3

Bacterial Foraging Optimization

Sonam Parashar
Department of Electrical Engineering
Malaviya National Institute of Technology, Jaipur, India

Nand K. Meena
School of Engineering and Applied Science
Aston University, Birmingham, B4 7ET, United Kingdom

Jin Yang
School of Engineering and Applied Science
Aston University, Birmingham, United Kingdom

Neeraj Kanwar
Department of Electrical Engineering
Manipal University Jaipur, Jaipur, India

CONTENTS

3.1 Introduction

The bacterial foraging optimization (BFO) is a swarm intelligence based optimization method introduced by Kevin Passino in 2002. The method is inspired

by the foraging theory of animals and mathematically models the foraging behavior of the bacteria present in the human intestine, named as *E. coli*, for the optimization purpose [1]. The foraging theory represents the idea that every animal or bacteria searches for food or nutrients in such a way that maximizes their energy intake per unit time spent on foraging subjected to some geographical and physiological constraints. The algorithm is further modified and some modifications in algorithm structure are suggested for the simplification of the algorithm such as elimination of the swim step parameter, a clear adaptation rule for the step size, the use of a uniform distribution, the position initialization, the inclusion of best individual information in the movement equation and the removal of cell-to-cell communication [2]. An adaptive BFO dynamically adjusts the run-length unit parameter to balance the trade-off between exploration and exploitation of the search space [3]. BFO can optimize the single objective functions efficiently.

However, for the multi-objective functions, a multi-objective variant of BFO is also introduced. The multi-objective bacteria foraging algorithm (MBFO) gives the pareto optimal solutions by taking decisions based on the idea of integration between the health sorting approach and pareto dominance mechanism [4]. In addition, a hybrid approach known as bacterial foraging algorithm-genetic algorithm (BFO-GA) is also proposed to improve the performance of the algorithm for solving the multi-objective optimization problems [5]. This chapter presents the insights of the BFO algorithm and is organized as follows: Section 3.1 presents the introduction of BFO followed by its algorithm in Section 3.2. In Section 3.3, the pseudo-codes of the BFO algorithm are presented. The Matlab source-code is given in Section 3.4 followed by some solved examples and conclusions in Sections 3.5 and 3.6 respectively.

3.2 Bacterial foraging optimization algorithm

The *E. coli* bacterium has a structure composed of the plasma membrane, cell wall, and capsule that contains the cytoplasm and nucleoid. The pili (singular, pilus) are used for a type of gene transfer to other *E. coli* bacteria, and flagella (singular, flagellum) are used for locomotion. The bacterium follows the foraging theory, always searching for high nutrient gradient to maximize the energy intake per unit time spent on foraging by using some intelligence. This process is achieved through two types of motions i.e. tumble and run or swim. The bacterium swims or runs in a nutrient gradient area and keep searching for the high nutrient gradient with the help of locomotion in a counterclockwise direction which groups the bacterium. If the bacterium finds no nutrient or noxious substance, the flagellum moves in the clockwise direction which means it pulls on the cell and results in a little displacement of the bacterium with undefined direction known as tumble motion of the bacterium. For this

switching between the two motions tumble and run, firstly, the cell slows down or stops and then starts to move slowly.

The BFO algorithm mainly comprises the four main steps, Chemotaxis, Swarming, Reproduction and Elimination or Dispersal. These steps are explained in the following sections:

3.2.1 Chemotaxis

Chemotaxis is a phenomenon which describes the motion patterns of the bacterium over the nutrient gradient. Generally, the bacterium follows the Brownian motion and chooses its direction of motion for searching the food or nutrient gradient with some self-intelligence of locomotion. The motion of a bacterium at ith iteration can be defined as

$$\theta_i(j+1,k,l) = \theta_i(j,k,l) + C_i\phi_j \tag{3.1}$$

where, θ_i is the location of ith bacterium, j is the index for the chemotaxis step, k is the index for the reproduction steps, l is the index for the elimination and dispersal step, C_i is a unit run length of the bacterium and ϕ_j is the direction angle of the jth step, expressed as

$$\phi_j = \frac{\Delta_i}{\sqrt{\Delta_i^T \cdot \Delta_i}} \tag{3.2}$$

where, Δ_i represents i^{th} value of direction vector in j^{th} chemotactic step, $\Delta_i \epsilon R^P$, R^P is a random vector of P dimension having values between $[-1, 1]$ and is used for tumble.

3.2.2 Swarming

This step is about the communication of each bacterium. The swarm of bacteria communicates by releasing some cell attractants and repellents. The attractant responses are obtained from the serine or aspartate whereas repellent responses are obtained from metal ions such as Ni and Co. The cell-to-cell communication in the presence of attractant and repellent is obtained from the following expression

$$\begin{aligned} J_{cc}(\theta, P(j,k,l)) &= \sum_{i=1}^{S} J_{cc}^{j}(\theta, \theta_i(j,k,l)) \\ = \sum_{i=1}^{S} \Big[&- d_{attract} \cdot \exp\Big(- w_{attract} \sum_{m=1}^{p} (\theta_m - \theta_m^i)^2 \Big) \Big] \\ &+ \Big[h_{repellent} \cdot \exp\Big(- w_{repellent} \sum_{m=1}^{p} (\theta_m - \theta_m^i)^2 \Big) \Big] \end{aligned} \tag{3.3}$$

where, $J_{cc}(\theta, P(j,k,l))$ is the objective function value to be added to the actual objective function, S is the total number of bacteria, P is the number of variables to be optimized which are present in each bacterium,

$\theta = [\theta_1, \theta_2, \theta_3, \ldots, \theta_p]^T$ denotes a point in the P-dimensional search space, θ_m^i is the mth component of the ith bacterium position, $d_{attract}$ is the depth of the attractant released by the cell, $w_{attract}$ is a measure of the width of attractant signal, $h_{repellant}$ is the height of the repellant effect (i.e., $h_{repellant} = d_{attract}$), and $w_{repellant}$ is the measure of the width of the repellant.

3.2.3 Reproduction

In this process, the members of the bacteria's swarm having high nutrients will only survive and reproduce new ones by splitting themselves into two. For this purpose, the population first sorted into ascending order according to the fitness of the objective function, the members responsible for poor fitness of the objective function will be considered as the members with nutrient deficiency or the weak members which will further be replaced by the newly reproduce members from the healthy bacteria responsible for the best fitness of the objective function. This replacement balances the number of bacteria members in the swarm.

3.2.4 Elimination and dispersal

After some reproduction steps, elimination and dispersal take place to avoid the local minima. At this stage, each bacterium will be assigned a probability P_{ed}. The bacterium is then selected according to their probability to change their location in the search space. The bacterium having highest probability will change its location first.

3.3 Pseudo-code of bacterial foraging optimization

In this section, the pseudo-codes for the basic version of BFO is presented in Algorithm 2.

Algorithm 2 Pseudo-code of basic Bacterial Foraging Optimization.

1: Determine the objective function $OF(.)$
2: set the algorithm parameters such as p=dimensions; N_c=chemotactic steps; N_s=swimming length; S=bacterium; N_{ed}=elimination; N_r=reproduction steps; p_{ed}=elimination probability etc.
3: set the lower and upper bounds $[LB, UB]$ for each variable/dimension p
4: randomly generate initial population of bacteria, as follows
5: **for** each k-th reproduction step **do**
6: **for** each j-th chemotactic step **do**
7: **for** each i-th bacterium **do**
8: **for** each d-th dimension **do**

$pp(d,k,j,i) = LB(d) + rand * (UB(d) - LB(d))$
end for
apply correction algorithm to i-th bacterium, if infeasible
calculate the fitness value of i-th bacterium
end for
end for
end for
Start the elimination or dispersal step...
for each l-th elimination-dispersal step **do**
for each k-th reproduction step **do**
for each j-th chemotactic step **do**
for each i-th bacterium **do**
Update the i-th bacterium by using (3.2) and (3.3), as
$pp_i(j+1,k,l) = pp_i(j,k,l) + C_i \cdot \frac{\Delta_i}{\sqrt{\Delta_i^T \cdot \Delta_i}}$
apply correction algorithm to i-th bacterium, if infeasible
calculate the fitness value of newly generated bacterium i
end for
apply the swimming operation for Ns bacteria
apply reproduction-elimination by sorting bacteria according to their fitness values and then split the top 50% and eliminate the remaining
end for
end for
end for
return the best result.

3.4 Matlab source-code of bacterial foraging optimization

The Matlab source-code of basic BFO is presented in Listing 3.1.

```
% Bacterial Foraging Optimization Algorithm...
clc;
clear;
%Algorithm parameters....
N_bact=60;          %Total number of bacteria,
N_die=round(N_bact/2); %Number of weak bacteria died during
    reproduction,
N_chemo=30;         %The number of chemotactic steps,
Ns=7;               %The swimming length,
N_repro=10;         %The number of reproduction steps,
N_disp=5;           %The number of elimination-dispersal events,
P_disp=0.5;         %Elimination-dispersal probability,
LB=[-3*pi -3*pi];   %lower bounds of variables,
UB=[3*pi 3*pi];     %upper bounds of variables,
p=length(UB);       %number of variables or dimensions,
C=1;                %unit length step size for a run/tumble
%-------------------------------------------------------------------
%initialization, random but feasible population
for r=1:N_repro     %reproduction starts...
for c=1:N_chemo     %chemotaxis starts...
```

```
for b=1:N_bact        %individual bacteria...
for v=1:p             %variable....
pp(v,b,c,r)=LB(v)+rand*(UB(v)-LB(v));
end
Fit(b,c,r)=OF(pp(:,b,c,r));
end
end
end
best_fit=min(Fit(:));
loc=find(best_fit==Fit);
best_sol=pp(:,loc);
%%%-----------------------------------------------------
%%dispersal starts here....
for d=1:N_disp
for r=1:N_repro       %reproduction starts...
for c=1:N_chemo       %chemotaxis starts...
for b=1:N_bact        %individual bacteria...
FitLast=Fit(b,c,r);
dd(:,b)=unifrnd(-1,1,[p,1]);
%add on the cell-to-cell attractant
pp(:,b,c,r)=pp(:,b,c,r)+C*dd(:,b)/sqrt(dd(:,b)'*dd(:,b));
%New fitness calculations....
Fit(b,c,r)=OF(pp(:,b,c,r));
%swimming behaviour of bacteria
sw = 1;
while sw < Ns
if Fit(b,c,r)<FitLast
%addonthecell-to-cell attractant
pp(:,b,c,r)=pp(:,b,c,r)+C*dd(:,b)/sqrt(dd(:,b)'*dd(:,b));
Fit(b,c,r)=OF(pp(:,b,c,r)); %New fitness calculations....
FitLast=Fit(b,c,r); %Restore the fittest bacterium..
else
sw=Ns;
end
sw=sw+1;
end
end
end
%Reprodution...
Fit_health=sum(Fit(:,:,r),2); %Health of bacteria, 1 to N_bact
[Health,loc]=sort(Fit_health);
%spliting the top (N_bact/2) bacteria with better fitness
g=0;                  %counter for spliting bateria...
for sp=N_die+1:N_bact
g=g+1;
pp(:,loc(sp),1,r)=pp(:,loc(g),1,r);
Fit(loc(sp),1,r)=OF(pp(:,loc(sp),1,r));
end
end
%elimination/dispersal operations...
for u=1:N_bact
rr=rand;
if rr<P_disp
for v=1:p    %variable....
pp(v,u,1,1)=LB(v)+rand*(UB(v)-LB(v));
end
Fit(u,1,1)=OF(pp(:,u,1,1));
end
end
% Preserve the best...
best_fit2=min(Fit(:));
if best_fit2<best_fit
loc2=find(best_fit2==Fit);
best_sol=pp(:,loc2);
best_fit=best_fit2;
end
end
```

```
%print the solution
disp(best_fit);
disp(best_sol);
```

Listing 3.1
Source-code of BFO in Matlab.

3.5 Numerical examples

In this section, the application of the BFO algorithm is demonstrated by a step-by-step example.

Example 2 *Determine the global minima of the Griewank function expressed as*

$$f(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) \tag{3.4}$$

where, $x\epsilon[-3\pi, 3\pi]$

Solution: A contour plot of the Griewank function is presented in Fig. 3.1. It shows that this function is non-convex with multiple minima.

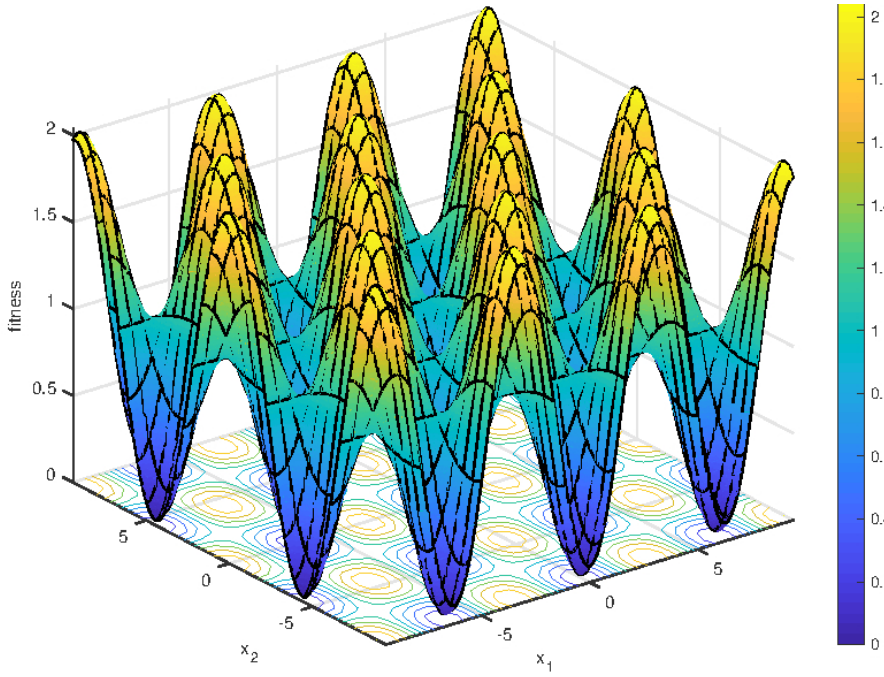


FIGURE 3.1
Griewank function for n=2.

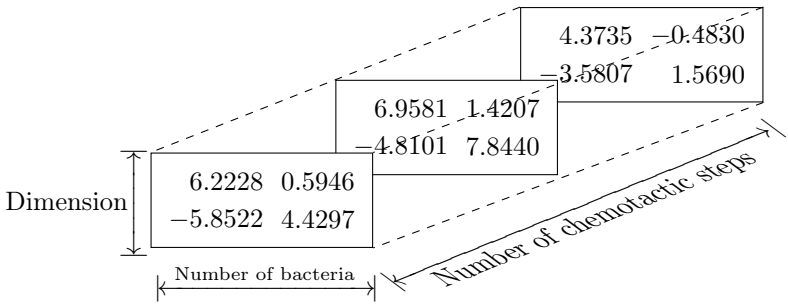
In Listing 3.2, the Matlab source code is presented for the Griewank function, $f(x)$.

```
%Griewank function
function [Fit] = OF(x)
n=length(x);
sum=0;
product=1;
for i=1:n
y=x(i);
sum=sum + y^2/4000;
product=product*cos(y/sqrt(i));
end
Fit=1+sum-product;
```

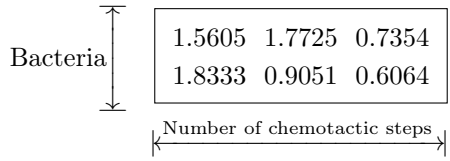
Listing 3.2
Definition of Griewank function $OF(.)$ in Matlab.

Now, we demonstrate how BFO can help to determine the optimal values of x such that $f(x)$ is minimum. In the first step, assume that $n = 2$, upper and lower limits [LB UB], i.e., [-3π 3π] along with all algorithm parameters are provided. For example, $N_{bact} = 2$, $N_{die} = round(N_{bact}/2)$, $N_{chemo} = 3$, $N_s = 3$, $N_{repro} = 1$, $N_{disp} = 1$, $P_{disp} = 0.5$, $p = length(UB)$, $C = 1$ etc.

In step two, a random population of $N_{bact} = 2$ bacteria, with $n = 2$ dimensions, are generated for each chemotactic step, as shown below.



In the third step, determine the fitness of each bacterium by using $OF(.)$. For example, the first bacterium of the first chemotactic step is $x(:, 1, 1) = [6.2228 \ -5.8522]^T$; thus its fitness value will be $OF(x(:, 1, 1)) = 1.5605$. The second bacterium of the first chemotactic step is $x(:, 2, 1) = [0.5946 \ 4.4297]^T$ and its fitness $OF(x(:, 2, 1)) = 1.8333$. Similarly, fitness values of all bacteria in chemotactic steps two and three are determined and presented below.



In the fourth step, we retain the best bacterium based on fitness value. In the proposed work, we are minimizing objective function $OF(.)$; therefore a bacterium with minimum fitness value will be the best. For example, minimum fitness value is 0.6064 which corresponds to the bacterium $[-0.4830\ 1.5690]^T$ in the third chemotactic step so $x_{best} = [-0.4830\ 1.5690]^T$ with $Fit_{best} = 0.6064$.

In the fifth step, if the maximum number of reproduction steps is reached then jump to the sixteenth step; otherwise move to step five.

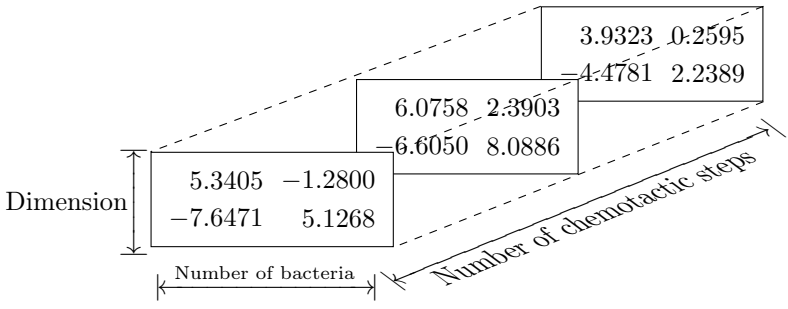
In the sixth step, we store the fitness of bacterium $x(:, 1, 1)$ or '1' into $FitLast = OF(x(:, 1, 1)) = 1.5605$ and a random vector Δ_1 is generated for bacterium '1', e.g., $\Delta_1 = [-0.4462\ -0.9077]^T$.

In the seventh step, we update the position of the first bacterium by using (3.1) as $x(:, 1, 1) = x(:, 1, 1) + C \cdot \frac{\Delta_1}{\sqrt{\Delta_1^T \cdot \Delta_1}} = [5.7816\ -6.7496]^T$.

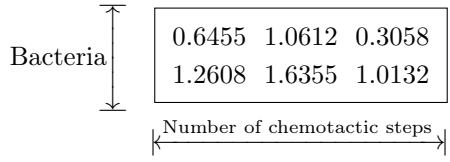
In the eighth step, we determine the fitness value of the updated bacterium by using $OF(.)$, i.e., $OF(x(:, 1, 1)) = 0.9669$.

In the ninth step, the swimming behaviour of the bacterium is executed. To do this, a swimming counter is set, i.e., $sw = 1$. While $sw < N_s$ then $FitLast$ is compared with $OF(x(:, 1, 1))$. If $OF(x(:, 1, 1)) < FitLast$ then $x(:, 1, 1)$ is further updated as $x(:, 1, 1) = x(:, 1, 1) + C \cdot \frac{\Delta_1}{\sqrt{\Delta_1^T \cdot \Delta_1}} = [5.3405\ -7.6471]^T$, where $OF(x(:, 1, 1)) = 0.6455$. If $OF(x(:, 1, 1)) > FitLast$ then set $sw = N_s$. Similarly, other bacteria of all chemotactic steps are updated and their fitness values are also evaluated, as shown below.

Updated bacteria in all chemotactic steps

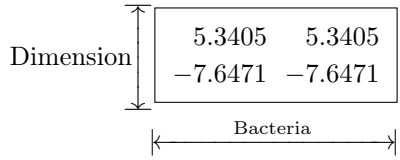


Fitness values of all new bacteria

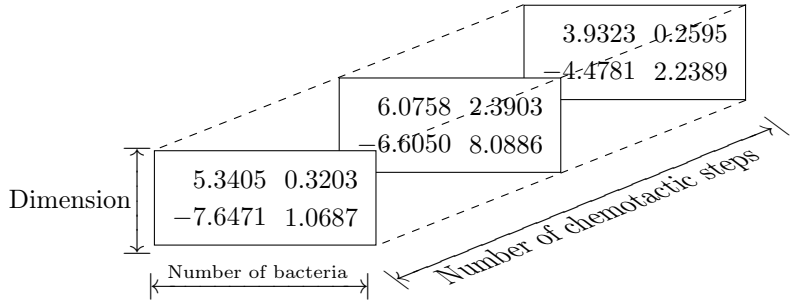


In the tenth step, reproduction is performed. First, we determine the sum of bacteria fitness in all chemotactic steps and then the population is sorted according to the evaluated sum. Assume that sum is called $FitHealth$ and sorted fitness values are referred as $Health$, as demonstrated below.
$FitHealth = [(0.6455 + 1.0612 + 0.3058) \quad (1.2608 + 1.6355 + 1.0132)]^T = [2.0125 \quad 3.9095]^T$ and $Health = [2.0125 \quad 3.9095]^T$.

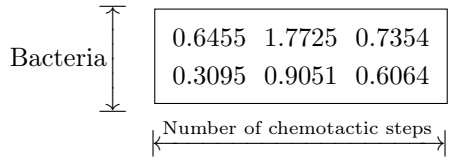
In the eleventh step, we eliminate the bottom $N_{bact}/2$ bacteria and split the top $N_{bact}/2$ bacteria population, with better fitness values, to keep the population constant. From $FitHealth$, it is observed that bacteria two has poor fitness; therefore we eliminate this poor bacteria by replacing with fittest bacteria, i.e. bacteria one, in the first chemotactic step. The updated bacteria of the first chemotactic step are given below; in the second and third chemotactic steps, bacteria will remain unaltered.



In the twelfth step, the elimination-dispersal operation is performed. To do this, a random number, $rr \epsilon [0\ 1]$ is generated for each bacterium. If $rr < P_{disp}$ then that bacterium is randomly updated. For example, $rr = 0.9963$ which is greater than P_{disp} therefore, bacterium one, i.e. $x(:, 1, 1) = [5.3405 \ -7.6471]^T$ will remain unaltered. Similarly, rr is generated for bacterium two, e.g., $rr = 0.3469$ which is less than P_{disp} therefore updated randomly as $x(:, 2, 1) = LB(2) + (UB(2) - LB(2)). * rand(2, 1) = [0.3203 \ 1.0687]^T$. The updated bacteria of the first chemotactic step are shown below; the others remain the same.



In the thirteenth step, we determine the fitness values of updated bacteria and then store in the fitness matrix as



In the fourteenth step, we determine the best bacterium of the new population and then compare with the best of the old one. If the new best is better, we replace with the new; otherwise not. For example, the best fitness of the updated population is $Fit_{best}^{new} = 0.3095$, which corresponds to bacterium $x_{best}^{new} = [0.3203\ 1.0687]^T$. Here we are minimising the function $OF(.)$ and $Fit_{best}^{new} < Fit_{best}$ thus $x_{best} = x_{best}^{new} = [0.3203\ 1.0687]^T$.

In the fifteenth step, we check whether the maximum number of elimination-dispersal events is reached. If yes, move to the sixteenth step; otherwise return to the fifth step.

In the sixteenth step, we print the best solution x_{best} with its corresponding fitness Fit_{best}.

3.6 Conclusions

In this chapter, the standard version of the BFO algorithm, inspired by the foraging behavior of bacteria, is discussed. The pseudo-codes of the algorithm are presented and explained. In order to understand the coding of this optimization method, Matlab source-code is also provided. The applicability of this method to mathematical optimization problems is demonstrated by step-by-step example in which the benchmark Griewank function is minimized by using the BFO algorithm.

3.7 Acknowledgement

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of the United Kingdom (Reference Nos.: EP/R001456/1 and EP/S001778/1).

References

1. K.M. Passino. "Biomimicry of bacterial foraging for distributed optimization and control". *IEEE Control Systems*, vol. 22(3), 2002, pp.52-67.
2. M. A. Munoz, S. K. Halgamuge, W. Alfonso and E. F. Caicedo. "Simplifying the Bacteria Foraging Optimization Algorithm". *IEEE Congress on Evolutionary Computation*, Barcelona, 2010, pp. 1-7. doi: 10.1109/CEC.2010.5586025

3. H. Chen, Y. Zhu, and K. Hu. "Adaptive bacterial foraging optimization". *Abstract and Applied Analysis, Hindawi*, vol. 2011. doi:10.1155/2011/108269
4. B. Niu, H. Wang, J. Wang, and L. Tan. "Multi-objective bacterial foraging optimization". *Neurocomputing*, vol. 116, 2013, pp.336-345.
5. P. Manikandan and D. Ramyachitra. "Bacterial Foraging Optimization-Genetic Algorithm for Multiple Sequence Alignment with Multi-Objectives". *Scientific Reports*, vol. 7(1), 2017, Article number: 8833.