Bacterial colony foraging optimization

Hanning Chen\textsuperscript{a,}\textsuperscript{*}, Ben Niu\textsuperscript{b}, Lianbo Ma\textsuperscript{a}, Weixing Su\textsuperscript{a}, Yunlong Zhu\textsuperscript{a}

\textsuperscript{a} Laboratory of Information Service and Intelligent Control, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China
\textsuperscript{b} College of Management, Shenzhen University, Shenzhen 518060, China

\textbf{A R T I C L E  I N F O}

Article history:
Received 23 November 2012
Received in revised form 14 March 2013
Accepted 17 April 2013
Available online 16 February 2014

Keywords:
Bacterial foraging
Chemotaxis
Self-adaptation
Cell-to-cell communication
Global optimization

\textbf{A B S T R A C T}

This paper proposes a novel bacterial colony foraging (BCF) algorithm for complex optimization problems. The proposed BCF extend original bacterial foraging algorithm to adaptive and cooperative mode by combining bacterial chemotaxis, cell-to-cell communication, and a self-adaptive foraging strategy. The cell-to-cell communication enables the historical search experience sharing among the bacterial colony that can significantly improve convergence. With the self-adaptive strategy, each bacterium can be characterized by focused and deeper exploitation of the promising regions and wider exploration of other regions of the search space. A rigorous performance analysis is given where the proposed algorithm is benchmarked against four state-of-the-art reference algorithms using both a classical and a composition test function suites. The individual and collective bacterial foraging behaviors of the proposed algorithmic model are also studied. Statistical analysis of all these tests highlights the significant performance improvement due to the beneficial combination and shows that the proposed algorithm outperforms the reference algorithms.

\textcopyright 2014 Elsevier B.V. All rights reserved.

1. Introduction

Nature serves as a rich source of concepts, principles, and mechanisms for designing artificial computational systems to solve complex engineering problems. In optimization domain, researchers have developed many effective stochastic techniques that mimic the specific structures or behaviors of certain creatures. For examples, genetic algorithm (GA), originally conceived by Holland [1], represent a fairly abstract model of Darwinian evolution and biological genetics; ant colony optimization (ACO), proposed by Marco Dorigo [2,3], is developed based on the foraging behaviors of real ant colonies; particle swarm optimization (PSO), proposed by Kennedy and Eberhart [4,5], glean ideas from social behaviors of bird flocking and fish schooling. These algorithms have been found to perform better than the classical heuristic or gradient-based methods, especially for optimizing the nondifferentiable, multimodal and discrete complex functions. Currently, these nature-inspired paradigms have already come to be widely used in many areas.

In recent years, the computational models of chemotaxis (i.e. the bacterial foraging behavior) have attracted more and more attention, due to its research potential in engineering applications. A few models have been developed to mimic bacterial foraging behavior and have been applied for solving some practical problems [6–8]. Among them, bacterial foraging optimization (BFO) is a successful population-based numerical optimization algorithm that mimics the foraging behavior of \textit{E. coli} bacteria [8]. Until now, BFO has been applied to some engineering problems, such as optimal control [9], optimal power flow [10], color image enhancement [11] and machine learning [12]. However, classical BFO algorithm suffers from two major drawbacks: (1) as a bacterial colony evolves, the fixed run-length unit of each bacterium may lead them trap in local optima or oscillate about the optima; and (2) there is no information sharing among the bacterial colony. Thus, experimentation with complex and multimodal benchmark functions reveals that the BFO algorithm possesses a poor convergence behavior and its performance heavily decreases with the growth of search space dimensionality and problem complexity.

Several BFO variants have been developed to improve its optimization performance. In [13], Tripathy and Mishra proposed an improved BFO algorithm using two approaches: (1) in order to speed up the convergence, the average value is replaced by the minimum value of all the chemotactic cost functions for deciding the bacterium’s health; and (2) for swarming, the distances of all the bacteria in a new chemotactic step are evaluated from the globally optimal bacterium to these points and not the distances of each bacterium from the rest of the others. Mishra [14] proposed a fuzzy bacterial foraging (FBF) algorithm using Takagi–Sugeno type fuzzy inference scheme to select the optimal chemotactic step size in BFO. The FBF algorithm was...
shown to outperform both classical BFO and a genetic algorithm when applied to the harmonic estimation problem. Kim et al. proposed a hybrid approach involving GA and BFO for function optimization [15]. The proposed algorithm outperformed both GA and BFO over a few numerical benchmarks and a practical PID controller design problem. Biswas et al. proposed a synergism of BFOA with the particle swarm optimization [16]. The new algorithm, named by the authors as bacterial swarm optimization (BSO), was shown to perform in a statistically better way as compared to both of its classical counterparts over several numerical benchmarks.

Three improved versions of BFO, namely the cooperative bacterial foraging optimizer (CBFO), adaptive bacterial optimization, and multi-colony bacterial foraging optimization (MCBFO), have been described in our previous works [29,30,34]. In [29], instead of simply considering chemotaxis behavior of single bacterium, MCBFO also introduced the communication strategies employed by natural multi-colony bacterial community in order to coordinate pattern emerges, which is lacking from the mentioned chemotaxis inspired algorithms. Then the MCBFO was evaluated on the static RFID network planning problem in comparison with GA and PSO. In [30], by employing the adaptive foraging strategies, two proposed ABFO algorithms are able to adjust the run-length unit parameter dynamically during algorithm execution in order to balance the exploration/exploitation tradeoff. In [34], the proposed CBFO applied two cooperative approaches to the original BFO, namely the serial heterogeneous cooperation on the implicit space decomposition level and the serial heterogeneous cooperation on the hybrid space decomposition level.

Building on the success of our preliminary work on bacterial foraging algorithms, this paper aims to demonstrate convincingly that the self-adaptive and communication approaches are both effective strategies and can be utilized to help scaling up the performance of bacterial foraging algorithms for solving complex optimization problems with high dimensionality. That is, this paper extends the classical BFO to a novel bacterial colony foraging (BCF) optimization model by mimics the area concentrated search (ACS) behaviors in nature. In the proposed BCF model, each artificial bacterium can climb the nutrient gradient based on not only its own experience but also the knowledge of the others; also, each bacterium can strike a balance between the exploration and the exploitation of the search space during its evolution, by adaptively tuning the magnitude of its chemotactic step size.

Several experiments were performed to evaluate the performance of the proposed algorithm. The first test-suite contains ten classical benchmark functions, namely the Sphere, Rosenbrock, Ackley, Rastrigrin, Griewank, Weierstrass, Sphere_noise, Noisy, Rotated hyper-ellipsoid, and Noncontinuous Rastrigin’s function functions. Moreover, this work used a novel composition test function suite developed by Liang et al. [17]. This benchmark set is made up by six composed functions, which were especially designed to test optimization algorithms providing many desirable properties, namely shifted, non-continuous, rotated, and narrow global basin of attraction. The proposed BCF algorithm has been compared with its classical counterpart, the classical BFO algorithm [8], the very popular swarm-intelligence algorithm known as PSO [45], a hybrid optimization technique that synergistically couples BFOA with PSO that called BSO [16], and a standard real-coded genetic algorithm (GA) [18] over both classical and composition test suites with respect to the statistical performance measures of solution quality and convergence speed.

The proposed BCF and its application on global optimization problem described in this paper enhance our previously proposed methods in the following aspects:

- The proposed BCF combines bacterial chemotaxis with cell-to-cell communication based cooperation. This combination successfully casts the bacterial foraging algorithm into cooperative foraging fashion and remarkably improves the convergence speed of the bacterial colony.
- By introducing a self-adaptive foraging strategy, namely the area concentrate search (ACS), the previous bacterial foraging optimization model has been extended to an adaptive scheme in this paper. This adaptive scheme can effectively maintain the bacterial colony diversity in the search process.

The rest of the paper is organized as follows. In Section 2, we outline the classical BFO in sufficient details. In Section 3, the proposed BCF algorithm will be introduced and its implementation details will be given. In Section 4, the experiment studies of the proposed BCF and the other algorithms are presented with descriptions of the benchmark functions, experimental settings, and experimental results. Finally, conclusions are drawn in Section 5.

2. Classical bacterial foraging optimization

Bacterial foraging optimization algorithm is inspired by an activity called “chemotaxis” exhibited by bacterial foraging behaviors. Motile bacteria such as E. coli and salmonella propel themselves by rotation of the flagella. To move forward, the flagella rotates counterclockwise and the organism “swims” (or “runs”), while a clockwise rotation of the flagellum causes the bacterium to randomly “tumble” itself in a new direction and swim again [19]. Alternation between “swim” and “tumble” enable the bacterium to search for nutrients in random directions. Swimming is more frequent as the bacterium approaches a nutrient gradient. Tumbling, hence direction changes, is more frequent as the bacterium moves away from some food to search for more. Basically, bacterial chemotaxis is a complex combination of swimming and tumbling that keeps bacteria in places of higher concentrations of nutrients. Chemotaxis can also be considered as the optimal foraging decision making capabilities of bacteria.

2.1. BFO model

The original BFO system consists of four principal mechanisms, namely chemotaxis, swarming, reproduction, and elimination-dispersal [8]. We briefly describe each of these processes as follows:

2.1.1. Chemotaxis

In the original BFO, a unit walk with random direction represents a “tumble” and a unit walk with the same direction in the last step indicates a “run”. Suppose \( \theta(j, k, l) \) represents the \( j \)th bacterium at \( j \)th chemotactic, \( k \)th reproductive, and \( l \)th elimination-dispersal step. \( C(i) \) is the chemotactic step-size for this bacterium during each run or tumble (run-length unit). Then in each computational chemotactic step, the movement of the \( j \)th bacterium can be represented as

\[
\theta(j + 1, k, l) = \theta(j, k, l) + C(i) \frac{\Delta(i)}{\sqrt{\Delta^2(i) + \Delta(i)}}
\]

where \( \Delta(i) \) is the direction vector of the \( j \)th chemotactic step. When the bacterial movement is run, \( \Delta(i) \) is the same with the last chemotactic step; otherwise, \( \Delta(i) \) is a random vector whose elements lie in \([-1, 1]\).

With the activity of “run” or “tumble” taken at each step of the chemotaxis process, a step fitness for each bacterium, denoted as \( f(i, j, k, l) \), will be evaluated.
2.1.2. Swarming

A simple cell-to-cell communication mechanism is also mimicked in the classical BFO model: as each bacterium moves, it releases attractant to signal other bacteria to swarm towards it; meanwhile, each bacterium releases repellent to warn other bacteria to keep a safe distance from it. BFO simulates this mechanism by representing the combined cell-to-cell attraction and repelling effect as:

\[ J_{c}(\theta, \theta) = \sum_{i=1}^{n} \left[ d_{\text{attract}} \exp(-a_{\text{attract}} \sum_{m=1}^{D} (\theta_i^m - \theta_m^m)^3) \right] 
+ \sum_{i=1}^{n} \left[ d_{\text{repellent}} \exp(-a_{\text{repellent}} \sum_{m=1}^{D} (\theta_i^m - \theta_m^m)^3) \right] \]  \hspace{1cm} (2)

where \( J_{c}(\theta, \theta) \) is the objective function value to be added to the actual objective function (to be minimized) to present a time varying objective function, \( S \) is the total number of bacteria, \( D \) is the number of variables to be optimized, and \( d_{\text{attract}}, a_{\text{attract}}, d_{\text{repellent}}, a_{\text{repellent}} \) are different coefficients that should be chosen properly.

2.1.3. Reproduction

The health status of each bacterium is calculated as the sum of the step fitness during its life, i.e. \( \sum_{j=1}^{n} j(i,j,k,l) \), where \( N_k \) is the maximum step in a chemotaxis process. All bacteria are sorted in descending values \( J(\theta) \) and for each \( i=1, 2, ..., S \), let \( J_{\text{health}} = \sum_{j=1}^{N_k+1} j(i,j,k,l) \) be the health of the bacterium. Sort bacteria in order of ascending values \( J_{\text{health}} \).

2.1.4. Elimination and dispersal

The chemotaxis provides a basis for local search, and the reproduction process speeds up the convergence which has been simulated by the classical BFO. While to a large extent, only chemotaxis and reproduction are not enough for global optima searching. Since bacteria may get stuck around the initial positions or local optima, it is possible for the diversity of BFO to change either gradually or suddenly to eliminate the accidents of being trapped into the local optima. In BFO, the dispersion event happens after a certain number of reproduction processes. Then some bacteria are chosen, according to a preset probability \( P_{ed} \) to be killed and moved to another position within the environment.

2.2. Step-by-step algorithm

In what follows we briefly outline the original BFO algorithm step by step:

[Step 1] Initialize parameters \( n, S, N_c, N_r, N_{re}, P_{ed}, C(i) \) (i=1,2,...,S). \( \theta \). Where,
- \( n \): dimension of the search space,
- \( S \): the number of bacterium,
- \( N_c \): chemotactic steps,
- \( N_r \): swim steps,
- \( N_{re} \): reproductive steps,
- \( N_{ed} \): elimination and dispersal steps,
- \( P_{ed} \): probability of elimination,
- \( C(i) \): the run-length unit. (i.e. the chemotactic step size during each run or tumble).

[Step 2] Elimination-dispersal loop: \( l=l+1 \).
[Step 3] Reproduction loop: \( k=k+1 \).
[Step 4] Chemotaxis loop: \( j=j+1 \).

[Substep a] For \( l=1, 2, ..., S \), take a chemotactic step for bacterium \( i \) as follows.

[Substep b] Compute fitness function \( J(i,j,k,l) \).
[Substep c] Let \( J_{\text{last}} = J(i,j,k,l) \) to save this value since we may find better value via a run.
[Substep d] Tumble: Generate a random vector \( \Delta(i) \in \mathbb{R}^4 \) with each element \( \Delta_m(i) = m, 1, 2, ..., S \), a random number on \( [-1, 1] \).
[Substep e] Move: Let the \( i \)th bacterium swim using Eq. (1). This results in a step of size \( C(i) \) in the direction of the tumble for bacterium \( i \).
[Substep f] Compute \( J(i,j+1,k,l) \) with \( \theta(j+1,k,l) \).
[Substep g] Swim:
- (i) Let \( m=0 \) (counter for swim length).
- (ii) While \( m < N_s \), if have not climbed down too long
- - Let \( m=m+1 \).
- - If \( J(i,j+1,k,l) < J_{\text{last}} \), let \( J_{\text{last}} = J(i,j+1,k,l) \). Then another step of size \( C(i) \) in this same direction will be taken as Eq. (1) and use the new generated \( \theta(j+1,k,l) \) to compute the new \( J(i,j+1,k,l) \).
- - Else let \( m=N_s \).
[Substep h] Go to next bacterium \( i+1 \) if \( i < S \) go to \( b \) to process the next bacterium.
[Step 5] If \( j < N_{re} \), go to step 3. In this case, continue chemotaxis since the life of the bacteria is not over.
[Step 6] Reproduction:
- [Substep a] For the given \( k \) and \( l \), and for each \( i=1, 2, ..., S \), let \( J_{\text{health}} = \sum_{j=1}^{N_k+1} j(i,j,k,l) \) be the health of the bacteria. Sort bacterium in order of ascending values \( J_{\text{health}} \).
- [Substep b] The \( S \) bacteria with the highest \( J_{\text{health}} \) values die and the other \( S \) bacteria with the best values split and the copies that are made are placed at the same location as their parent.
[Step 7] If \( k < N_{re} \), go to step 2. In this case the number of specified reproduction steps is not reached and start the next generation in the chemotactic loop.
[Step 8] Elimination-dispersal: For \( i=1, 2, ..., S \), with probability \( P_{ed} \), eliminate and disperse each bacterium, which results in keeping the number of bacteria in the population constant. To do this, if a bacterium is eliminated, simply disperse one to a random location on the optimization domain. If \( l < N_{ed} \), then go to step 2; otherwise end.

2.3. Bacterial behaviors in BFO

In order to get an insight into the behaviors of the virtual bacteria in BFO model, we illustrate the foraging trajectories of single bacterium and bacterial colony in two 2-D environments, namely the Rosenbrock function and the Ackley function, respectively. The formulations of Rosenbrock and Ackley functions have been presented in Section 4.

The first case is the minimization of the 2-D Rosenbrock function, which illustrates the swarming pattern of bacterial colony by the cell-to-cell attraction and repulsion in classical BFO. The motion trajectories of a bacterial colony with cell-to-cell communications are shown in Fig. 1, in which the search space that defines by the Rosenbrock function is contour plotted. At the beginning of the optimization process, six bacteria are scattered into random locations in the optimization domain. According to Eq. (2), the bacteria in the local optima attract those in the global optimum and the convergence speed of the whole population is pulled down; on the other hand, since the repelling effect results
in scattered (not concentrated) swarm of bacteria, the global optimum may be covered by the bacterial group whereas not occupied by any bacterium. Therefore, ultimately in Fig. 1 there are more bacteria gathering in the inferior optimum, the genuine global optimum is ignored. By analyzing the Eq. (2) and Fig. 1, we can see that the simple attract and repellent mechanism in classical BFO cannot take account for the information sharing of the global optima among bacterial colony and whilst slows down the speed of convergence and reduces the precision of the optimization.

The second case is the minimization of the 2-D Ackley function, which illustrates the foraging ability of single bacterium by tuning its run-length unit that can essentially influence the bacterial behaviors. Fig. 2 shows the contour lines of the Ackley function together with the foraging path of two bacteria that simultaneously start at (−1.5, 1.5) and (1.5, −1.5) with different chemotactic step-size (or run-length unit). From the chemotactic motions of the two virtual bacteria as in Fig. 2, we can observe that the bacterium with larger run-length unit C(1) = 0.1 that represented by the black line can explore the whole search space and stay for a while in several domains with local optima. It can also escape from these local optima to enter the domain with the global optimum, but was not able to stop there. On the other hand, the bacteria with the smaller run-length unit C(2) = 0.01 that represented by the blue line was attracted into the domain with a local optima, which closed to its start point, and exploited this local minimum for its whole life cycle. That is, if the bacteria with small chemotactic step-size traps in a local minima, it is not able to escape from it. Therefore, in the context of the classical BFO model, the bacteria with large chemotactic step-size have exploring ability (i.e. global investigation of the search place) while the bacteria with relatively small chemotactic step-size have exploiting skill (i.e. the fine search around a local optimum). However, it is difficult to decide which value of the static chemotactic step-size is best suited for BFO on a given problem.

3. The bacterial colony foraging algorithm

In the proposed BCF model, the most important contributions are: (1) to define the bacterial self-adaptive foraging strategy that dynamically balances the exploration and exploitation behaviors during the foraging process of each bacterium; and (2) to define the bacterial cell-to-cell communication mechanism that enables the information sharing among bacterial colony. Thus, we first introduce a widely-used adaptive foraging strategy in nature, namely the area concentrated search (ACS), and the bacterial group behavior based on cell-to-cell communication, namely quorum sensing, from which the proposed BCF algorithm glean ideas.

3.1. Area concentrated search

A central problem for the natural predators in the foraging process is how to balance two conflicting alternatives: the exploitation (i.e., to search thoroughly in promising areas) and the exploration (i.e., to move to distant areas potentially better than the actual one). Through natural selection, some predatory animals have developed the area concentrated search (ACS, also called area-restricted search) strategy, by which a predator is able to respond to variations in prey distributions by varying its searching efforts: following an encounter with food resource, a forager searches intensively in a more circumscribed region, while a failure to encounter a resource leads to a more extensive, less circumspect mode of search. That is, ACS assumes that regions dense with preys should be exploited slowly, to maximize the chances of encounter, and less dense regions explored rapidly, to minimize the time spent searching in unprofitable areas [20]. ACS in a continuous patchy environment thereby ensures that foraging behaviors will, to some extent, match the distribution of resource, and may be viewed as a simple form of optimal habitat selection [21]. The ACS behavior is regarded as an efficiently adaptive search strategy [22], which is employed by many search-intensive predators, such as birds, lizards, insects and other higher organisms.

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pseudocode for the dynamic self-adaptive strategy.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>FOR (each bacterium i) IN PARALLEL</td>
</tr>
<tr>
<td>2:</td>
<td>IF (Criterion-1) then | exploitation</td>
</tr>
<tr>
<td>3:</td>
<td>[C_i(t + 1) = C_i(t)/\lambda;]</td>
</tr>
<tr>
<td>4:</td>
<td>[s_i(t + 1) = s_i(t)/\lambda;]</td>
</tr>
<tr>
<td>5:</td>
<td>ELSE IF (Criterion-2) then | exploration</td>
</tr>
<tr>
<td>6:</td>
<td>[C_i(t + 1) = C_{\text{min}};]</td>
</tr>
<tr>
<td>7:</td>
<td>[s_i(t + 1) = s_{\text{max}};]</td>
</tr>
<tr>
<td>8:</td>
<td>ELSE</td>
</tr>
<tr>
<td>9:</td>
<td>[C_i(t + 1) = C_i(t);]</td>
</tr>
<tr>
<td>10:</td>
<td>[s_i(t + 1) = s_i(t);]</td>
</tr>
<tr>
<td>11:</td>
<td>END IF</td>
</tr>
<tr>
<td>12:</td>
<td>END FOR IN PARALLEL</td>
</tr>
</tbody>
</table>
3.2. Bacterial cooperation by cell-to-cell communication

As a new branch of microbiology, quorum sensing was discovered by Miller and Bassler [23]. Generally, it is a process that allows bacteria to search for similar cells in their close surroundings using secreted chemical signaling molecules called autoinducers. This is also called “cell-to-cell communication”. Recent studies of microorganisms have revealed that bacteria could function as groups and the individuals within the group could respond to and benefit from the group as a whole. That is, through cell-to-cell communication, bacteria can glean information from the environment and from other organisms, interpret such information into common knowledge and learn from past experience [24]. For this reason, bacterial colonies display diverse complex social behaviors, including cooperation in foraging, building, reproducing, purposeful alteration of colony structure and decision-making. In such a perspective, the bacterial colony behaves much like a multicellular organism or a social community [25].

The cooperative behavior illustrated by bacterial colony is regarded as a universal mechanism, which is employed by many natural animals, such as bird, lizard, fish and higher organisms. Drew inspiration from such cooperation within species, several successful particle swarm systems have been presented [26,35].

3.3. BCF model

Inspired by the above phenomenon in nature, this work extends the classical BFO to a self-adaptive and cooperative optimization model by constructing the following processes:

3.3.1. Self-adaptation

In order to combine the bacterial foraging with ACS strategy, this process enables the single bacterium to modify the magnitude of its “individual run-length unit” during the foraging. That is, each bacterium of the colony can modify its search behavior by...

![Flowchart of the BCF algorithm.](image)

Table 2. Classical test suite.

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
<th>Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere</td>
<td>$f_1(x) = \sum_{i=1}^{D} x_i^2$</td>
<td>$x_i \in [-5.12, 5.12] \in [0, 1]$</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>$f_2(x) = \sum_{i=1}^{D} 100 \times (x_{i+1}^2 - x_i^2)^2 + (1 - x_i)^2$</td>
<td>$x_i \in [-2.048, 2.048] \in [0, 1]$</td>
</tr>
<tr>
<td>Ackley</td>
<td>$f_3(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^{D} \cos(2 \pi x_i)) + 20 + c$</td>
<td>$x_i \in [-32, 32] \in [0, 1]$</td>
</tr>
<tr>
<td>Rastrigrin</td>
<td>$f_4(x) = \sum_{i=1}^{D} x_i^2 - 10 \sum_{i=1}^{D} \cos(2 \pi x_i) + 10$</td>
<td>$x_i \in [-5.12, 5.12] \in [0, 1]$</td>
</tr>
<tr>
<td>Griewank</td>
<td>$f_5(x) = \frac{1}{\prod_{i=1}^{D} x_i} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{2}}) + 1$</td>
<td>$x_i \in [-600, 600] \in [0, 1]$</td>
</tr>
<tr>
<td>Weierstrass</td>
<td>$f_6(x) = \sum_{i=1}^{D} [\sum_{j=1}^{D} \cos(2 \pi b_{ij}(x_j + 0.5))] - n \sum_{i=1}^{D} \cos(2 \pi b_{ii} \cdot 0.5)$</td>
<td>$x_i \in [-0.5, 0.5] \in [0, 1]$</td>
</tr>
<tr>
<td>Sphere_noise</td>
<td>$f_7(x) = (1 + 0.1 \times \text{rand}[0, 1]) \sum_{i=1}^{D} x_i^2$</td>
<td>$x_i \in [-5.12, 5.12] \in [0, 1]$</td>
</tr>
<tr>
<td>Noisy</td>
<td>$f_8(x) = \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} \text{rand}(0, 1)$</td>
<td>$x_i \in [-5.12, 5.12] \in [0, 1]$</td>
</tr>
<tr>
<td>Rotated hyper-ellipsoid</td>
<td>$f_9(x) = \sum_{i=1}^{D} [\sum_{j=1}^{D} (x_j + 0.5]^2$</td>
<td>$x_i \in [-65.536, 65.536] \in [0, 1]$</td>
</tr>
<tr>
<td>Noncontinuous Rastrigrin</td>
<td>$y_i = \frac{\text{rand}[0, 1]}{</td>
<td>x_i</td>
</tr>
</tbody>
</table>
analyzing the current status of its own. In this way, both the position (solution vector) and the chemotactic step-size of each bacterium can undergo evolution. In this self-adaptive foraging process, each bacterium alternatively displays two distinct search states:

- **Exploration**, during which the bacterium employs a large run-length unit to explore the previously unscanned regions in the search space as fast as possible.
- **Exploitation**, during which the bacterium uses a small run-length unit to exploit the promising regions slowly in its immediate vicinity.

Here each bacterium in the colony has to permanently maintain an appropriate balance between “Exploration” and “Exploitation” states by varying its own run-length unit adaptively. This adaptation is done by taking into account two decision indicators: (1) a fitness improvement, which means that the bacterium finds a promising domain; and (2) no fitness improvement registered lately, which means that the current domain is food exhausted. The criteria that determine the adjustment of individual run-length unit and the entrance into one of the states are defined as following:

- **Criterion-1**: if the bacterium discovers a promising domain, the run-length unit of this bacterium is adapted to a smaller one. Here “discovers a promising domain” means this bacterium registers a fitness improvement beyond a certain precision from the last generation to the current. Following Criterion-1, the bacterium’s behavior will self-adapt into the Exploitation state.
- **Criterion-2**: if the bacterium’s current fitness is unchanged for a number $K_u$ (user-defined) of consecutive generations, then augment this bacterium’s run-length unit and this bacterium enters the Exploration state. This situation means that the bacterium searches on an un-promising domain or the domain
where this bacterium focuses its search has nothing new to offer.

This self-adaptive strategy is given in pseudocode in Table 1. Where \( t \) is the current generation number, \( C(t) \) is the current run-length unit of the \( i \)th bacterium, \( \epsilon(t) \) is the required precision in the current generation of the \( i \)th bacterium, \( \lambda \) is the run-length unit decreasing parameter that is a user-defined constant, \( C^{\text{initial}} \) and \( \epsilon^{\text{initial}} \) are the initialized run-length unit and the precision goal respectively.

### 3.3.2. Cell-to-cell communication

In the analogy between bacterial quorum sensing and swarming pattern of PSO model [32], a novel principle of cell-to-cell communication for bacterial foraging algorithm is introduced.

In BCF model, when a bacterial turns, its choice of a new direction should not be governed by a probability distribution, while be dominated by the information combination of itself and its colony members. Accordingly, we introduce an additional direction component \( D_i \) to each bacterium. Then in the BCF algorithm, at the \( i \)th iteration the direction is computed as:

\[
D_i(t) = kD_i(t-1) + \phi_i R_1 \|X_{\text{pi}} - X_i(t-1)\| + \phi_i R_2 \|X_{\text{si}} - X_i(t-1)\|
\]

where \( k \) is the weight for the previous direction of the \( i \)th bacterium, which represents how the bacterium trusts its own status at present location, \( X_{\text{pi}} \) is the best position where this bacterium had been, \( X_{\text{si}} \) is the overall global best position ever achieved by the bacterial colony, \( \|X_{\text{pi}} - X_i(t-1)\| \) and \( \|X_{\text{si}} - X_i(t-1)\| \) are both unit vectors for indicating the directions only. \( \phi_i \) and \( \phi_i \) are the learning rates that control the influence levels of cognitive and social components to make different swimming directions, \( R_1 \) and \( R_2 \) are random numbers uniformly distributed in \([0, 1]\).

In this context, each bacterium adjusts its tumble angle according to the personal historical experience and the bacterial colony social knowledge. This cell-to-cell communication based cooperation may provide bacteria with more accurate information about the search because it is the whole colony that engaged in searching the solution space not just the single bacterium.

### 3.3.3. Enhanced chemotaxis

Then in each chemotactic step, the self-adaptive chemotactic step-size \( C \) controls the swim amplitude taken by the \( i \)th bacterium towards a desired direction, which is specified by the cell-to-cell communication based tumble direction \( D_i \):

\[
X_i(t) = X_i(t-1) + C(t-1)D_i(t-1)
\]

It should be noted that the major difference between the classical BFO and the proposed BCF is: in BFO, reproduction and elimination-dispersal processes are applied to ensure convergence and colony diversity, respectively; while in BCF, the self-adaptation

<table>
<thead>
<tr>
<th>Functions</th>
<th>BCF</th>
<th>BFO</th>
<th>PSO</th>
<th>BSO</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>Best</td>
<td>7.5543e−057</td>
<td>12.7788</td>
<td>1.8656e−055</td>
<td>4.9749e−004</td>
</tr>
<tr>
<td>Worst</td>
<td>4.1406e−050</td>
<td>31.3801</td>
<td>6.9834e−044</td>
<td>0.0011</td>
<td>0.1746</td>
</tr>
<tr>
<td>Mean</td>
<td>6.0662e−051</td>
<td>22.3747</td>
<td>3.5030e−045</td>
<td>8.3078e−004</td>
<td>0.0939</td>
</tr>
<tr>
<td>Std</td>
<td>1.3102e−050</td>
<td>6.1294</td>
<td>1.5613e−044</td>
<td>1.6060e−004</td>
<td>0.0433</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>Best</td>
<td>0</td>
<td>9.1282e+006</td>
<td>0.0317</td>
<td>0.3680</td>
</tr>
<tr>
<td>Worst</td>
<td>0.4837</td>
<td>4.7983e+007</td>
<td>6.8775</td>
<td>9.3028</td>
<td>2.5378e+003</td>
</tr>
<tr>
<td>Mean</td>
<td>0.2359</td>
<td>2.5727e+007</td>
<td>3.5233</td>
<td>4.7843</td>
<td>501.0495</td>
</tr>
<tr>
<td>Std</td>
<td>0.1357</td>
<td>1.0547e+007</td>
<td>2.1613</td>
<td>2.2227</td>
<td>611.4213</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>Best</td>
<td>3.5527e−015</td>
<td>15.4152</td>
<td>3.5527e−015</td>
<td>0.0047</td>
</tr>
<tr>
<td>Worst</td>
<td>3.5527e−015</td>
<td>7.8396</td>
<td>3.5527e−015</td>
<td>0.0380</td>
<td>0.7858</td>
</tr>
<tr>
<td>Mean</td>
<td>0.1501</td>
<td>71.4563</td>
<td>0.1795</td>
<td>0.2240</td>
<td>0.9297</td>
</tr>
<tr>
<td>Std</td>
<td>0.0742</td>
<td>129.2755</td>
<td>0.0841</td>
<td>0.0931</td>
<td>0.3875</td>
</tr>
<tr>
<td>( f_4 )</td>
<td>Best</td>
<td>0</td>
<td>7.2698</td>
<td>0</td>
<td>1.9527</td>
</tr>
<tr>
<td>Worst</td>
<td>0.0039</td>
<td>11.0638</td>
<td>0.4593</td>
<td>3.0549</td>
<td>9.4945</td>
</tr>
<tr>
<td>Mean</td>
<td>0.0025</td>
<td>9.4676</td>
<td>0.0038</td>
<td>2.5988</td>
<td>2.0553</td>
</tr>
<tr>
<td>Std</td>
<td>6.1556e−004</td>
<td>1.2219</td>
<td>0.0117</td>
<td>0.2565</td>
<td>1.2341</td>
</tr>
<tr>
<td>( f_5 )</td>
<td>Best</td>
<td>1.4814e−035</td>
<td>18.9217</td>
<td>6.5939e−011</td>
<td>9.3498e−004</td>
</tr>
<tr>
<td>Worst</td>
<td>1.7392e−031</td>
<td>39.6094</td>
<td>4.5858e−009</td>
<td>0.0034</td>
<td>17.8932</td>
</tr>
<tr>
<td>Mean</td>
<td>3.7036e−032</td>
<td>26.3679</td>
<td>1.0680e−009</td>
<td>0.0022</td>
<td>6.8486</td>
</tr>
<tr>
<td>Std</td>
<td>5.3866e−032</td>
<td>57.8004</td>
<td>1.4433e−009</td>
<td>0.0031</td>
<td>4.9662</td>
</tr>
<tr>
<td>( f_6 )</td>
<td>Best</td>
<td>5.8551e−029</td>
<td>18.1046</td>
<td>0.0012</td>
<td>5.9051e−004</td>
</tr>
<tr>
<td>Worst</td>
<td>1.3887e−022</td>
<td>31.6904</td>
<td>0.0035</td>
<td>0.0014</td>
<td>14.9319</td>
</tr>
<tr>
<td>Mean</td>
<td>1.4238e−023</td>
<td>25.2225</td>
<td>0.0022</td>
<td>9.5108e−004</td>
<td>4.3179</td>
</tr>
<tr>
<td>Std</td>
<td>4.3795e−023</td>
<td>14.3067</td>
<td>6.8855e−004</td>
<td>2.7905e−004</td>
<td>3.9466</td>
</tr>
<tr>
<td>( f_7 )</td>
<td>Best</td>
<td>1.4589e−004</td>
<td>5.6722e−005</td>
<td>2.0526e−004</td>
<td>1.5109e−005</td>
</tr>
<tr>
<td>Worst</td>
<td>0.0038</td>
<td>0.0438</td>
<td>0.0023</td>
<td>0.0022</td>
<td>0.0746</td>
</tr>
<tr>
<td>Mean</td>
<td>0.0038</td>
<td>0.0079</td>
<td>0.0012</td>
<td>0.0022</td>
<td>0.0366</td>
</tr>
<tr>
<td>Std</td>
<td>8.8900e−004</td>
<td>0.0140</td>
<td>0.0064</td>
<td>0.0022</td>
<td>7.9292e−004</td>
</tr>
<tr>
<td>( f_8 )</td>
<td>Best</td>
<td>1.0997</td>
<td>62.2500</td>
<td>9.2112</td>
<td>3.4268</td>
</tr>
<tr>
<td>Worst</td>
<td>3.7425</td>
<td>140.7500</td>
<td>8.0254</td>
<td>8.0254</td>
<td>19.3078</td>
</tr>
<tr>
<td>Mean</td>
<td>2.1522</td>
<td>91.8884</td>
<td>4.4047</td>
<td>5.0048</td>
<td>12.5887</td>
</tr>
<tr>
<td>Std</td>
<td>0.8508</td>
<td>24.2865</td>
<td>1.7146</td>
<td>1.6161</td>
<td>5.4971</td>
</tr>
</tbody>
</table>
process can help the bacterium avoid being trapped into local optima and keep the species diversity in a satisfied level, at the same time, the cell-to-cell based information sharing can significantly speed up the bacterial colony to converge to the global optima. Therefore, the reproduction and elimination-dispersal processes are discarded in BCF. The flowchart of the BCF algorithm is illustrated in Fig. 3, where $S$ is the colony size, $t$ is the chemotactic generation counter from 1 to max-generation, $i$ is the bacterium’s ID counter from 1 to $S$, $X_i$ is the position of the $i$th bacterium, $N_s$ is the maximum number of steps for a single activity of swim, $\text{flag}$ is the number of generations that the $i$th bacterium has not improved its own fitness.

4. Benchmark function test

4.1. Classical test functions

The classical test suite includes ten 30-dimension benchmark functions, which are commonly used in evolutionary computation literature [17,27,33] to show solution quality and convergence rate. The first problem is the unimodal Sphere function ($f_1$) that is easy to solve. The second problem is the Rosenbrock function ($f_2$), which has a narrow valley from the perceived local optima to the global optimum and can be treat as a multimodal problem. Ackley function ($f_3$) has one narrow global optimum basin and many minor local optima. Rastrigrin function ($f_4$) is a complex multimodal problem with a large number of local optima. Griewank function ($f_5$) has a $\prod_{i=1}^{D} \cos (x_i / \sqrt{i})$ component causing linkages among variables, thereby making it difficult to reach the global optimum. Weierstrass function ($f_6$) is continuous but differentiable only on a set of points. Sphere_noise ($f_7$) and Noisy function ($f_8$) are two noisy problems where the uniformly distributed random noises are added to the objective functions. Due to the presence of noises, the objective functions keep changing from time to time and each of them become a challenge for an optimization algorithm to locate the optimum. Rotated hyper-ellipsoid function ($f_9$) is produced by rotated the coordinate axes of axes parallel hyper-ellipsoid. It is continuous, convex and unimodal. Noncontinuous Rastrigin function ($f_{10}$) is constructed based on the Rastrigin’s

![Fig. 10. The median convergence results of classical test functions. (a) Sphere, (b) Rosenbrock, (c) Ackley, (d) Rastrigrin, (e) Griewank, (f) Weierstrass, (g) Sphere_noise, (h) Noisy, (i) Rotated hyper-ellipsoid, and (j) Noncontinuous Rastrigrin.](image-url)
function and has the same number of local optima as the continuous Rastrigin function. The formula for each classical test function is shown in Table 2.

4.2. Composition test functions

The second test suite contains six composition functions ($C_f_1$–$C_f_6$), each of which is composed by 10 basic functions of 10 dimensions:

$$C_f_1 \ldots C_f_6(x) = \sum_{i=1}^{n} w_i f_i((x - o_i + o_{old,i})/\lambda_i M_i) + \text{bias}_i + f_{bias}$$ (6)

where $n$ is the number of basic function, $w_i$ is the weight value for each $f_i(x)$, $f_i(x)$ is the $i$th basic function used to construct the composition function, $o_i$ is the new shifted optimum position for each $f_i(x)$, $o_{old,i}$ is the old optimum position for each $f_i(x)$, $\lambda_i$ is used to stretch or compress the function, $M_i$ is the orthogonal rotation matrix for each $f_i(x)$, and $\text{bias}_i$ defines which optimum is global optimum.

The basic function set that constructs composition functions is given by the Sphere, Rastrigin, Weierstrass, Griewank, and Ackley functions, which have been formulated in Table 2. The search range for all the composed functions is $[-5, 5]^D$. The detailed description of the method for constructing these composite functions can be referred to [17].

In general the multimodal composite function $C_f_1$ (Fig. 4) was constructed using 10 unimodal Sphere functions, this resulting function has one global optimum and nine local optima. $C_f_2$ (Fig. 5), and $C_f_3$ (Fig. 6) were constructed using 10 Griewank and Rastrigin functions, respectively. Since they have more complex multimodal functions, localizing their local optima becomes more complex. $C_f_4$, $C_f_5$, and $C_f_6$ are all called hybrid functions because they are constructed with different basic functions. That is, $C_f_4$ (Fig. 7) is composed by the Ackley function ($f_1$–$f_2$), Rastrigin function ($f_3$–$f_4$), Weierstrass function ($f_5$–$f_6$), Griewank function ($f_7$–$f_8$), Sphere function ($f_9$–$f_{10}$). Sphere function ($f_9$–$f_{10}$). $C_f_5$ (Fig. 8) and $C_f_6$ (Fig. 9) are composed by the Rastrigin function ($f_1$–$f_2$), Weierstrass function ($f_3$–$f_4$), Griewank function ($f_5$–$f_6$), Ackley function ($f_7$–$f_8$), Sphere function ($f_9$–$f_{10}$). The difference between $C_f_5$ and $C_f_6$ is the values of $\sigma$ and $\lambda$, which makes $C_f_6$ has a narrower coverage area for the basic function with the global optimum and a flatter coverage area.

![Box plot](image-url)

Fig. 11. A simple box plot.
for the basic function with the local optima. For these composite functions the global optima are very difficult to reach even when the global optima areas have been found.

4.3. Settings

To fully evaluate the performance of the proposed BCF algorithm, four successful nature-inspired optimization algorithms were used for comparison:

- Canonical particle swarm optimization with constriction factor (PSO) [28];
- Classical bacterial foraging optimization (BFO) [8];
- Bacteria swarm optimization (BSO) [16]; and
- Standard genetic algorithm with elitism (GA) [18].

In all experiments in this section, the values of the population size used in each algorithm were chosen to be the same as 50. The other specific parameters of algorithms are given as follows:

**PSO Settings**: For canonical PSO, the learning rates $c_1$ and $c_2$ are both 2.05, the constriction factor $\chi=0.729$, and the global communication topology is used [31].

**BFO Settings**: For the classical BFO algorithm, we take $C=0.1$ and $P_{ed}=0.25$, which is the standard set of these two parameter values as recommended in [8]. We set $N_c=200$ or 100 for classical and composition functions respectively, $N_s=4$, $N_{re}=5$, and $N_{ed}=2$, then BFO performs totally 2000 (or 1000) chemotactic steps in each run, which make a fair comparison in regard of the parameter values.

**BSO Settings**: The BSO algorithm, referred to as the bacterial swarm optimization, performs local search through the

---

*Fig. 12.* ANOVA test for all algorithms on classical test functions. Here 1, 2, 3, 4 and 5 is the algorithm index of BCF, PSO, GA, BFO, BSO, respectively. (a) Sphere, (b) Rosenbrock, (c) Ackley, (d) Rastrigrin, (e) Griewank, (f) Weierstrass, (g) Sphere_noise, (h) Noisy, (i) Rotated hyper-ellipsoid, and (j) Noncontinuous Rastrigrin. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)
chemotactic movement operation of BFO whereas the global search over the entire search space is accomplished by a social-only PSO operator. Parametric setup for the algorithm was kept exactly same as described in [16]. For the PSO operator we choose inertia weight as 0.8 and social learning rate as 1.494; for the BFO operators the parameter-values were kept same as described in BFO Settings.

**GA Settings:** The experiment employed a binary coded standard GA having random selection, crossover, mutation and elite units. Stochastic uniform sampling technique was the chosen

<table>
<thead>
<tr>
<th>Functions</th>
<th>BCF</th>
<th>BFO</th>
<th>PSO</th>
<th>BSO</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C_f_1)</td>
<td>Best</td>
<td>7.7864e-023</td>
<td>385.3748</td>
<td>0.0479</td>
<td>0.0177</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>309.0175</td>
<td>749.5304</td>
<td>400</td>
<td>600.0163</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>160.0902</td>
<td>543.6946</td>
<td>170.0000</td>
<td>170.0277</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>102.9637</td>
<td>305.4715</td>
<td>115.9502</td>
<td>165.7474</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>326.6335</td>
<td>981.2563</td>
<td>412.2278</td>
<td>409.7877</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>174.1728</td>
<td>704.4957</td>
<td>206.1286</td>
<td>188.5714</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>99.3604</td>
<td>177.0293</td>
<td>121.0506</td>
<td>157.6103</td>
</tr>
<tr>
<td>(C_f_2)</td>
<td>Best</td>
<td>8.2786</td>
<td>614.8877</td>
<td>11640e+003</td>
<td>409.1549</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>407.4050</td>
<td>911.7979</td>
<td>249.5267</td>
<td>971.9039</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>171.6260</td>
<td>305.4715</td>
<td>206.1286</td>
<td>188.5714</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>133.5019</td>
<td>184.0234</td>
<td>71.9234</td>
<td>258.6098</td>
</tr>
<tr>
<td>(C_f_3)</td>
<td>Best</td>
<td>294.6677</td>
<td>606.2175</td>
<td>298.3002</td>
<td>313.1631</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>608.4582</td>
<td>1.4776e+003</td>
<td>721.5762</td>
<td>800.0023</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>375.8556</td>
<td>1.0816e+003</td>
<td>464.4095</td>
<td>481.8318</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>97.2601</td>
<td>269.5876</td>
<td>164.5941</td>
<td>157.1497</td>
</tr>
<tr>
<td>(C_f_4)</td>
<td>Best</td>
<td>2.7576</td>
<td>628.9068</td>
<td>405.14</td>
<td>3.8620</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>416.5646</td>
<td>1.1668e+003</td>
<td>511.6996</td>
<td>803.1800</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>166.0945</td>
<td>860.4539</td>
<td>175.7435</td>
<td>254.0215</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>161.3410</td>
<td>175.0730</td>
<td>178.5401</td>
<td>236.2705</td>
</tr>
<tr>
<td>(C_f_5)</td>
<td>Best</td>
<td>502.9761</td>
<td>939.8848</td>
<td>500.7369</td>
<td>500.5456</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>902.8107</td>
<td>1.0067e+003</td>
<td>902.0582</td>
<td>902.6389</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>821.9914</td>
<td>957.2450</td>
<td>821.2012</td>
<td>856.4042</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>167.5225</td>
<td>21.4395</td>
<td>168.8585</td>
<td>123.5293</td>
</tr>
</tbody>
</table>
selection method. Single point crossover operation with the rate of 0.8 was employed. Mutation operation restores genetic diversity lost during the application of reproduction and crossover. Mutation rate in the experiment was set to be 0.01.

**BCF Settings:** For the proposed BCF algorithm, the initialized run-length unit $C_{\text{initial}}$ was set to be 1% of the search space, the initial precision goal $\varepsilon_{\text{initial}} = 100$, the generation limit for jumping out of local optima $K_u = 20$, and after performing a series of hand-tuning experiments, we found that keeping the run-length unit decreasing parameter $\lambda = 10$ provides considerably good result over all benchmark functions considered here.

### 4.4. Experiment 1: numerical results and comparison

In this section, the numerical results were classified in two groups: experiments with classical test functions and experiments with composition test functions.

#### 4.4.1. Results for the classical test functions

Table 3 presents the best, worst, mean and standard deviations of the 30 runs of the five involved algorithms on the classical test suite. The best results among the five algorithms are shown in bold in Table 3. Fig. 10a–j presents the average convergence characteristics of each algorithm for each classical test function.

From Table 3, the proposed BCF achieved significantly better results than all the other algorithms. From Fig. 10, we can observe that BCF is the fastest one for finding good results within relatively few generations. For example, BCF found the global minimum every time of two noise function $f_7$ and $f_8$ and can also consistently find the minimum of multimodal function $f_3$ within relatively fewer generations, while the other algorithms generated poorer results on them. On function $f_5$ and $f_9$, the BCF algorithm yielded similar results to the PSO, BSO and GA, while still achieved significantly better performance than the original BFO algorithm. Comparing the results and the convergence graphs, BCF has good local and global search abilities and converges fast according to its self-adaptive character, which will be elaborately studied in the next section.

---

**Fig. 13.** The median convergence results of composition test functions. (a) composition function 1, (b) composition function 2, (c) composition function 3, (d) composition function 4, (e) composition function 5, and (f) composition function 6.
In order to determine whether the results obtained by BCF are statistically different from the results generated by other algorithms, the analysis of variance (ANOVA) test are conducted between the BCF’s result and the result achieved by the other four algorithms for each problem. In this work, the graphical ANOVA analyses are done through the box plots that are shown in Fig. 11. A box plot is a graphical tool, which provides an excellent visual summary of many important aspects of a distribution. The box stretches from the lower hinge (defined as the 25th percentile) to the upper hinge (the 75th percentile) and therefore contains the middle half of the scores in the distribution. The median is shown as a line across the box. Therefore, one-fourth of the distribution is between this line and the top of the box and one-fourth of the distribution is between this line and the bottom of the box. Looking at the box plot, the general features of the distribution can be noticed.

Fig. 12 implies the statistical performance representation of all algorithms on each classical function in 30 runs. From this box plot representation, it is clearly visible and proved that the BCF provides better results than that of all the other algorithms. The box plots shown in Fig. 12a–j indicate the performance of each involved algorithms on the classical test suite individually. According to the red lines in Fig. 12a–j, which refer to the low median values obtained by each algorithm, it is very clear that the proposed BCF yields optimal result in all five algorithms.

4.4.2. Results for the composition test functions

The experiments conducted on 30-D classical test suite are repeated on the 10-D composition problems. The optimization results, the convergence characteristics, and the box plots of the ANOVA test of the 30 runs are presented in Table 4, Figs. 13 and 14 respectively. From these results, we can observe that the algorithms achieved similar ranking as in the classical test problems. All composition functions become more difficult than their classical counterparts and the results are not as good as in the classical test cases, although BCF still surpasses all other algorithms on all functions composition test suite. Better results were achieved on
the composition function 3 by BCF: as it traps all other algorithms in local optima, the BCF successfully avoids falling into the deep local optimum which is far from the global optimum. With the new self-adaptive foraging strategy, the artificial bacteria can dynamic balance their exploration and exploitation behaviors. Due to this, the BCF explores a larger search space than the original BFO. The larger search space is not achieved randomly. Instead, it is based on the historical search experience sharing among the whole bacterial colony. Because of this, the BCF performs comparably to or better than many successful evolutionary and SI-based algorithms on most of the composition functions.

4.5. Experiment 2: bacterial behavior simulation in BCF model

In order to further analyze the self-adaptive and group foraging behaviors of the BCF model, the evolution dynamics of the single artificial bacterium and the bacterial colony are both simulated in this section.

In the first simulation, the evolution of the BCF population was simulated on 2-D Sphere, Rosenbrock, Rastrigrin and griewank functions, which are illustrated in Fig. 15(a–d) respectively. In each case, the bacteria colony is distributed randomly over the nutrient map defined by each function. The evolution process proceeds 1000 chemotatic steps, and the other parameters were the same as in Section 4.3, except \( S = 6 \).

In Fig. 15(b), we can observe that with the new cell-to-cell communication mechanism, all the bacteria have found the long and narrow valley of the Rosenbrock function (which contains the global optimum) and move around it in the end of the foraging phase. We found a similar pattern in Fig. 15(c), where the bacteria colony pursue the valleys and avoid the peaks of the multimodal Rastrigrin function. In the first phase the bacterial colony explore many regions of the nutrient map. According to the information transferred by the bacteria in the better positions, the other bacteria join these food sources and then find a good deal of local optima, including the global optimum. The similar group foraging pattern can be also found in Fig. 15(a) and (d), which also plotted the foraging trajectories of bacterial colony on Sphere and Griewank function.

In the second simulation, we demonstrate the self-adaptive foraging behaviors of a single bacterium in the BCF model. Fig. 16 (a–d) illustrates the trajectories of a single bacterium on four benchmark functions, namely the 2-D Dejong, Rosenbrock, Rastrigrin and Griewank function, respectively. In each case, the evolution process proceeds 1000 chemotatic steps, and the other parameters were the same as in Section 4.3, except \( S = 1 \).

Fig. 16(a) illustrates the bacterial trajectories in the 2-D unimodal Sphere function, which start at point \((-5, -5)\). As we can see, the proposed self-adaptive strategy is important because it permits the bacteria to refine its foraging behaviors adaptively. At the beginning of the simulation, the bacterium starts exploring the search space. In that manner, the bacterium does not waste much time before finding the promising region that contains the global optimum, because the large run-length unit encourages long-range search. On the other hand, by self-adapting the parameters, the bacterium slows down (i.e. the bacterium enter the exploitation state) near the optimum in order to pursue the more and more precise solutions. Fig. 16(c) illustrates the bacterial trajectories (start at point \((-4, -4)\)) in the contour plotted 2-D multimodal Rastrigrin function. We can observe that the bacterium is switching between exploitation and exploration states by self-adapting its run-length unit. Whenever the bacterium encounters a fitness improvement, this forager starts searching intensively in

---

**Fig. 15.** Cooperative foraging trajectories of bacterial colony on (a) Sphere function; (b) Rosenbrock function; (c) Rastrigrin function; and (d) Griewank function.
Fig. 16. Self-adaptive foraging trajectories of single bacterium on (a) Sphere function; (b) Rosenbrock function; (c) Rastrigrin function; and (d) Griewank function.

Fig. 17. The run-length unit shifting on (a) Sphere function; (b) Rosenbrock function; (c) Rastrigrin function; and (d) Griewank function.
this promising region. Whereas, whenever it is highly probable that the good solutions lying in this region have been found by this bacterium, it moves away from this region and starts to explore the other regions of the search space until another better region is discovered. Finally, we can observe that the bacterium find the domain that contains the global optimum. The similar self-adaptive pattern can be noticed in Fig. 16(b) and (d), which plotted the bacterial foraging trajectories on Rosenbrock and Griewank function respectively. Clearly, this result captures the important aspects of the ACS mechanism that takes place in nature.

In Fig. 17, we have also drawn the associated evolution of the run-length unit of this bacterium along its search on all the functions. This provides an intuitive explanation of what is happening during the search of the self-adaptive bacterial foraging algorithm. From Fig. 17, we can notice that when the bacterium finds difficulties during an exploit state (with low run-length unit), then it increases the run-length unit to improve the exploration, and vice versa.

5. Conclusions

In order to apply bacterial foraging inspired algorithms to solve complex optimization problems efficiently and effectively, we have proposed a new bacterial colony foraging algorithm (BCF). The proposed BCF combines several optimal foraging approaches, namely the bacterial chemotaxis, cell-to-cell communication based cooperation, and a self-adaptive foraging strategy called area concentrate search (ACS). This combination successfully cast bacterial foraging algorithm into the adaptive and cooperative fashion. In BCF, each bacterium can be characterized by focused and deeper exploitation regions of the search space. The algorithm achieves this by decreasing the run-length unit of each bacterium to encourage exploitation in the self-adaptive bacterial foraging algorithm. From Fig. 17, we can notice that the bacterium finds difficulties during an exploit state (with low run-length unit), then it increases the run-length unit to improve the exploration, and vice versa.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grant nos. 61105067, 61174164, 71001072 and 71271140, and the Strategic Cooperation Project of Foshan and Chinese Academy of Sciences under Grant no. 2011BY100383.

References


An extensive performance analysis has been provided. Results show the proposed algorithm outperformed four state-of-the-art nature inspired algorithms on both a classical test suite and a recently developed composition test suite that totally including 16 mathematical functions. The individual bacterial foraging behaviors and the whole colony collective behaviors of the proposed algorithmic model were also studied. The simulation results proved that BCF has significantly better performance than the other four algorithms.
Hanning Chen received the B.Sc. degree in Mechanical Design and Automation and the M.Sc. degree in Computer Software and Theory from Yanshang University, Qinhuangdao, China, in 2002 and 2005, respectively, and the Ph.D. degree in Mechanical And Electronic Engineering from Chinese Academy of Sciences, China, in 2009. He has been an Assistant Research Fellow in the Key Laboratory of Industrial Informatics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China. His current research interests are in the area of biologically inspired computation and the application of these techniques for the problems in the areas of Engineering, Economics and Life Sciences.

Yunlong Zhu is the director of the Key Laboratory of Advanced Manufacturing Technology, Shenyang Institute of Automation of the Chinese Academy of Sciences. He received his Ph.D. in 2005 from the Chinese Academy of Sciences, China. He has research interests in various aspects of Enterprise Information Management but he has ongoing interests in RFID technology, Artificial Intelligence, Machine Learning, and related areas. Prof Zhu’s research has led to a dozen professional publications in these areas.

Ben Niu received the B.S. degree in Mechanical Design and Manufacture from Hefei Union University, Hefei, China, in 2001, the M.S. degree in Enterprise Information Management from Anhui Agriculture University, Hefei, China, in 2004, and the Ph.D. Degree in Mechanical and Electrical Engineering from Shenyang Institute of Automation of the Chinese Academy of Sciences, Shenyang, China, in 2008. He is presently serving as an associate professor in Department of Management Science, Shenzhen University. Currently he is also a Post-doc at Hefei Institute of Intelligent Machines, CAS and at e-Business Technology Institute, The University of Hong Kong, respectively. His main fields of research are Swarm Intelligence, Bio-inspired Computing, Multi-objective Optimization and their applications on RFID System, Business Intelligence, and Portfolio Optimization.

Lianbo Ma received the B.Sc. degree in Communication Engineering and the M.Sc. degree in Communication and Information System from Northeast University, Shenyang, China, in 2004 and 2007, respectively, and he is currently pursuing his Ph.D. in the Shenyang Institute of Automation of the Chinese Academy of Sciences. His current research interests include Swarm Intelligence, Bioinformatics and Computational Biology, FPGA technology and Printed Electronics.

Weixing Su received his B. S. and M. C. degree from Northeastern University, Shenyang, China, in 2003 and 2006 respectively. He is currently a lecturer and a doctoral student at Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China. His research interests are data mining, intelligent data processing and outlier detection technology.