Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2009, Article ID 815247, 17 pages
doi:10.1155/2009/815247

Research Article

Cooperative Bacterial Foraging Optimization

Hanning Chen, Yunlong Zhu, and Kunyuan Hu

Key Laboratory of Industrial Informatics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, Liaoning 110016, China

Correspondence should be addressed to Hanning Chen, chenhanning@sia.cn

Received 28 March 2009; Accepted 4 August 2009

Recommended by Manuel De La Sen

Bacterial Foraging Optimization (BFO) is a novel optimization algorithm based on the social foraging behavior of *E. coli* bacteria. This paper presents a variation on the original BFO algorithm, namely, the Cooperative Bacterial Foraging Optimization (CBFO), which significantly improve the original BFO in solving complex optimization problems. This significant improvement is achieved by applying 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. The experiments compare the performance of two CBFO variants with the original BFO, the standard PSO and a real-coded GA on four widely used benchmark functions. The new method shows a marked improvement in performance over the original BFO and appears to be comparable with the PSO and GA.

1. Introduction

In recent years, bacterial foraging behaviors (i.e., bacterial chemotaxis) as a rich source of potential engineering applications and computational model have attracted more and more attentions. A few models have been developed to mimic bacterial foraging behaviors and been applied for solving practical problems [1–3]. Among them, Bacterial Foraging Optimization (BFO) is a population-based numerical optimization algorithm. Until date, BFO has been applied successfully to some engineering problems, such as optimal control [4], harmonic estimation [5], transmission loss reduction [6] and machine learning [7]. However, experimentation with complex optimization problems reveal that the original BFO algorithm possesses a poor convergence behavior compared to other nature-inspired algorithms and its performance also heavily decreases with the growth of the search space dimensionality.

It should be noted that even the most successful nature-inspired optimization techniques, such as Genetic Algorithm (GA) [8, 9] and Particle Swarm Optimization (PSO) [10, 11], are also sensitive to the increase of the problem complexity and dimensionality, due to their stochastic nature [12, 13]. Cooperative search is one of the solutions to this problem that have been extensively studied in the past decade. The basic approach involves having more than one search module running and exchanging information among each other in order to explore the search space more efficiently and reach better solutions [14]. In order to improve the BFO's performance on complex optimization problems, this paper applies cooperative search technique to the BFO model and then proposed the Cooperative Bacterial Foraging Optimization (CBFO) algorithm. In order to evaluate the performance of the CBFO, extensive studies based on a set of 4 widely used benchmark functions have been carried out. For comparison purposes, we also implemented the original BFO, the standard PSO, and a simple real-coded GA on these functions, respectively. The simulation results are encouraging. The CBFO algorithm shows remarked performance improvement over the original BFO.

The rest of the paper is organized as follows. In Section 2, we will give the briefly reviews of the bacterial chemotaxis and the original BFO algorithm. A discussion of the artificial bacterial behaviors in BFO model is also presented in this section. Section 3 summarizes the state of the art on the cooperative search methods. Then our Cooperative Bacterial Optimization algorithm will be introduced and its implementation details will be given in Section 4. Section 5 tests the algorithms on the benchmarks, and gives out the results. Finally, Section 6 outlines the conclusions.

2. The Classical BFO Algorithm

The motile bacteria such as *E. coli* and salmonella propel themselves by rotating their flagella. To move forward, the flagella counterclockwise rotate and the organism "swims" (or "runs"). While a clockwise rotation of the flagellum causes the bacterium randomly "tumble" itself in a new direction and then swims again [15]. An alternation between "swim" and "tumble" enables the bacterium 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 concentration of nutrients. Bacterial chemotaxis can also be considered as the optimization process of the exploitation of known resources, and costly exploration for new, potentially more valuable resources.

2.1. Bacterial Foraging Optimization

The original Bacterial Foraging Optimization system consists of three principal mechanisms, namely, chemotaxis, reproduction, and elimination-dispersal. 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 $\theta^i(j,k,l)$ represents the bacterium at jth chemotactic, kth reproductive, and lth elimination-dispersal step.

$C(i)$ is the chemotactic step size during each run or tumble (i.e., run-length unit). Then in each computational chemotactic step, the movement of the ith bacterium can be represented as

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}, \tag{2.1}$$

where $\Delta(i)$ is the direction vector of the jth 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, denoted as $J(i,j,k,l)$, will be evaluated.

2.1.2. Reproduction

The health status of each bacterium is calculated as the sum of the step fitness during its life, that is, $\sum_{j=1}^{N_c} J(i,j,k,l)$, where N_c is the maximum step in a chemotaxis process. All bacteria are sorted in reverse order according to health status. In the reproduction step, only the first half of population survives and a surviving bacterium splits into two identical ones, which are then placed in the same locations. Thus, the population of bacteria keeps constant.

2.1.3. 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 $n, S, N_c, N_s, N_{re}, N_{ed}, P_{ed}, C(i)$ $(i = 1,2,\ldots,S), \theta^i$, where

n: dimension of the search space,

S: the number of bacteria in the colony,

N_c: chemotactic steps,

N_s: 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 size of the step taken in 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 4.1. For $i = 1 = 1, 2, \ldots, S$, take a chemotactic step for bacterium i as follows.

Substep 4.2. Compute fitness function, $J(i, j, k, l)$.

Substep 4.3. Let $J_{\text{last}} = J(i, j, k, l)$ to save this value since we may find better value via a run.

Substep 4.4. Tumble. Generate a random vector $\Delta(i) \in R^n$ with each element $\Delta m(i)$, $m = 1, 2, \ldots, n$, a random number on $[-1, 1]$.

Substep 4.5. Move. Let

$$\theta^i(j+1, k, l) = \theta^i(j, k, l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}. \tag{2.2}$$

This results in a step of size $C(i)$ in the direction of the tumble for bacterium i.

Substep 4.6. Compute $J(i, j+1, k, l)$ with $\theta^i(j+1, k, l)$.

Substep 4.7. Swimming.

(i) Let $m = 0$ (counter for swim length).

(ii) While $m < N_s$ (if has not climbed down too long), the following hold.

- 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 (2.2) and use the new generated
    - $\theta^i(j+1, k, l)$ to compute the new $J(i, j+1, k, l)$.
    - Else let $m = N_s$.

Substep 4.8. Go to next bacterium $(i + 1)$. if $i \neq S$, go to Substep 4.2 to process the next bacterium.

Step 5. If $j < N_c$, go to Step 3. In this case, continue chemotaxis since the life of the bacteria is not over.

Step 6. Reproduction.

Substep 6.1. For the given k and l, and for each $i = 1, 2, \ldots, S$, let

$$J^i_{\text{health}} = \sum_{j=1}^{N_c+1} J(i, j, k, l) \tag{2.3}$$

be the health of the bacteria. Sort bacteria in order of ascending values (J_{health}).

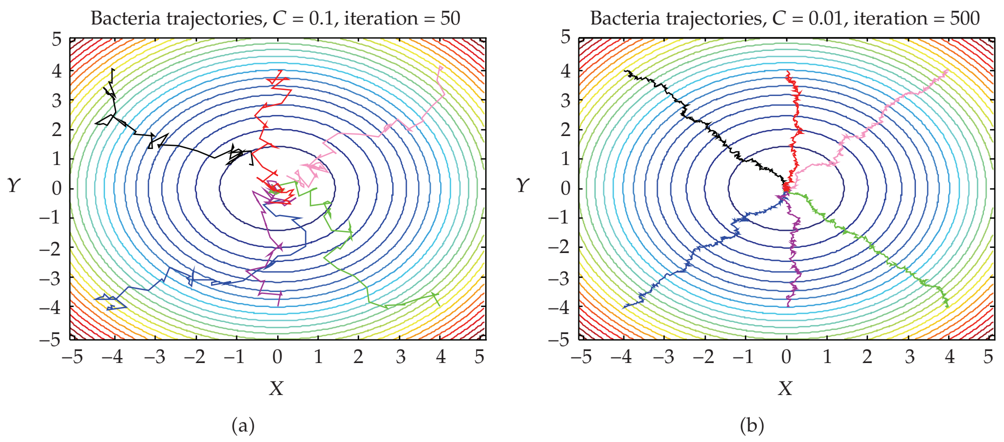


Figure 1: Bacterial Foraging trajectories on the 2D Sphere function.

Substep 6.2. The S_r bacteria with the highest J_{health} values die and the other S_r 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 $i = 1, 2, \ldots, S$, 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 Behavior in BFO*

In order to get an insight into the behavior of the virtual bacteria in BFO model, we illustrate the bacterial trajectories in two distinct environments (the 2D unimodal Sphere function and the 2D multimodal Rastrigrin function) by tuning the run-length unit parameter C, which can essentially influence the bacterial behaviors.

The first case is the minimization of the 2D Sphere function (formulated in Section 5), which is a widely used unimodal benchmark with a single optimum $(0, 0)$ and the minimum is 0. Figure 1 illustrates the trajectories of six bacteria foraging for the minimum in the landscape defined by the Sphere function (which is contour plotted). The six bacteria simultaneously start at $(-4, -4), (0, -4), (4, -4), (4, 4), (0, 4)$, and $(-4, 4)$. In Figure 1(a), the simulation takes 50 chemotactic steps with $C = 0.1$. While in Figure 1(b), the simulation takes 200 chemotactic steps with $C = 0.01$. The Reproduction and Elimination-dispersal events are not considered here. From Figure 1, it is clear to see that all the bacteria can travel up the gradient to pursue the minimum. We can also observe from Figure 1 that the larger the parameter C, the smaller the number of steps to enter the domain including the optimum, although the path seems to be miss direct sometimes. On the other hand, the bacteria with the smaller C enhance a fine-grained local search to find more precise solutions.

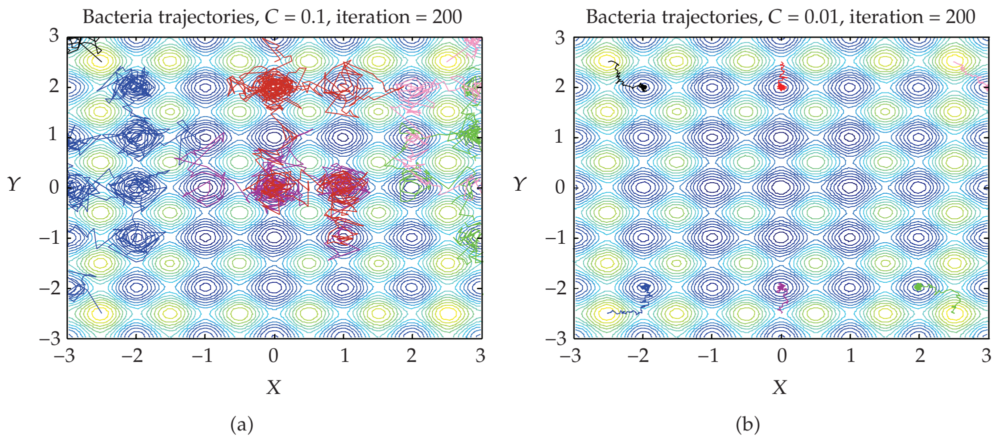


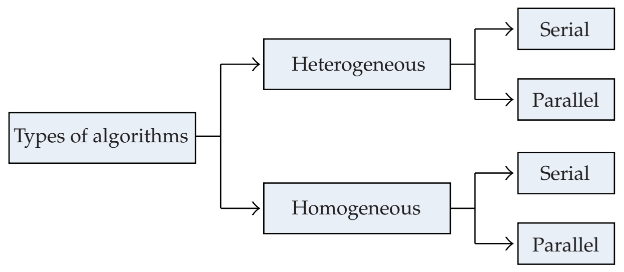
Figure 2: Bacterial Foraging trajectories on the 2D Rastrigrin function.

The other simulation case is on the 2D Rastrigrin function, which is a complex multimodal problem with a large number of local optima. It is a widely used benchmark function with the global optimum $(0,0)$ and the minimum is 0. Figure 2 shows the contour lines of this function together with the foraging path of six bacteria that simultaneously start at $(-2.5,-2.5)$, $(0,-2.5)$, $(2.5,-2.5)$, $(2.5,2.5)$, $(0,2.5)$, and $(-2.5,2.5)$. In Figure 2(a), the simulation takes 200 chemotactic steps with $C = 0.1$. While in Figure 2(b), the simulation takes 200 chemotactic steps with the different parameter $C = 0.01$. From the chemotactic motions in Figure 2, we can observe that the bacteria with larger $C = 0.1$ can explore the search space and stay for a while in several regions with local optima. they can also escape from these local optima to enter the domain with the global optimum, but were not able to stop there. On the other hand, the bacteria with the smaller C were attracted into the domain with local optima, which closed to these organisms, and exploited these local minimum for their whole life cycles. That is, if a bacterium with small C traps in a local minimum, it is not able to escape from it.

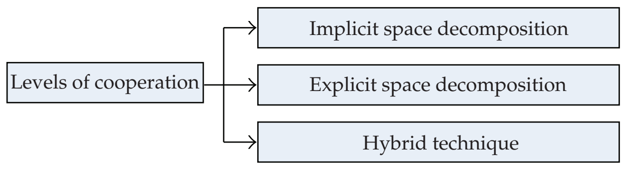
Obviously, the bacteria with large run length unit have the exploring ability while the bacteria with relatively small run length unit have the exploiting skill. Hence, the parameter C can be used to control the exploration of the whole search space and the exploitation of the promising areas.

3. Taxonomy of the Cooperative Search Algorithms

In this section, we review the classification of cooperative search algorithms, which can also be used to guide the readers in classifying the kind of algorithms we are dealing with in this work. El-Abd and Kamel proposed two different taxonomies for classifying cooperative search algorithms [14]: one is based on the types of algorithms used in it, and the other is based on the level of space decomposition achieved by the cooperative system (shown as in Figure 3).



(a)



(b)

Figure 3: Taxonomy of the Cooperative Search Algorithms. (a) Taxonomy based on the types of used algorithms; (b) Taxonomy based on the level of space decomposition.

The first taxonomy gives rise to four different categories.

(i) Serial homogenous cooperation: this is concerned with having different instances of the same algorithms searching in a sequential manner. Each algorithm provides a partial solution to the problem. These partial solutions are used to provide a complete solution that is evaluated and used in subsequent runs.

(ii) Parallel homogenous cooperation: this category involves having different instances of the same algorithm running in parallel and searching for a solution with the information passed between these algorithms.

(iii) Serial heterogeneous cooperation: this class involves having different algorithms running in a pipeline fashion. The output of each algorithm supplied as an input to the next algorithm.

(iv) Parallel heterogeneous cooperation: the same as the second class but with different running algorithms.

The second taxonomy gives rise to three different categories.

(i) Implicit space decomposition: this category involves the decomposition of the search space between different algorithms, which refers to having different algorithms (or different instances of the same algorithm) looking for a solution and sharing useful information between them.

(ii) Explicit space decomposition: in this class, each algorithm searches for a subsolution in a different subspace of the problem. That is, each algorithm provides a partial solution and these partial solutions are combined into the complete solution. This approach was originally introduced using genetic algorithms [16] and also applied to the original PSO algorithm [17].

(iii) Hybrid approach: this class refers to the idea of having a cooperative system that employs both methods of space decomposition.

Furthermore, these types of cooperation are related to each other. For example, the explicit space decomposition class is similar to the homogeneous serial class in the first taxonomy.

4. Cooperative Bacterial Foraging Optimization

This work proposes two variants of Cooperative Bacterial Foraging Algorithm, namely, CBFO-S and CBFO-H. According to the taxonomies on cooperation search algorithms in Section 3, they can be classified into the serial heterogeneous cooperation on the implicit space decomposition level and the serial heterogeneous cooperation on the hybrid space decomposition level, respectively.

4.1. The CBFO-S Algorithm

As indicate in Section 2, the bacterium with a large run-length unit parameter has the exploring ability, while the bacterium with a relatively small run-length unit parameter has the exploiting skill. This inspired us to divide the foraging procedure of artificial bacteria colony into multiple phases each occupies a portion of generations and characterized by the different value of run-length unit parameter C.

In CBFO-S, different BFO algorithms (with different run-length unit parameters) execute in sequential phases. The output of each BFO (the best positions found by each bacterium in each phase) supplies as an input to the algorithm in the next phase. In the initial phase, the bacteria colony searches the whole solution space with a large C, which permits the bacteria to locate promising regions and avoid trapped in local optima. Each bacterium records all its visited positions in this phase and the position with the highest fitness value is considered as the promising solution candidate. When entrance into the next phase, the bacteria colony is reinitialized with relatively small C from these promising solution candidates and starts exploiting the promising regions (the neighborhoods of these candidates) until the needed criteria for switch to the next phase is reached. Then a bacteria colony is reinitialized again with even smaller C to fine-tune the best-so-far solutions found in the foregoing phase. Hence, the CBFO-S algorithm can be classified into the serial heterogeneous cooperation on the implicit space decomposition level.

The pseudocode of CBFO-S is described in Algorithm 1, where N_p indicates the number of evolutionary phases, N_c represents the number of chemotactic steps in a bacterium's life time, S is the bacteria colony size and N_s is the maximum number of steps in the process of Run, and $\alpha > 1$ is a user-defined constant that is used to decrease the run-length unit C. We also embed the reproduction, elimination, and dispersal processes into each chemotactic step. This can speed up the algorithm convergence rate significantly.

4.2. The CBFO-H Algorithm

The CBFO-H Algorithm consists of two search stages working in a serial fashion. The first stage, which applied the original BFO model with a large run-length unit parameter C_L, runs for a number of iterations to locate promising regions including the global optimum. Then the algorithm passes the best found solutions to the next stage. The second stage reinitializes the bacteria colony in these best-so-far positions with a smaller run-length unit parameter C_S,

```
INITIALIZE: the position and the associated run-length unit $C_{initial}$ of the bacteria colony;
  For(each phase $k = 1 : N_p$)
    For(each chemotactic step $t = 1 : N_c$)
      For(each bacterium $i = 1 : S$)
        Calculate the fitness $J^i(t,k)$ of $i$th bacterium;
        TUMBLE: Generate a random vector $\Delta$, where each element belongs to $[-1,1]$.
                Move the bacterium $i$ in the direction of $\Delta\sqrt{\Delta^T\Delta}$ by a unit
                walk of size $C(k)$. Then calculate the new fitness
                $J_i(t+1,k)$ of bacterium $i$;
        Set $flag = 0$;
        RUN: While($flag < N_s$)
                If($J_i(t+1,k) < J_i(t,k)$)
                  Take another unit walk in the same direction;
                  Calculate the new fitness as $J_i(t+1,k)$;
                  $flag = flag + 1$;
                End if
             End while
      End for
        REPRODUCTION: The $S/2$ bacteria with the worst fitness die and the
                      other $S/2$ bacteria with the best fitness split;
        ELIMINATION and DISPERSAL: With probability $p_{ed}$, eliminate and
                                   disperse each bacterium;
    End for
      REINITIALIZE: bacteria positions from the potential candidate positions
                    found by each bacterium in the phase $k$.
      EVOLUTION: Evolution is added to run-length unit by:
                 $C(k+1) = C(k)/\alpha$; //$\alpha$ is user-defined constant.
  End for
```

Algorithm 1: Pseudocode for CBFO-S.

and applies the explicit space decomposition cooperative approach to the BFO. This approach relies on splitting the search space (n-dimensional vector) into $n/2$ subspaces (which is a 2-dimensional vector), where each subspace is optimized by a separate bacteria colony. The overall solution is the vector combining the best bacterium of each colony. This algorithm works by sequentially passing over the colonies. To evolve all the bacteria in colony j, the other $n/2 - 1$ components in the overall solution are kept constant (with their values set to the global best bacteria from the other $n/2 - 1$ colonies). Then the jth colony evolves and replaces the jth component of the overall solution by its best bacterium.

The pseudocode of CBFO-H is described in Algorithm 2, where N_c^{s1} and N_c^{s2} represent the number of chemotactic steps in stages 1 and 2, respectively, and $\alpha > 1$ is a user-defined constant that is used to decrease the run-length unit C.

5. Experiments

5.1. Benchmark Functions

The set of benchmark functions contains four functions that are commonly used in evolutionary computation literature [18] to show solution quality and convergence rate. The formulas and the properties of these functions are listed as follows.

```
Stage 1: the original BFO algorithm
  INITIALIZE: the position and the associated run-length unit $C_L$ of the bacteria colony;
  For(each chemotactic step $t = 1 : N_c^{s1}$)
    For(each bacterium $i = 1 : S$)
      TUMBLE;
      RUN;
      REPRODUCTION;
      ELIMINATION and DISPERSAL;
    End for
  End for
  PASS the best found solutions of each bacterium to stage 2;
Stage 2: the multi-colony cooperative BFO algorithm using explicit space decomposition
    REINITIALIZE: bacteria positions from the best found solutions and the associate
                  run-length unit $C_S$.
    SPLIT the whole population into $n/2$ separate colonies of 2D vectors;
    For(each chemotactic step $t = 1 : N_c^{s2}$)
      For(each colony $j = 1 : n/2$)
        For(each bacterium $i = 1 : S$)
          TUMBLE;
          RUN;
          REPRODUCTION;
          ELIMINATION and DISPERSAL;
        End for
        UPDATE the best bacterium replace the $j$th component of the overall solution;
      End for
      EVOLUTION: Evolution is added to run-length unit by:
                  If ($t$ mod $\beta = 0$)        // $\beta$ is user-defined constant.
                     $C(t+1) = C(t)/\alpha$;    // $\alpha$ is user-defined constant.
                  End if
    End for
```

Algorithm 2: Pseudocode for CBFO-H.

(1) Sphere function

$$f_1(x) = \sum_{i=1}^{n} x_i^2. \tag{5.1}$$

(2) Rosenbrock function

$$f_2(x) = \sum_{i=1}^{n} 100 \times \left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2. \tag{5.2}$$

(3) Rastrigrin function

$$f_3(x) = \sum_{i=1}^{n} x_i^2 - 10\cos(2\pi x_i) + 10. \tag{5.3}$$

Table 1: Parameters of the test functions.

Function	R	X^*	$f(X^*)$
Sphere	$[-5.12, 5.12]^D$	$[0, 0 \cdots 0]$	0
Rosenbrock	$[-2.048, 2.048]^D$	$[1, 1 \cdots 1]$	0
Rastrigrin	$[-5.12, 5.12]^D$	$[0, 0 \cdots 0]$	0
Griewank	$[-600, 600]^D$	$[0, 0 \cdots 0]$	0

(4) Griewank function

$$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1. \tag{5.4}$$

The first problem is the sphere function, which is a widely used unimodal benchmark and is easy to solve. The second problem is the Rosenbrock function. It can be treated as a multimodal problem. It has a narrow valley from the perceived local optima to the global optimum. The third problem, namely, the Rastrigrin function, is a complex multimodal problem with a large number of local optima. Algorithms may easily fall into a local optimum when attempting to solve it. The last problem is the multimodal Griewank function. Griewank has linkage among variables that makes it difficult to reach the global optimum. The interesting phenomenon of the Griewank is that it is more difficult for lower dimensions than higher dimensions. All functions are tested on 10 dimensions. The search ranges R, the global optimum X^* and the corresponding fitness $f(X^*)$ value of each function are listed in Table 1 .

5.2. Parameter Settings for the Involved Algorithms

Experiment was conducted to compare five algorithms, namely, the original BFO, the simple real-coded GA, the PSO with inertia weight, and the proposed CBFO-S and CBFO-H on the four benchmark functions with 10 dimensions. The parameters setting for BFO, CBFO-S, and CBFO-H are summarized in Table 2.

The PSO algorithm we used is the standard one and the parameters were given by the default setting of the work in [18]: the acceleration factors c_1 and c_2 were both 2.0; a decaying inertia weight ω starting at 0.9 and ending at 0.4 was used. The population size was set at 50 for the PSO algorithm.

The GA algorithm we executed is a real-coded Genetic Algorithm with intermediate crossover and Gaussian mutation. The population of the GA is 50 and all the control parameters, for example, mutation rate, crossover rate, and so forth, were set to be the same of [19].

5.3. Simulation Results for Benchmark Functions

This experiment runs 30 times respectively for two proposed CBFO variants, the original BFO, the real-coded GA, and the Particle Swarm Optimization on each benchmark function.

Table 2: Parameters of the BFO algorithms.

Type	BFO	CBFO-S	CBFO-H
S	50	50	50
N_s	4	4	4
N_c	100	100	—
N_{re}	5	—	—
N_{ed}	2	—	—
N_p	—	10	—
C	$10^{-3} \times$ R	—	—
P_{ed}	0.1	0.1	0.1
C_{initial}	—	$10^{-2}\times$R	—
α	—	10	10
C_L	—	—	$10^{-2} \times$ R
C_S	—	—	$10^{-4} \times$ R
N_c^{s1}	—	—	200
N_c^{s2}	—	—	800
β	—	—	100

Table 3: Comparison among CBFO-H, BFO, PSO, and GA on 10-D problems.

10D		BFO	CBFO-S	CBFO-H	PSO	GA
f_1	Best	6.0597	0	0	0	$3.1142e-004$
	Worst	30.4956	0	0	0	0.0345
	Mean	19.6331	0	0	0	0.0118
	Std	5.4090	0	0	0	0.0094
f_2	Best	8.9081	0.1645	0	0.0196	7.8064
	Worst	62.2445	0.5903	$2.5879e-007$	4.0735	9.8140
	Mean	12.0991	0.3492	$1.4813e-007$	0.7294	8.6634
	Std	9.5047	0.0966	$4.6694e-008$	1.4964	0.5557
f_3	Best	9.9505	1.1011	0	2.9849	3.3034
	Worst	53.7285	9.0694	0.0975	25.8689	7.0136
	Mean	36.3513	4.8844	0.0111	10.8450	4.9512
	Std	10.5818	1.6419	0.0257	4.5956	1.1516
f_4	Best	35.5694	0	0.1385	27.1332	0.1679
	Worst	132.1065	0.1328	0.4401	79.7406	0.6318
	Mean	99.7775	0.0647	0.2702	62.9737	0.3439
	Std	24.8497	0.0308	0.0846	12.2581	0.1538

The total numbers of chemotactic steps (or iterations) were set to be 1000. Table 3 lists the experimental results (including the best, worst, mean, and standard deviation of the function values found in 30 runs) for each algorithm on functions $f_1 \sim f_4$. If the average optimum value or error is less than 10^{-10}, it is shown as 0 in the table. Figure 4 shows the search progresses of the average values found by all algorithms over 30 runs for functions $f_1 \sim f_4$.

From the results, we observe that the proposed two CBFO algorithms achieved significantly better performance on all benchmark functions than the original BFO algorithm. CBFO-S surpasses all other algorithms on function 1, which is the unimodal function that

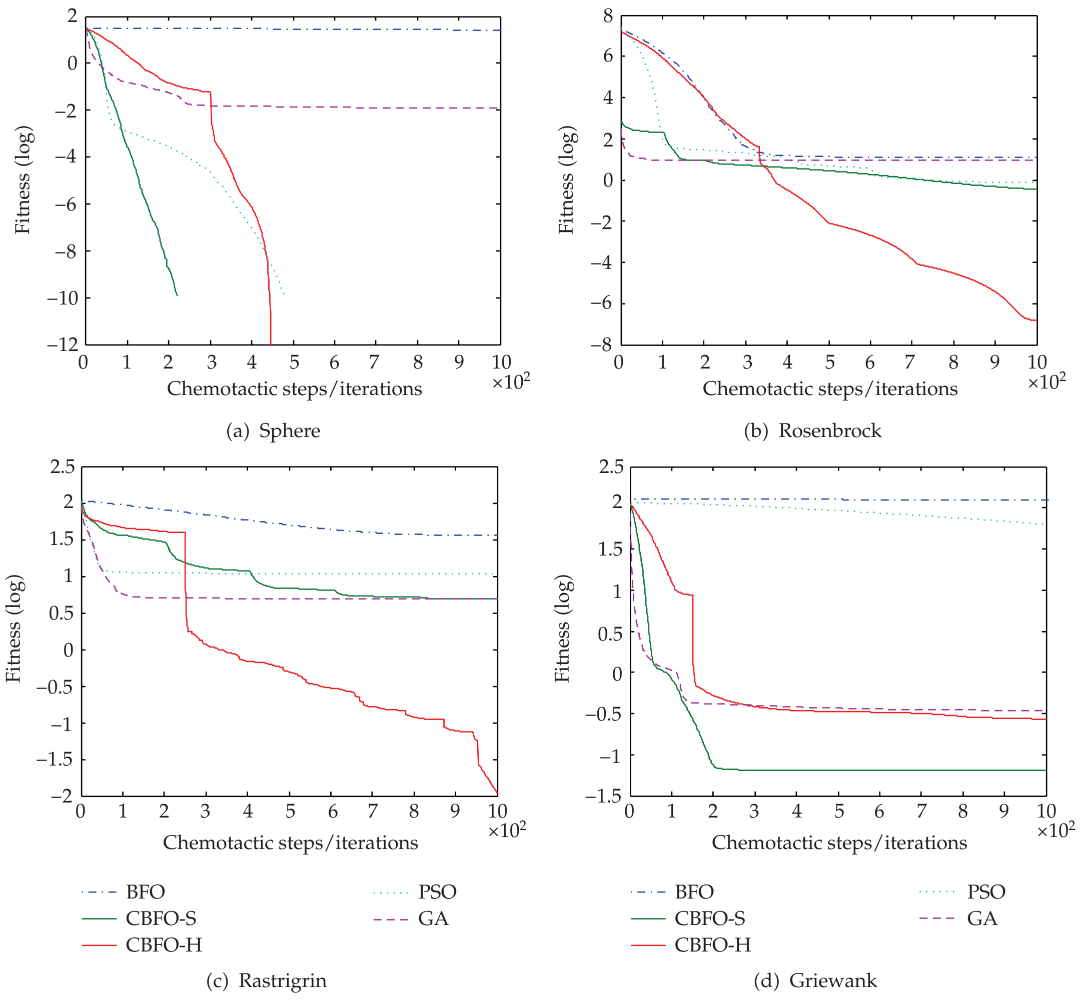
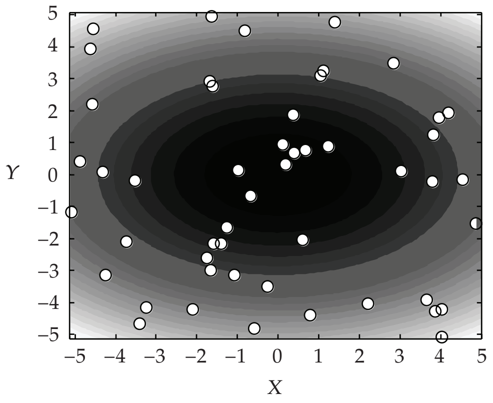


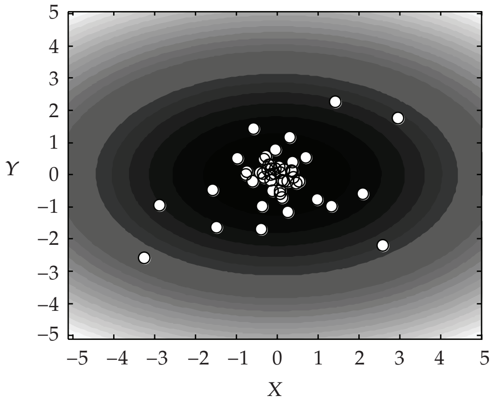
Figure 4: Convergence results of all algorithms.

adopted to assess the convergence rates of optimization algorithms. The CBFO-H performs better on all multimodal functions when the other algorithms miss the global optimum basin. That is, CBFO-H successfully avoids falling into local optima and continues to find better results even after the PSO, GA, and BFO seem to have stagnated. The CBFO-S achieved almost the same performance as the CBFO-H on multimodal functions $f_2 \sim f_4$. The Griewank function is a good example.

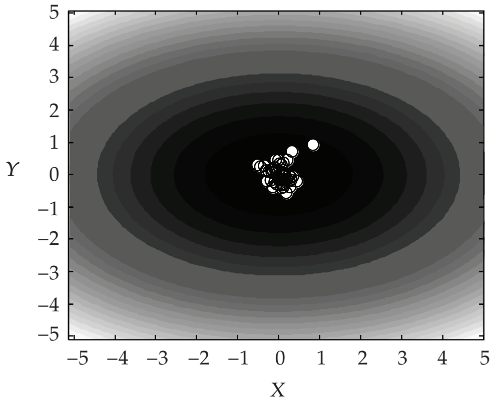
As we can see in Figure 4, in CBFO model, under the influence of the serial heterogeneous cooperative approach, the bacteria colony starts exploring the search space at the initial phase. That is, the bacterial colony do not waste much time before finding the promising regions that contains the local optima because of the large run-length unit C, which encourages long-range exploration. In the succeeding phases, by decreasing C, the bacteria slow down near the optima to pursue the more and more precise solutions. The search performance of the algorithms tested here is ordered as CBFO-S = CBFO-H > PSO = GA > BFO.



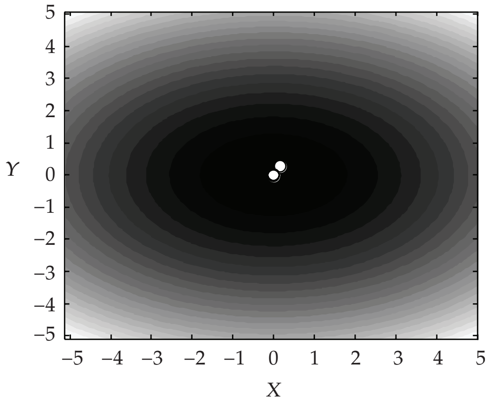
(a) $t = 0$



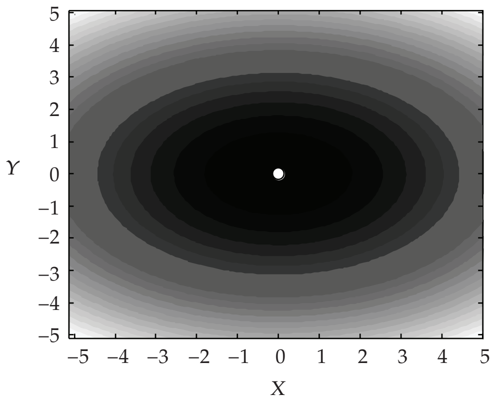
(b) $t = 100$



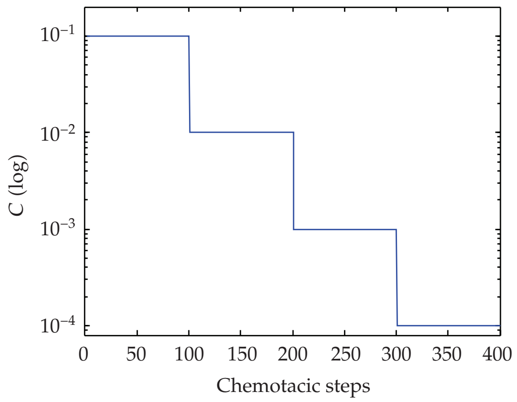
(c) $t = 200$



(d) $t = 300$



(e) $t = 400$

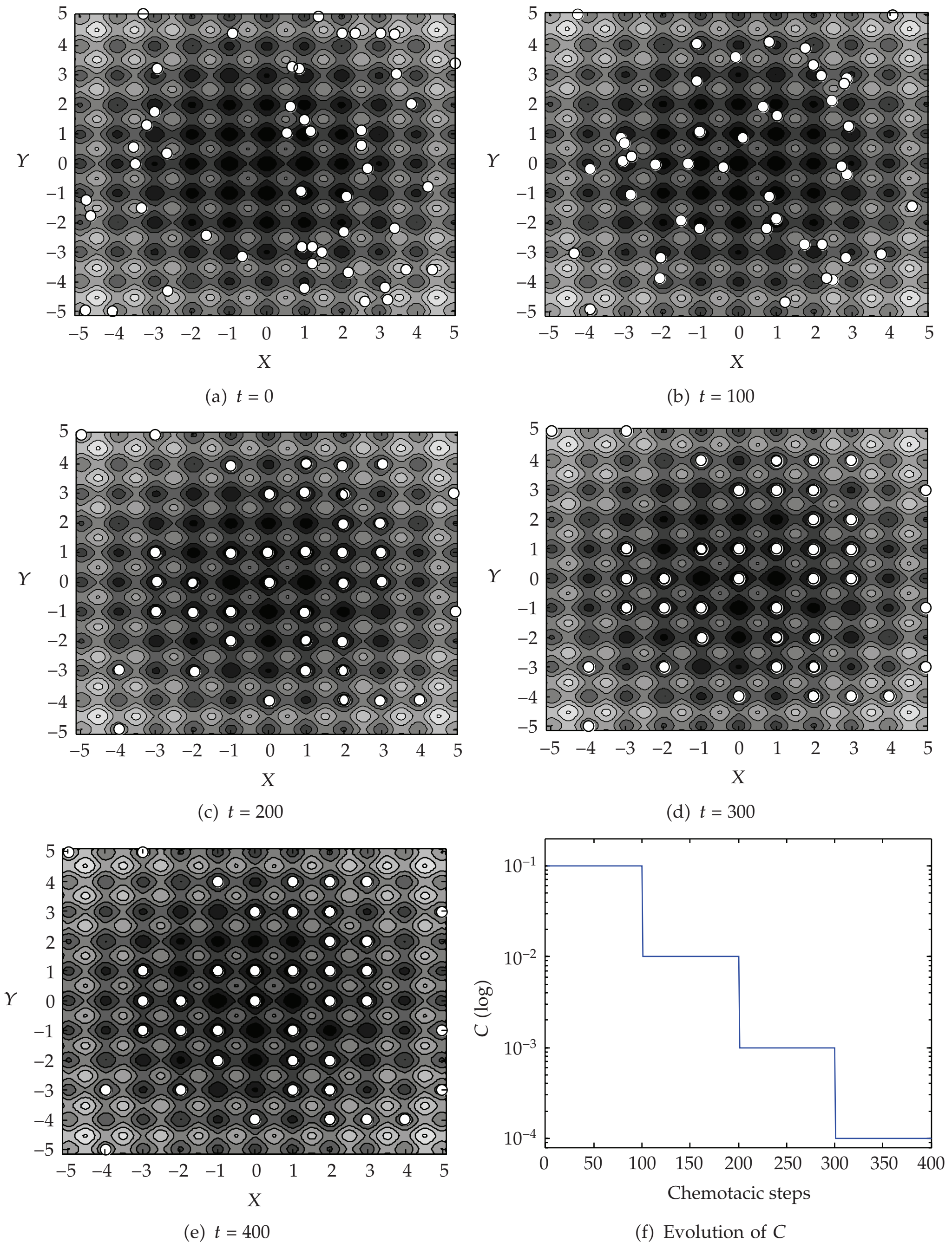


(f) Evolution of C

Figure 5: Population evolution of CBFO on 2D Sphere function in 400 chemotactic steps.

5.4. Bacterial Behaviors in CBFO Model

In order to further analyze the cooperative foraging behaviors of the proposed CBFO model, we run two simulations based on CBFO-S algorithm. In both simulations, we excluded the reproduction, elimination, and dispersion events to illustrate the bacterial behaviors



(a) $t = 0$

(b) $t = 100$

(c) $t = 200$

(d) $t = 300$

(e) $t = 400$

(f) Evolution of C

Figure 6: Population evolution of CBFO on 2D Rastrigrin function in 400 chemotactic steps.

clearly. In both cases, it shows the positions of the bacterial colony on certain chemotactic steps, where each white circle represents a bacterium. The evolution process proceeds 400 chemotactic steps, and we choose $S = 50, N_s = 4, N_c = 100, N_p = 4, C_{initial} = 0.1$, and $\alpha = 10$.

In the first simulation, the population evolution of the CBFO-S was simulated on 2D Sphere function, which is illustrated in Figure 5. Initially, in Figure 5(a), we see that the bacteria colony is distributed randomly over the nutrient map defined by the 2D Sphere function. In Figure 5(b), we can observe that, at the end of the first phase, all the bacterial colony members have found the basin that contains the global optimum of Sphere function and move around it. In the second and third phases (Figures 5(c) and 5(d)), the bacterial colony reinitialized in this basin and then exploit the global optimum. In the final phase (Figure 5(e)), we can observe that all the bacteria have converged to the global optimum of Sphere function. In Figure 5(f), we have also drawn the associated evolution of the run-length unit C of this bacterial colony along its search in the fitness landscape. This provides an intuitive explanation of what is happening during the search of the proposed cooperative bacterial foraging algorithm.

We found a similar pattern in the second simulation shown as in Figure 6, where the bacteria colony pursue the valleys and avoid the peaks of the multimodal 2D Rastrigrin function. In the first phase (Figures 6(a) and 6(b)), starting from their random initial positions, the bacterial colony explore many regions of the nutrient map defined by Rastrigrin function. In the second and third phases (Figures 6(c) and 6(d)), the bacterial colony find many local optima of Rastrigrin function, including the global optimum. Then, in the final phase (Figure 6(e)), we can observe that all the bacteria have converged to several local optima and the global optimum of Rastrigrin function. The associated evolution of the run-length unit C is shown as in Figure 6(f).

6. Conclusions

This paper applied the cooperative approaches to the Bacterial Foraging Optimization (BFO) and proposed the Cooperative Bacterial Foraging Optimization (CBFO) model with two variants, namely CBFO-S and CBFO-H, which can be classified into the serial heterogeneous cooperation on the implicit space decomposition level and the serial heterogeneous cooperation on the hybrid space decomposition level, respectively. These cooperative approaches used here resulted in a significant improvement in the performance of the original Bacterial Foraging Optimization algorithm in terms of convergence speed, accuracy, and robustness.

Four widely used benchmark functions have been used to test the CBFO algorithms in comparison with the original BFO, the stand PSO, and the real-coded GA. The simulation results are encouraging. The CBFO-S and CBFO-H are definitely better than the original BFO for all the test functions and appear to be comparable with the PSO and GA.

There are ways to improve our proposed algorithms. The further research efforts should focus on the tuning of the user-defined parameters for CBFO algorithms based on extensive evaluation on many benchmark functions and real-world problems. Moreover, the self-adaptive mechanisms for the parameters of CBFO may be worthy to undertake to obtain some additional improvements (e.g., which can remove the need of specifying a particular parameter setting of CBFO for a particular problem).

Acknowledgments

This work was supported by the National 863 plans projects of China under Grant 2006AA04A124 and the National 863 plans projects of China under Grant 2006AA04A117.

References

[1] H. J. Bremermann and R. W. Anderson, “An alternative to back-propagation: a simple rule of synaptic modification for neural net training and memory,” Tech. Rep. PAM-483, Center for Pure and Applied Mathematics, University of California, San Diego, Calif, USA, 1990.

[2] S. Müeller, J. Marchetto, S. Airaghi, and P. Koumoutsakos, “Optimization based on bacterial chemotaxis,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 16–29, 2002.

[3] K. M. Passino, “Biomimicry of bacterial foraging for distributed optimization and control,” *IEEE Control Systems Magazine*, vol. 22, pp. 52–67, 2002.

[4] D. H. Kim and J. H. Cho, “Adaptive tuning of PID controller for multivariable system using bacterial foraging based optimization,” in *Proceedings of the 3rd International Atlantic Web Intelligence Conference (AWIC ’05)*, vol. 3528 of *Lecture Notes in Computer Science*, pp. 231–235, Lodz, Poland, June 2005.

[5] S. Mishra, “A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation,” *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 1, pp. 61–73, 2005.

[6] M. Tripathy, S. Mishra, L. L. Lai, and Q. P. Zhang, “Transmission loss reduction based on FACTS and bacteria foraging algorithm,” in *Proceedings of the Parallel Problem Solving from Nature (PPSN ’06)*, pp. 222–231, Reykjavik, Iceland, September 2006.

[7] D. H. Kim and C. H. Cho, “Bacterial foraging based neural network fuzzy learning,” in *Proceedings of the Indian International Conference on Artificial Intelligence*, pp. 2030–2036, Pune, India, December 2005.

[8] J. H. Holland, *Adaptation in Nature and Artificial System*, MIT Press, Cambridge, Mass, USA, 1992.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, New York, NY, USA, 1989.

[10] J. Kennedy and R. C. Eberhart, “Particle swarm optimization,” in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, New York, NY, USA, April 1995.

[11] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.

[12] A. W. Mohemmed and N. C. Sahoo, “Efficient computation of shortest paths in networks using particle swarm optimization and noising metaheuristics,” *Discrete Dynamics in Nature and Society*, vol. 2007, Article ID 27383, 25 pages, 2007.

[13] A. M. Senthil and M. V. C. Rao, “On the performance of the particle swarm optimization algorithm with various inertia weight variants for computing optimal control of a class of hybrid systems,” *Discrete Dynamics in Nature and Society*, vol. 2006, Article ID 79295, 17 pages, 2006.

[14] M. El-Abd and M. Kamel, “A taxonomy of cooperative search algorithms,” in *Proceedings of the 2nd International Workshop on Hybrid Metaheuristics*, vol. 3636 of *Lecture Notes in Computer Science*, pp. 32–41, Barcelona, Spain, August 2005.

[15] J. Adler, “Chemotaxis in bacteria,” *Science*, vol. 153, pp. 708–716, 1966.

[16] M. Potter and K. D. Jong, “A cooperative coevolutionary approach to function optimization,” in *Proceedings of the 3rd Parallel Problem Solving from Nature (PPSN ’94)*, pp. 530–539, Jerusalem, Israel, October 1994.

[17] F. Bergh and A. P. Engelbrecht, “A cooperative approach to particle swarm optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[18] Y. Shi and R. C. Eberhart, “Empirical study of particle swarm optimization,” in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC ’99)*, vol. 3, pp. 1945–1950, Piscataway, NJ, USA, 1999.

[19] S. Sumathi, T. Hamsapriya, and P. Surekha, *Evolutionary Intelligence: An Introduction to Theory and Applications with Matlab*, Springer, New York, NY, USA, 2008.