THE EFFECTS OF CROSSOVER AND MUTATION RATES ON CHEMOTAXIS DIFFERENTIAL EVOLUTION OPTIMIZATION ALGORITHM

Add your Author(s) NAME
Affiliation

ABSTRACT
The manuscript should not exeed 4000 words. Nature inspired, bacterial foraging optimization algorithm (BFOA), and bio inspired, differential evolution (DE), have been employed to solve complex search optimization problems. Researchers have been investigating the performance of different DE parameters (crossover rate and mutation factor) in solving optimization problems. In the present paper, the performance of a hybrid technique called Chemotaxis Differential Evolution Optimization Algorithm (CDEOA) which hybridizes BFOA with DE using different crossovers and mutation rates is reported along with the impact their combinations have on CDEOA in terms of exploration and exploitation of the population. In the present investigation, 6 unimodal and multimodal benchmark functions were involved.

***Keywords*:** bacterial, foraging, optimization,

1. INTRODUCTION

Differential Evolution (DE) and its variants with adapted parameters have been employed to solve the real-world optimization problems. Recent studies (Zaharie 2002; Jingqiao and Sanderson, 2009; Qin *et a*l., 2009) have fostered the fine-tuning the parameters of DE. So far, Bacterial Foraging Optimization Algorithm (BFOA)has been successfully applied in the area of optimal control design (Passino, 2002), harmonic estimation (Mishra, 2005), transmission loss reduction (Tripathy *et al*, 2006).

The CDEOA's (hybrid technique of BFOA) behavior on different DE parameter pairs such as mutation and crossover rate has been reported (Yıldız and Altun, 2015). BFOA mimics the chemotaxis behaviors of a bacterium, whereas DE employs the evolutionary operators, i.e. mutation, crossover, and selection. CDEOA hybridizes the aforementioned techniques in such a way that if a bacterium fails to explore the food, the behavior of the algorithm turns out to be explorative and, if it discover the nutrient-rich areas, the algorithm acts as an exploitative fashion.

Bacterial foraging optimization algorithm (BFOA)

The mechanism of Passino (2002) consisting of chemotaxis, reproduction, and elimination-dispersal is here introduced briefly. Figure 1 depicts a flowchart of BFOA adapted from (Dasgupta, 2009). A pseudo code of BFOA is embedded in the Algorithm 1.

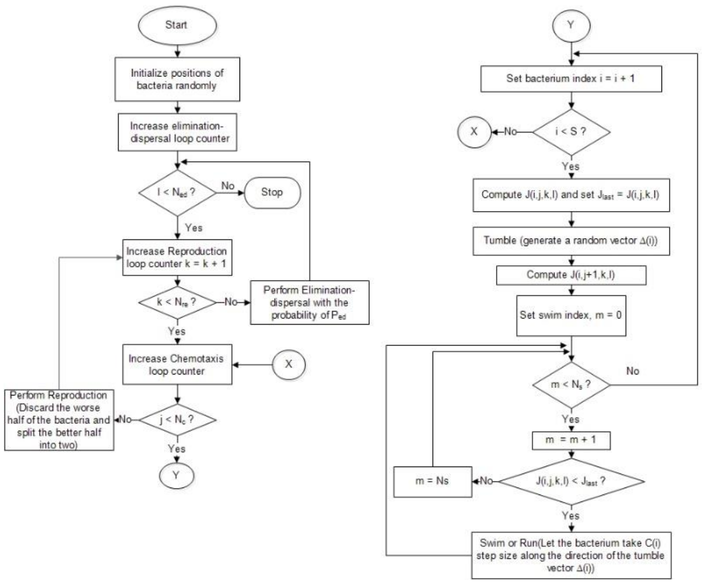

Start
Initialize positions of bacteria randomly
Increase elimination-dispersal loop counter
$l < N_{ed}$?
No
Stop
Yes
Increase Reproduction loop counter k = k + 1
$k < N_{re}$?
No
Perform Elimination-dispersal with the probability of P_{ed}
Yes
Increase Chemotaxis loop counter
X
Perform Reproduction (Discard the worse half of the bacteria and split the better half into two)
No
$j < N_c$?
Yes
Y
Y
Set bacterium index i = i + 1
X
No
i < S ?
Yes
Compute J(i,j,k,l) and set J_{last} = J(i,j,k,l)
Tumble (generate a random vector Δ(i))
Compute J(i,j+1,k,l)
Set swim index, m = 0
$m < N_s$?
No
Yes
m = m + 1
m = Ns
No
J(i,j,k,l) < J_{last} ?
Yes
Swim or Run(Let the bacterium take C(i) step size along the direction of the tumble vector Δ(i))

Fig. 1: A flowchart of BFOA.

Chemotaxis

An *E.coli* bacterium makes *tumble* and *run* steps in succession via flagella in its lifetime. The *tumble* is the random direction of a swim, whereas *swim* is the successive step in the same direction. $\theta(i, j, k, l)$represents the position of the ith bacterium at jth chemotactic, kth reproductive, and lth elimination-dispersal step. Eq. 1 and 2 refer to the position of a bacterium in the following steps:

$$\vec{t}(j) = \frac{\Delta(i)}{\sqrt{\Delta^T(i) * \Delta(i)}} \tag{1}$$

$$\theta(i, j+1, k, l) = \theta(i, j, k, l) + C(i) * \vec{t}(j) \tag{2}$$

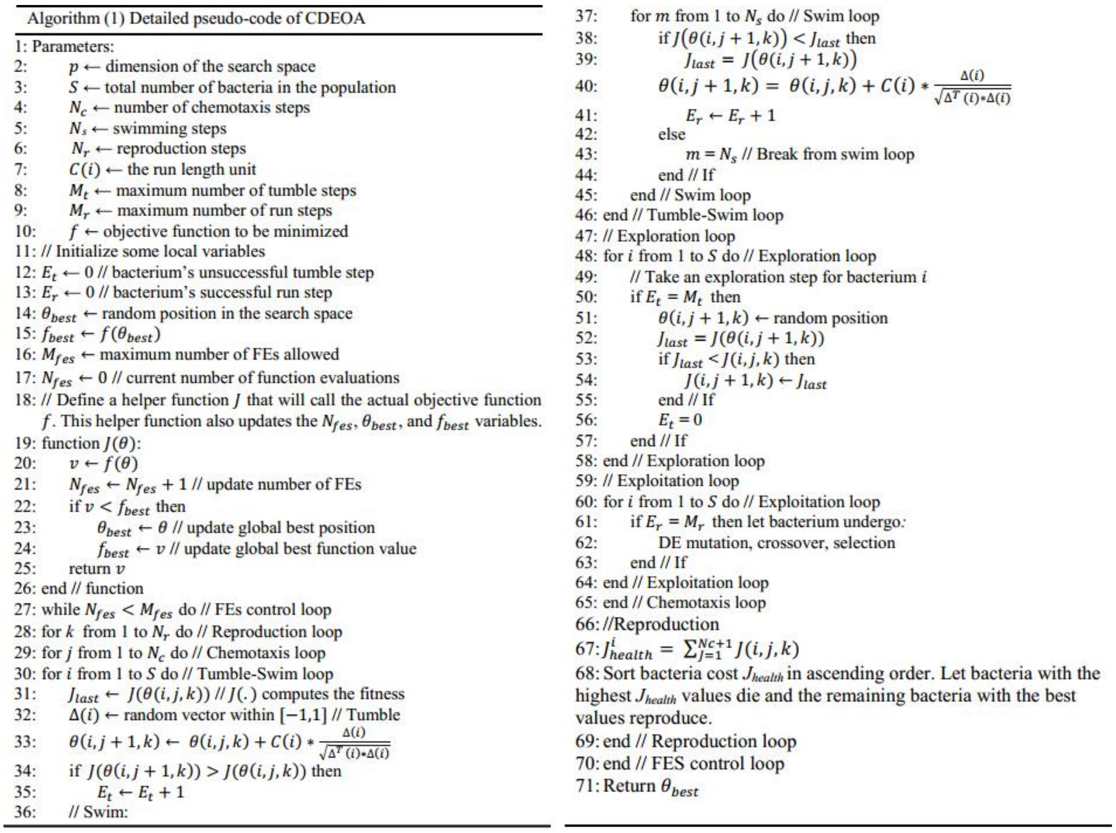
where $C(i)$ is a constant,$\vec{t}(j)$, Eq. 1 refers to the direction of the jth step, and $\Delta(i)$ is a random vector whose elements vary from [-1, 1].

Reproduction

The cost function values, health of a bacterium are accumulated in the life-time of a bacterium. Based on each bacterium's health, the bacteria in the swarm were classified from the lowest (the healthiest ones) to highest cost. The population members, which have lowest health, were split into two bacteria and placed at the same positions while the other bacteria were not considered.

Chemotaxis differential evolution optimization algorithm (CDEOA)

CDEOA is a hybrid population-based nature inspired optimization technique whose driving forces rely on the operators of chemotaxis, reproduction, mutation, crossover, and selection. It employs the local search operator (chemotaxis) from BFOA and global search operators (mutation and crossover) from DE. CDEOA works on the "weak" bacteria to make the algorithm explorative where "weak" bacteria are the individuals in positions with nutrients-poor medium and "strong" bacteria to make the algorithm exploitative where "strong" bacteria are the individuals in positions with nutrients-rich medium (Yıldız and Altun 2015).



```
Algorithm (1) Detailed pseudo-code of CDEOA
1: Parameters:
2:      $p \leftarrow$ dimension of the search space
3:      $S \leftarrow$ total number of bacteria in the population
4:      $N_c \leftarrow$ number of chemotaxis steps
5:      $N_s \leftarrow$ swimming steps
6:      $N_r \leftarrow$ reproduction steps
7:      $C(i) \leftarrow$ the run length unit
8:      $M_t \leftarrow$ maximum number of tumble steps
9:      $M_r \leftarrow$ maximum number of run steps
10:     $f \leftarrow$ objective function to be minimized
11: // Initialize some local variables
12: $E_t \leftarrow 0$ // bacterium's unsuccessful tumble step
13: $E_r \leftarrow 0$ // bacterium's successful run step
14: $\theta_{best} \leftarrow$ random position in the search space
15: $f_{best} \leftarrow f(\theta_{best})$
16: $M_{fes} \leftarrow$ maximum number of FEs allowed
17: $N_{fes} \leftarrow 0$ // current number of function evaluations
18: // Define a helper function $J$ that will call the actual objective function
    $f$. This helper function also updates the $N_{fes}$, $\theta_{best}$, and $f_{best}$ variables.
19: function $J(\theta)$:
20:     $v \leftarrow f(\theta)$
21:     $N_{fes} \leftarrow N_{fes} + 1$ // update number of FEs
22:     if $v < f_{best}$ then
23:         $\theta_{best} \leftarrow \theta$ // update global best position
24:         $f_{best} \leftarrow v$ // update global best function value
25:     return $v$
26: end // function
27: while $N_{fes} < M_{fes}$ do // FEs control loop
28: for $k$ from 1 to $N_r$ do // Reproduction loop
29: for $j$ from 1 to $N_c$ do // Chemotaxis loop
30: for $i$ from 1 to $S$ do // Tumble-Swim loop
31:     $J_{last} \leftarrow J(\theta(i,j,k))$ // $J(.)$ computes the fitness
32:     $\Delta(i) \leftarrow$ random vector within $[-1,1]$ // Tumble
33:     $\theta(i,j+1,k) \leftarrow \theta(i,j,k) + C(i) * \frac{\Delta(i)}{\sqrt{\Delta^T(i) \cdot \Delta(i)}}$
34:     if $J(\theta(i,j+1,k)) > J(\theta(i,j,k))$ then
35:         $E_t \leftarrow E_t + 1$
36:     // Swim:
37:     for $m$ from 1 to $N_s$ do // Swim loop
38:         if $J(\theta(i,j+1,k)) < J_{last}$ then
39:             $J_{last} = J(\theta(i,j+1,k))$
40:         $\theta(i,j+1,k) = \theta(i,j,k) + C(i) * \frac{\Delta(i)}{\sqrt{\Delta^T(i) * \Delta(i)}}$
41:             $E_r \leftarrow E_r + 1$
42:         else
43:             $m = N_s$ // Break from swim loop
44:         end // If
45:     end // Swim loop
46: end // Tumble-Swim loop
47: // Exploration loop
48: for $i$ from 1 to $S$ do // Exploration loop
49:     // Take an exploration step for bacterium $i$
50:     if $E_t = M_t$ then
51:         $\theta(i,j+1,k) \leftarrow$ random position
52:         $J_{last} = J(\theta(i,j+1,k))$
53:         if $J_{last} < J(i,j,k)$ then
54:             $J(i,j+1,k) \leftarrow J_{last}$
55:         end // If
56:         $E_t = 0$
57:     end // If
58: end // Exploration loop
59: // Exploitation loop
60: for $i$ from 1 to $S$ do // Exploitation loop
61:     if $E_r = M_r$ then let bacterium undergo:
62:         DE mutation, crossover, selection
63:     end // If
64: end // Exploitation loop
65: end // Chemotaxis loop
66: //Reproduction
67: $J^i_{health} = \sum_{j=1}^{Nc+1} J(i,j,k)$
68: Sort bacteria cost $J_{health}$ in ascending order. Let bacteria with the
highest $J_{health}$ values die and the remaining bacteria with the best
values reproduce.
69: end // Reproduction loop
70: end // FES control loop
71: Return $\theta_{best}$
```

Algorithm 1 Detailed pseudo-code of CDEOA

If the bacterium explores a nutrient-rich medium and carries on running for a predetermined M_r number of consecutive generations, this bacterium will enter exploitation state which is the use of the mutation, crossover, and selection operators of DE (line 60 in Algorithm 1). If the bacterium's current cost remains fixed for a predefined number M_t of consecutive generations, then this bacterium will enter exploration state which is its liquidation (line 48 in Algorithm 1). Yıldız and Altun (2015) reported that the balance between exploration and exploitation of search space is established due to these two strategies. The pseudocode of CDEOA is presented in Algorithm 1.

Experimental study

The performance of CDEOA on different mutation and crossover rate pairs were tested using a set of 6 standard benchmark functions. Functions 1 and 2 are unimodal and functions from 3 to 6 are multimodal functions. Table 1 describes the benchmark functions used in the experiments. CDEOA/rand/1 implies the algorithm which employs DE/rand/1 mutation strategy (Eq. 3) whereas CDEOA/best/1 implies DE/best/1 (Eq. 4).

Tab. 1 Description of benchmark functions used. D: dimensionality of the functions

No	Name	Formula	Search space
f_1	Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	(-2,2)
f_2	Schwefel 2.21	$f_2(x) = max\{\lvert x_i \rvert, 1 < i < D\}$	(-2,2)
f_3	Rosenbrock	$f_3(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 2)^2]$	(-2,2)

Comparison of five mutation and crossover rate paired techniques based on DE/rand/1 mutation strategy

DE/rand/1 (Eq. 3), one of the most used mutation strategy, is characterized by a slow convergence speed. In addition, it exhibits much stronger exploration ability as three individuals acting in distinct search space information out of the current population are randomly chosen. Here, the aforementioned mutation strategy empowers the exploitation ability. In contrast, it slows down the exploitation ability of an individual (Qin, *et al*., 2009).

- Sphere 9.27E-09 9.02E-09 1.77E-06 3.31E-03 9.09E-09
- Schwefel 2.21 1.59E-02 1.03E-02 4.29E+002.53E+019.41E+00
- Rosenbrock 2.57E+012.37E+012.15E+013.68E+012.23E+01

1. Ackley 1.66E+001.66E+001.66E+001.67E+001.66E+00
2. Rastrigin 1.69E+013.22E+001.75E+004.00E+013.30E+00
3. Griewank4.93E-04 8.89E-09 4.19E-08 6.54E-03 8.91E-09

Unimodal functions

In these two unimodal functions in Table 2, [0.5, 0.9] and [0.5, 0.5] exhibit superior performance to the other $[F,CR]$ pairs. Even though the CRs are different in each pair, they end up with the similar results. We can infer that F = 0.5 and CR which is within the range of [0.5, 0.9] yield better results in terms of quality of final solution. [0.1, 0.9] is unable to perform well in two unimodal problems because F which is close to 0.0 has tendency to lead less exploration ability in searching the search space.

2. CONCLUSIONS

CDEOA/best/1 exhibits better performance than CDEOA/rand/1 in sphere. CDEOA/rand/1 outperforms the CDEOA/best/1 in Shwefel 2.21 showing that the behavior of CDEOA depends on unimodal functions. In multimodal functions, CDEOA/best/1 is generally better than CDEOA/rand/1 in Rosenbrock with the exception of [0.1, 0.1] and [0.8, 0.2] pairs, so these exceptions have a great impact in the success of CDEOA/best/1. Regardless the different $[F, CR]$ pairs, both CDEOA/rand/1 and CDEOA/best/1 possess identical results in Ackley. CDEOA/rand/1 exhibits in Rastrigin and Griewank better performance than CDEOA/best/1. We can infer that the results are problem specific. In particular¸ $F \in [0.1, 0.5]$ tends to yield better performance while CR is equal to 0.5. Generally speaking, there are no common parameter settings for all the problems. Rather, there is an optimum parameter values for each problem after fine-tuning experiments. The Python source code of the CDEOA can be found in (https://sites.google.com/site/yeyildiz12/).

REFERENCES

Dasgupta S, Das S, Abraham A, Biswas A. 2009. Adaptive Computational Chemotaxis in Bacterial Foraging Optimization: An Analysis. IEEE Transactions on Evolutionary Computation 13:919–941. doi: 10.1109/TEVC.2009.2021982.

Jingqiao Z, Sanderson AC. 2009. JADE: Adaptive Differential Evolution with Optional External Archive. IEEE Transactions on Evolutionary Computation 13:945–958. doi: 10.1109/TEVC.2009.2014613

Mishra S. 2005. A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation. Evolutionary Computation, IEEE Transactions on 9, 61–73.