6.94E−02 | 2.09E+01 | 5.40E−02 |
| Fun9 | 1.14E−13 | 0.00E+00 | 1.77E+02 | 6.93E+00 | 1.52E−13 | 2.68E−14 | 1.17E+00 | 1.23E−01 | 5.68E−14 | 0.00E+00 |
| Fun10 | 2.30E+02 | 1.02E+01 | 5.77E+02 | 7.90E+01 | 1.97E+02 | 5.48E+00 | 2.43E+02 | 6.32E+00 | 1.34E+02 | 5.34E+00 |
| Fun11 | 3.24E+01 | 3.71E−01 | 2.95E+01 | 5.74E−01 | 2.62E+01 | 1.18E+00 | 2.65E+01 | 1.04E+00 | 2.50E+01 | 2.66E−01 |
| Fun12 | 2.84E+04 | 3.01E+03 | 1.90E+05 | 2.27E+04 | 1.58E+04 | 3.26E+03 | 1.75E+04 | 4.23E+03 | 1.05E+04 | 2.05E+03 |
| Fun13 | 2.08E+00 | 1.86E−01 | 1.69E+02 | 4.40E+01 | 1.33E+00 | 7.91E−02 | 1.46E+00 | 1.13E−01 | 7.39E−01 | 1.94E−01 |
| Fun14 | 1.35E+01 | 1.25E−01 | 1.33E+01 | 1.24E−01 | 1.28E+01 | 1.57E−01 | 1.27E+01 | 1.56E−01 | 1.27E+01 | 4.65E−02 |
| Fun15 | 6.69E+01 | 6.54E+00 | 4.92E+02 | 4.38E+01 | 3.30E+01 | 1.47E+00 | 1.85E+01 | 4.66E+00 | 2.39E−03 | 1.81E−03 |
| Fun16 | 2.44E+02 | 1.19E+01 | 4.71E+02 | 7.58E+01 | 2.07E+02 | 1.66E+01 | 2.23E+02 | 2.25E+01 | 1.58E+02 | 2.51E+01 |
| Fun17 | 3.61E+02 | 1.60E+01 | 5.67E+02 | 3.53E+01 | 3.27E+02 | 4.74E+01 | 2.74E+02 | 1.48E+01 | 2.35E+02 | 3.37E+00 |
| Fun18 | 9.12E+02 | 4.88E−01 | 9.92E+02 | 3.46E+01 | 8.80E+02 | 5.29E+01 | 8.82E+02 | 4.80E+01 | 9.09E+02 | 1.32E+00 |
| Fun19 | 9.12E+02 | 1.51E+00 | 9.87E+02 | 8.38E+00 | 9.19E+02 | 2.42E+00 | 8.79E+02 | 4.40E+01 | 9.10E+02 | 1.13E+00 |
| Fun20 | 9.14E+02 | 1.56E+00 | 9.92E+02 | 1.11E+01 | 9.18E+02 | 2.04E+00 | 9.92E+02 | 4.71E+01 | 9.09E+02 | 1.47E+00 |
| Fun21 | 5.00E+02 | 4.68E−13 | 1.23E+03 | 3.79E+01 | 5.00E+02 | 6.92E−11 | 5.00E+02 | 1.55E−01 | 5.00E+02 | 0.00E+00 |
| Fun22 | 9.97E+02 | 6.21E+00 | 1.18E+03 | 3.54E+01 | 1.02E+03 | 1.18E+01 | 1.03E+03 | 2.54E+01 | 9.38E+02 | 2.68E+00 |
| Fun23 | 5.34E+02 | 9.18E−04 | 1.17E+03 | 1.56E+01 | 5.34E+02 | 7.32E−01 | 5.34E+02 | 9.01E−04 | 5.34E+02 | 0.00E+00 |
| Fun24 | 2.61E+02 | 5.40E+00 | 1.16E+03 | 5.75E+01 | 2.10E+02 | 4.27E+00 | 2.07E+02 | 2.79E+00 | 4.54E+02 | 3.59E+02 |
| Fun25 | 1.65E+03 | 2.75E+00 | 1.76E+03 | 1.97E+01 | 1.63E+03 | 1.17E+01 | 1.64E+03 | 1.95E+00 | 1.64E+03 | 0.00E+00 |

Mean and std denote the average and standard deviation of the corresponding function values obtained in 25 runs

SABC-GB has better exploration and exploitation abilities.

It can also be seen from these results that the proposed algorithm finds better solutions for most functions. Figure 2 shows the change in fitness of SABC-GB, GABC, ABC/best/1, and ABC/best/2 for Fun2, Fun13, and Fun25, respectively. These functions are from different classes. Obviously, the convergence speed of SABC-GB is faster than that of the other algorithms. Although the optimization process is similar in the intermediate phase of some problems, SABC-GB eventually gives better solutions than the other algorithms. From these tables, we can conclude that the SABC-GB algorithm is better for global optimization problems than the other algorithms. The proposed SABC-GB avoids becoming trapped in local optima while improving the global search ability.

#### *4.2.3 Performance comparisons between the proposed algorithm with other self-adaptive algorithms*

To further verify the efficiency and superiority of SAB

C-GB, its performance was compared with that of some recent excellent self-adaptive algorithms, namely SACABC (Li and Yin 2014), ISABC (Rajasekhar and Pant 2014), QAABC (He et al. 2013), and SSABC (Liu et al. 2015). These algorithms were applied to unimodal, multimodal, and composite benchmark functions. The optimization results of SABC-GB, SA

CABC, ISABC, QAABC, and SSABC on 25 test functions with different dimensions are shown in Tables 8 and 9. Overall, the final solutions generated by SABC-GB are better than those given by the other self-adaptive algorithms for all benchmark functions. This is because SABC-GB employs a

**Table 9** Optimization results of SABC-GB, SACABC, ISABC, QAABC and SSABC on 25 test functions with 50-D

| Functions | SACABC | | ISABC | | QAABC | | SSABC | | SABC-GB | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Mean | Std | Mean | Std | Mean | Std | Mean | Std | Mean | Std |
| Fun1 | 3.39E−12 | 1.00E−12 | 8.72E+04 | 3.00E+04 | 5.03E−10 | 1.62E−10 | 1.17E+01 | 2.94E+00 | 2.27E−13 | 4.64E−14 |
| Fun2 | 8.20E+04 | 2.69E+03 | 1.15E+05 | 4.36E+03 | 5.87E+04 | 3.66E+03 | 4.14E+04 | 1.81E+03 | 6.02E+04 | 3.79E+03 |
| Fun3 | 9.74E+07 | 1.77E+07 | 3.38E+08 | 6.30E+07 | 6.24E+07 | 6.69E+06 | 3.77E+07 | 3.79E+06 | 5.85E+07 | 9.96E+06 |
| Fun4 | 1.20E+05 | 7.15E+04 | 1.63E+05 | 1.95E+04 | 1.21E+05 | 1.29E+04 | 8.31E+04 | 6.43E+03 | 1.30E+04 | 6.23E+03 |
| Fun5 | 2.16E+04 | 7.52E+02 | 3.05E+04 | 2.25E+03 | 2.20E+04 | 1.70E+03 | 2.65E+04 | 6.57E+02 | 2.37E+04 | 1.26E+03 |
| Fun6 | 8.99E+01 | 2.03E+01 | 1.00E+11 | 2.14E+10 | 3.52E+01 | 1.01E+01 | 5.00E+03 | 9.29E+02 | 3.77E+00 | 1.91E+00 |
| Fun7 | 6.20E+03 | 7.65E−06 | 9.56E+03 | 6.80E+02 | 7.23E+03 | 1.13E+02 | 6.20E+03 | 2.44E−03 | 6.20E+03 | 0.00E+00 |
| Fun8 | 2.12E+01 | 3.07E−02 | 2.11E+01 | 4.19E−03 | 2.11E+01 | 7.09E−03 | 2.11E+01 | 4.03E−02 | 2.10E+01 | 0.00E+00 |
| Fun9 | 6.88E−08 | 5.94E−09 | 4.68E+02 | 9.20E+00 | 6.39E−02 | 7.79E−02 | 8.40E+00 | 1.16E−02 | 1.71E−13 | 0.00E+00 |
| Fun10 | 5.51E+02 | 3.39E+01 | 1.17E+03 | 7.42E+01 | 5.52E+02 | 3.65E+01 | 7.56E+02 | 4.21E+01 | 4.24E+02 | 3.92E+01 |
| Fun11 | 6.15E+01 | 5.17E−01 | 5.59E+01 | 1.51E+00 | 5.69E+01 | 1.68E+00 | 5.53E+01 | 1.33E+00 | 5.60E+01 | 1.61E+00 |
| Fun12 | 1.35E+05 | 4.22E+04 | 1.40E+06 | 6.97E+04 | 1.11E+05 | 3.81E+04 | 1.18E+05 | 1.59E+04 | 5.78E+04 | 1.28E+04 |
| Fun13 | 4.94E+00 | 3.00E−01 | 1.19E+03 | 2.24E+02 | 3.85E+00 | 6.50E−02 | 3.30E+00 | 1.66E−01 | 1.58E+00 | 2.61E−01 |
| Fun14 | 2.31E+01 | 2.07E−01 | 2.29E+01 | 2.01E−01 | 2.27E+01 | 9.81E−01 | 2.28E+01 | 3.21E−01 | 2.26E+01 | 1.98E−01 |
| Fun15 | 8.54E+01 | 1.12E+01 | 6.15E+02 | 4.31E+01 | 5.21E+01 | 9.86E+00 | 4.55E+01 | 1.66E+01 | 2.27E+01 | 9.77E+00 |
| Fun16 | 3.71E+02 | 2.94E+01 | 7.01E+02 | 6.47E+01 | 3.46E+02 | 1.14E+01 | 3.51E+02 | 4.29E+01 | 3.10E+02 | 3.58E+01 |
| Fun17 | 5.81E+02 | 1.95E+01 | 8.88E+02 | 7.27E+01 | 5.48E+02 | 3.29E+01 | 4.90E+02 | 2.79E+01 | 4.33E+02 | 1.93E+01 |
| Fun18 | 9.43E+02 | 4.56E+00 | 1.14E+03 | 2.14E+01 | 9.69E+02 | 8.91E+00 | 9.65E+02 | 1.36E+01 | 9.39E+02 | 4.18E+00 |
| Fun19 | 9.37E+02 | 5.60E+01 | 1.13E+03 | 5.32E+01 | 9.75E+02 | 7.31E+00 | 9.66E+02 | 1.49E+01 | 9.37E+02 | 6.34E+00 |
| Fun20 | 9.43E+02 | 4.80E+00 | 1.16E+03 | 2.90E+01 | 9.69E+02 | 1.37E+01 | 9.54E+02 | 7.76E+00 | 9.35E+02 | 3.88E+00 |
| Fun21 | 5.25E+02 | 2.16E+00 | 1.31E+03 | 4.18E+01 | 5.00E+02 | 1.20E−01 | 5.02E+02 | 6.31E−01 | 1.02E+03 | 5.59E−01 |
| Fun22 | 1.03E+03 | 5.19E+01 | 1.37E+03 | 5.06E+01 | 1.11E+03 | 5.21E+01 | 1.10E+03 | 4.61E+01 | 9.91E+02 | 3.46E+01 |
| Fun23 | 5.69E+03 | 1.31E+01 | 1.15E+03 | 5.33E+01 | 5.39E+03 | 5.17E−02 | 5.39E+03 | 1.64E−02 | 1.02E+03 | 0.00E+00 |
| Fun24 | 1.21E+03 | 4.04E+00 | 1.47E+03 | 4.31E+01 | 1.29E+03 | 6.57E+00 | 1.31E+03 | 5.27E+00 | 1.08E+03 | 1.37E+01 |
| Fun25 | 1.78E+03 | 6.20E+00 | 1.94E+03 | 2.49E+01 | 1.85E+03 | 1.09E+01 | 1.73E+03 | 7.15E+00 | 1.69E+03 | 5.95E+00 |

Mean and std denote the average and standard deviation of the corresponding function values obtained in 25 runs

self-adaptive strategy selection mechanism and solves each function using the best strategy. Additionally, SABC-GB finds the most accurate solutions for Fun1 and Fun9 in all dimensions, whereas SACABC, ISABC, QAABC, and SSABC cannot find precise solutions for Fun1 and Fun9 in all dimensions. This implies that these two problems are not easy to solve, but can be handled by SABC-GB. It is interesting to note that SABC-GB, SACABC, QAABC, and SSABC found the best solution for Fun21. The optima given by SABC-GB for Fun7, Fun9, and Fun23 are very stable for all dimensions. As shown in Tables 8 and 9, the solutions of f15–f25, which are composite functions, are relatively poor. However, SABC-GB outperformed the other algorithms in terms of accuracy.

Figure 3 shows the convergence performance of SABC-GB, SACABC, ISABC, QAABC and SSABC in solving three different types of functions. The convergence performance of SABC-GB is similar to that of SACABC, QAABC, and SSABC on Fun12 and Fun25 with $D=30$, but SABC-GB outperforms these self-adaptive algorithms significantly on Fun12 and Fun25 with $D=50$. The efficiency of SABC-GB is analogous to that of the other algorithms at the beginning of Fun6 and Fun12. However, the other approaches fall into local optima, and so the performance of SABC-GB is superior at the end of the evolution process. Thus, the convergence performance of SABC-GB is better than that of the other algorithms in general, and the results demonstrate that the modified self-adaptive strategies are very effective.

#### *4.2.4 Application of SABC-GB in a real-world problem*

To validate the feasibility of SABC-GB, we concluded experiments using real-life problems. This paper employs two standard datasets: Wine and Ionosphere (Frank and Asuncion 2010). The information of datasets is shown in Table 10, and the fitness comparisons between $K$-means and SABC-GB are shown in Table 11. In Table 11, the best results are in bold.

First, the initial centers were generated randomly. $K$-means was used to calculate the distance from each point to

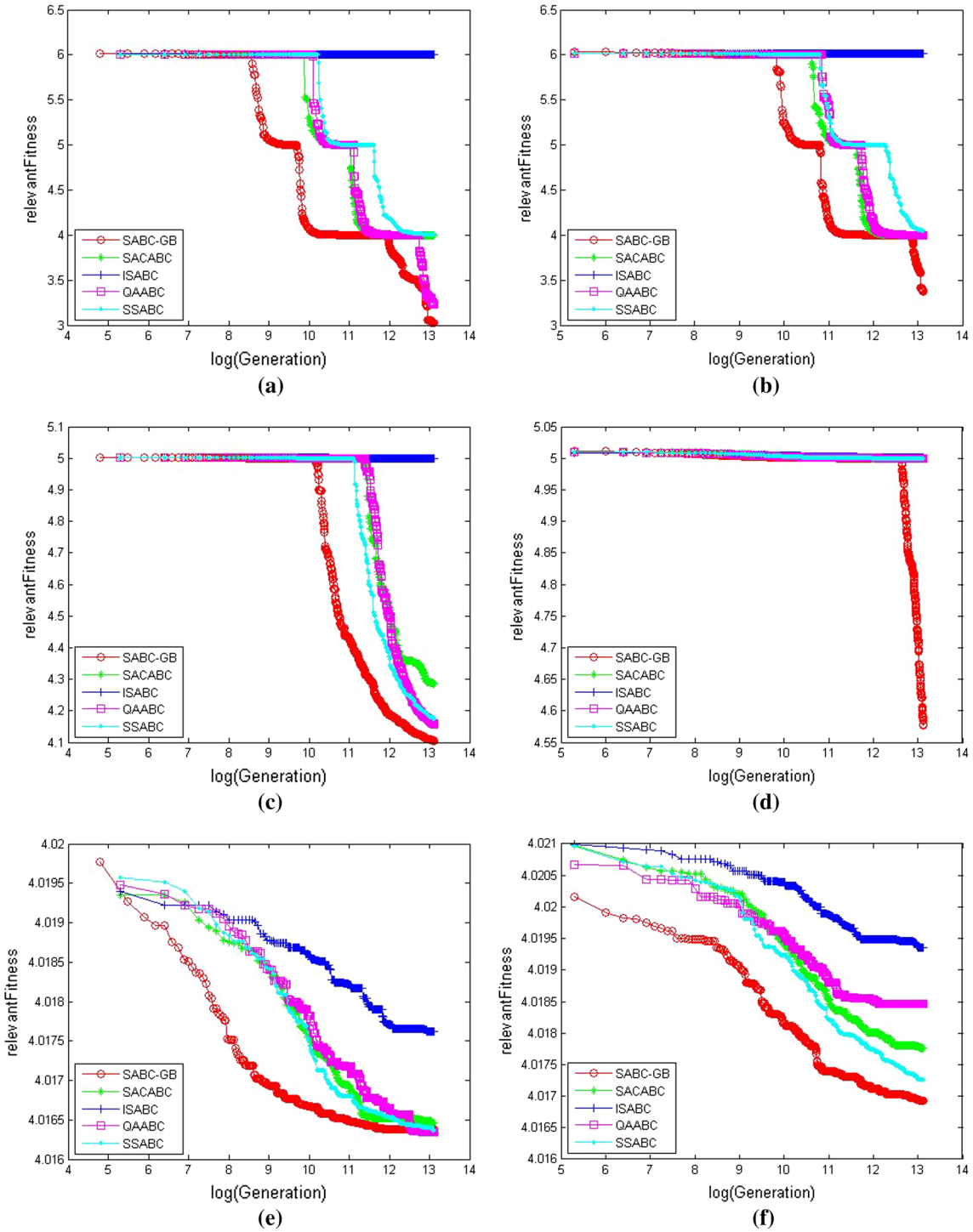


**Fig. 3** Convergence performance of SABC-GB, SACABC, ISABC, QAABC and SSABC. **a** Fun6 with $D=30$. **b** Fun6 with $D=50$. **c** Fun12 with $D=30$. **d** Fun12 with $D=50$. **e** Fun25 with $D=30$. **f** Fun25 with $D=50$

**Table 10** Datasets involved in experiments

| Datasets | Sample size | Classes | Dimension |
|---|---|---|---|
| Wine | 178 | 3 | 17 |
| Ionosphere | 351 | 2 | 34 |

the clustering centers. SABC-GB was then used to generate better centers according to previous experience in the evolutionary process. Previous experimental results show that the fitness is greatly influenced by the centers, and better centers would produce better results.

**Table 11** Fitness comparisons between $K$-means and SABC-GB

| Datasets | Algorithms | Minimum | Maximum | Mean | Std |
|---|---|---|---|---|---|
| Wine | $K$-means | 1.6550E+04 | 1.9320E+04 | 1.7288E+04 | 1.0521E+03 |
| | SABC-GB | **1.6531E+04** | **1.6550E+04** | **1.6545E+04** | **8.0829E+00** |
| Ionosphere | $K$-means | 7.9633E+02 | 7.9647E+02 | 7.9640E+02 | 7.2089E−01 |
| | SABC-GB | **7.4538E+02** | **7.4575E+02** | **7.4560E+02** | **1.1831E-02** |

It can be seen from Table 11 that the variation in the fitness values of SABC-GB is small, and that the standard deviation of the SABC-GB results is less than that of $K$-means on each dataset. SABC-GB can generate better fitness values and find better centers. This is mainly thanks to the self-adaptive candidate strategies, which adjust the search step length adaptively and find optimal solutions. Obviously, the standard deviation of SABC-GB is relatively small, because $K$-means find better solutions with better initial centers. The initialization and self-adaptive divisor of SABC-GB improve the clustering accuracy. From the above results, we conclude that the clustering precision and problem-solving efficiency are greatly improved by the proposed SABC-GB algorithm.

## 5 Conclusion

In this paper, we have proposed a novel algorithm called SABC-GB to solve global optimization problems. The initial populations of SABC-GB are generated by a dedicated algorithm, and the solution search strategy is self-adaptively selected. Experimental results using 25 benchmark functions show that the SABC-GB algorithm outperforms GABC, ABC/best/1, and ABC/best/2. The SABC-GB algorithm has excellent optimization ability, and it is a very strong algorithm. It avoids falling into local optima and is worthy of further research. The application of SABC-GB also indicates $K$-means clustering could be improved using SABC-GB.

However, the proposed approach also has some shortcomings. SABC-GB exhibited worse performance than previous algorithms with a few test functions. There are many other problems with this algorithm at present, but it should be pointed out that it is still a very good probabilistic algorithm.

The present study can be extended in various directions. To accelerate the search process and find better solutions to global optimization problems, an optimal means of combining local search with global search would clearly be beneficial. As the integration of self-adaptive candidate strategies outperformed previous algorithms, we intend to analyze candidate strategies and include the best strategies for each problem. We will also research the candidate solution pool size and the selection of strategies for this pool.

**Funding** This study was funded by National Natural Science Foundation of China (Grant Number 61403206), by Natural Science Foundation of Jiangsu Province (Grant Number BK20141005), by Natural Science Foundation of the Jiangsu Higher Education Institutions of China (Grant Number 14KJB520025), by Priority Academic Program Development of Jiangsu Higher Education Institutions.

**Compliance with ethical standards**

**Conflict of interest** Authors Yu Xue , Jiongming Jiang, Binping Zhao and Tinghuai Ma declares that they have no conflict of interest.

**Ethical approval** This article does not contain any studies with human participants or animals performed by any of the authors.

## References

Agrawal SK, Sahu OP (2015) Artificial bee colony algorithm to design two-channel quadrature mirror filter banks. Swarm Evol Comput 21:24–31

Alatas B (2010) Chaotic bee colony algorithms for global numerical optimization. Expert Syst Appl 37(8):5682–5687

Al-Salamah M (2015) Constrained binary artificial bee colony to minimize the make span for single machine batch processing with non-identical job sizes. Appl Soft Comput 29(C):379–385

Babaoglu I (2015) Artificial bee colony algorithm with distribution-based update rule. Appl Soft Comput 34:851–861

Bansal JC, Sharma H, Arya KV, Nagar A (2013) Memetic search in artificial bee colony algorithm. Soft Comput 17(10):1911–1928

Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1(1):53–66

Epitropakis MG, Tasoulis DK, Pavlidis NG, Plagianakos VP, Vrahatis MN (2011) Enhancing differential evolution utilizing proximity-based mutation operators. IEEE Trans Evol Comput 15(1):99–119

Frank A, Asunction A (2010) UCI machine learning repository. http://archive.ics.uci.edu/ml/datasets.html

Gao WF, Liu SY (2012) A modified artificial bee colony algorithm. Comput Oper Res 39(3):687–697

Gu B, Sheng VS (2013) Feasibility and finite convergence analysis for accurate on-line-support vector machine. IEEE Trans Neural Netw Learn Syst 24(8):1304–1315

Gao W, Liu S, Huang L (2012) A global best artificial bee colony algorithm for global optimization. J Comput Appl Math 236(11):2741–2753

Gao WF, Liu SY, Huang LL (2013) A novel artificial bee colony algorithm with Powell's method. Appl Soft Comput 13(9):3763–3775

Gao WF, Liu SY, Huang LL (2014) Enhancing artificial bee colony algorithm using more information-based search equations. Inf Sci 270:112–133

He P, Yan XD, Shi HB (2013) A quick self-adaptive artificial bee colony algorithm and its application. J East China Univ Sci Technol 5:588–595

Holland JH (1975) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press, Cambridge

Horng SC (2015) Combining artificial bee colony with ordinal optimization for stochastic economic lot scheduling problem. IEEE Trans Syst Man Cybern Syst 45(3):373–384

Kang F, Li JJ, Li HJ (2013) Artificial bee colony algorithm and pattern search hybridized for global optimization. Appl Soft Comput 13(4):1781–1791. doi:10.1016/j.asoc.2012.12.025

Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39(3):459–471

Kennedy J, Eberhart R (1995) Particle swarm optimization. IEEE Int Conf Neural Netw 4:1942–1948

Ji J, Pang W, Zheng Y, Wang Z, Ma Z (2015) A novel artificial bee colony based clustering algorithm for categorical data. PLoS ONE 10(5):e0127125. doi:10.1371/journal.pone.0127125

Li X, Yin M (2014) Self-adaptive constrained artificial bee colony for constrained numerical optimization. Neural Comput Appl 24(3):723–734

Li JQ, Pan QK (2015) Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. Inf Sci 316:487–502

Liu TT, Zhang CS, Zhang B, Sun RN (2015) A strategy self-adaptive selection bee colony algorithm based on feedback. J Northeast Univ 5(3):618–630

Macqueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley symposium on mathematical statistics and probability, vol 1, no 14, pp 281–297, University of California Press, Berkeley

Rahnamayan S, Tizhoosh HR, Salama MA (2008) Opposition-based differential evolution. IEEE Trans Evolut Comput 12(1):64–79

Rajasekhar A, Pant M (2014) An improved self-adaptive artificial bee colony algorithm for global optimisation. Int J Swarm Intell 1(2):115–132

Roy R, Sevick-Muraca EM (1999) Truncated Newton's optimization scheme for absorption and fluorescence optical tomography: part I theory and formulation. Opt Express 4(10):353–371

Setiono R, Hui LK (1995) Use of a quasi-newton method in a feedforward neural network construction algorithm. IEEE Trans Neural Netw 6(1):273–277

Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359

Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Nanyang Technological University, Singapore

Wen X, Shao L, Fang W, Xue Y (2015) Efficient feature selection and classification for vehicle detection. IEEE Trans Circuits Syst Video Technol 25(3):508–517

Xia F, Liu L, Li J, Ahmed AM, Yang LT, Ma J (2015) Beeinfo: interest-based forwarding using artificial bee colony for socially aware networking. IEEE Trans Veh Technol 64(3):1188–1200

Yi J, Gao L, Li X, Gao J (2016) An efficient modified harmony search algorithm with intersect mutation operator and cellular local search for continuous function optimization problems. Appl Intell 44(3):725–753

Zhang X, Zhang X, Ho SL, Fu WN (2014) A modification of artificial bee colony algorithm applied to loudspeaker design problem. IEEE Trans Magn 50(2):737–740

Zhu GP, Kwong S (2010) Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl Math Comput 217(7):3166–3173