IMPROVED CHEMOTAXIS DIFFERENTIAL EVOLUTION OPTIMIZATION ALGORITHM

Y. Emre Yıldız[1], Oğuz Altun[2], A. Osman Topal[3]

[1]Epoka University, Tirana, Albania, yeyildiz12@epoka.edu.al
[2]Yıldız Technical University, Istanbul, Turkey, oaltun@yildiz.edu.tr
[3]Epoka University, Tirana, Albania, aotopal@epoka.edu.al

ABSTRACT. The social foraging behavior of Escherichia coli has recently received great attention and it has been employed to solve complex search optimization problems. This paper presents a modified bacterial foraging optimization BFO algorithm, ICDEOA (Improved Chemotaxis Differential Evolution Optimization Algorithm), to cope with premature convergence of reproduction operator. In ICDEOA, reproduction operator of BFOA is replaced with probabilistic reposition operator to enhance the intensification and the diversification of the search space. ICDEOA was compared with state-of-the-art DE and non-DE variants on 7 numerical functions of the 2014 Congress on Evolutionary Computation (CEC 2014). Simulation results of CEC 2014 benchmark functions reveal that ICDEOA performs better than that of competitors in terms of the quality of the final solution for high dimensional problems.

Keywords: bacterial foraging optimization algorithm (BFOA), differential evolution (DE), computational chemotaxis, hybrid optimization, improved chemotaxis differential evolution optimization algorithm (ICDEOA)

INTRODUCTION

Nature and natural living forms have been investigated by several researchers in order to get insight into solving complex real-world problems for a few decades. Natural selection has a tendency to get rid of living organisms with poor foraging strategies and supports the spread of genes of living organisms with successful foraging. These evolutionary concepts have guided Kevin M. Passino to propose a new bio-inspired optimization method known as the bacterial foraging optimization algorithm (BFOA) which attempts to maximize the energy intake per unit time (Passino, 2002; Liu et al., 2002). In order to increase the BFOA success, a number of improvements have been carried out on hybridization with evolutionary algorithms (EA) (Kim et al., 2007; Biswas et al., 2007a; Biswas et al., 2007b). So far, BFOA has been successfully applied in applications in optimal control design (Passino, 2002), harmonic estimation (Mishra, 2005).

In this paper, a modified approach named ICDEOA is introduced. ICDEOA improves the optimization performance of CDEOA (Yıldız et al., 2015). CDEOA incorporates two strategies into the chemotaxis step of BFOA: weak bacterium's search and strong bacterium's foraging. The weak bacteria's explorative capability is boosted by randomly moving to new positions whereas the strong bacteria's exploitation capability is boosted by incorporating the ideas of differential evolution (DE) operators (Yıldız et al., 2015). In our contribution, in place of reproduction operator of BFOA, we employed the *probabilistic repositioning* operator which acts based on the bacterium's cost. If the cost of a bacterium is high, the bacterium most likely will change its position. If the cost is low, the bacterium is moved to the vicinity of the best bacterium.

Reproduction operator of CDEOA possesses intensive exploitation capability which may result in premature convergence since it chooses the best of the population and kills the rest for the next generation. By incorporating *probabilistic repositioning* operator into CDEOA, we prevent not only the premature convergence problem of CDEOA, but also diversify the half of the population. The simulation results reveal that ICDEOA has shown superior performance in unimodal and multimodal functions in terms of the quality of final solution.

CLASSICAL BACTERIAL FORAGING OPTIMIZATION ALGORITHM (BFOA)

Passino (2002) establishes the bacterial foraging system on three mechanisms such as chemotaxis, reproduction, and elimination-dispersal. Below, we define these processes.

Chemotaxis

An *E.coli* bacterium makes successive *tumble* and *swim* steps via flagella. The *tumble* specifies 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. The equations Eq. (1) and Eq. (2) represent the position of a bacterium in the next step,

$$\vec{t}(j) = \frac{\Delta(i)}{\sqrt{\Delta^T(i) * \Delta(i)}} \tag{5}$$

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

where $C(i)$ is a fixed predefined length of the unit walk, $\vec{t}(j)$. Eq. (1) defines the direction of the jth step, and $\Delta(i)$ generates a random vector whose elements are between [-1, 1].

Reproduction

During a bacterium's life-time, the objective function values which is health of a bacterium are added to each other. Depending on each bacterium's health, all the bacteria in the population are ranked from the lowest health (the healthiest ones) to largest health. The healthiest population 50% which has less function value asexually split into two bacteria and then are placed at the same positions. The rest of the bacteria 50% with poor health is thrown away to keep the number of population fixed.

Elimination and dispersal

In case that the bacteria are subjected to gradual or sudden changes such as significant rise of temperature or sudden flow of water, elimination and dispersal events may take place. BFOA mimics these events in such a way that some bacteria are liquidated randomly with a predetermined fixed probability value while the new replacements are randomly initialized over the search space.

DIFFERENTIAL EVOLUTION (DE)

Differential evolution (DE) is an evolutionary optimization technique which employs mutation, crossover, and selection operators to solve complex problems. For each generation G, a new population is generated from the current population, $x_{i,G}|i = 1,2,\dots,N$ where N is the size of the population. The initial population is randomly generated according to a uniform distribution. After initialization of the population, DE undergoes mutation, crossover, and selection operators (Storn et al., 1997).

Mutation

At each generation G, a mutant vector $v_{i,G}$ is generated for each target vector $x_{i,G}, i = 1,2,\dots,N$ in the current population. The one of the mutation strategies (Eq. (3)) is as follows:

- "DE/best/1"

$$v_{i,G} = x_{best,G} + F * (x_{r_1,G} - x_{r_2,G}) \tag{3}$$

where $x_{best,G}$ is the best vector in the current generation G, r_1 and r_2 are different integers from each other which are randomly chosen from the current population and are different from ith index, and F is the mutation scaling factor which usually ranges [0,1+].

Crossover

After mutation process, the target vector $x_{i,G}$ is combined with the mutated vector $v_{i,G}$ and a new trial vector $u_{i,G} = (u_{i,1,G}, u_{i,2,G}, \dots, u_{i,D,G})$ is generated by Eq. (4):

$$u_{i,j,G} = \begin{cases} v_{i,j,G}, & if\ R_j(0,1) \le C_r\ or\ j = j_{rand}, \\ x_{i,j,G}, & if\ R_j(0,1) > C_r, \end{cases} \tag{4}$$

where $j = 1,2,\dots,D,\ j_{rand}$ is a randomly chosen integer within the range of [1,D], $R_j(0,1)$ is uniform random number between 0 and 1 for each j, and $C_r \in [0,1]$ is the predetermined crossover rate parameter.

Selection

The selection process selects the better of the parent vector $x_{i,G}$ and the trial vector $u_{j,G}$. In a minimization problem, the selected parent vector in the next generation is given by Eq. (5),

$$x_{i,G+1} = \begin{cases} u_{i,G}, & if\ f(u_{i,G}) < f(x_{i,G}), \\ x_{i,G}, & otherwise, \end{cases} \tag{5}$$

where $f(\cdot)$ is the cost function. If the trial vector $u_{i,G}$ yields a better cost function value than $x_{i,G}$, it replaces its parent in the next generation; or the parent is retained in the search space.

ICDEOA (IMPROVED CDEOA)

The concept of ICDEOA depends on two approaches: a) making "weak" bacteria more diversified, where "weak" bacteria are the ones in positions with nutrient-poor medium, and b) making "strong" bacteria more intensified, where "strong" bacteria are the ones in positions with nutrient-rich medium.

Based on the aforementioned approaches, we introduce a new operator, *probabilistic repositioning*, which balances the exploration and the exploitation trade-off. The reproduction process of classical BFOA is replaced with *probabilistic repositioning* operator. Unlike the reproduction process, *probabilistic repositioning* operator retains the strong bacteria in the vicinity of the best bacterium, whereas the weak bacteria are dispersed to the random positions in the search space.

```
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:     $N_r \leftarrow$ repositioning 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 // Reposition 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:    $\theta(i,j+1,k) \leftarrow \theta(i,j,k) + C(i) * \frac{\Delta(i)}{\sqrt{\Delta^T(i) * \Delta(i)}}$
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: //Repositioning of all bacteria
67:    prob ← Assign probabilities of each bacterium.
68:    for i, e in enumerate (prob)
69:        if e > random number within [0,1]
70:            $\theta(i,j+1,k) \leftarrow$ to a random position
71:        else
72:            $\theta(i,j+1,k) \leftarrow$ to the vicinity of the best bacterium
73: end // FEs control loop
74: Return $\theta_{best}$
```

Figure 1. Pseudocode of ICDEOA

In our contribution, in addition to the ideas of CDEOA, all bacteria are graded according to their cost function values at the end of the each chemotaxis step. The bacterium with high scaled cost which is far away from the global optimum will be dispersed to a random position to make the search process more diversified (line 69-70 in Figure 1). On the other hand, the bacterium with low scaled cost which is close to the global optimum will approach to the best bacterium's vicinity to make the search process more focused (line 72 in Figure 1).

EXPERIMENTAL STUDY

The study introduced in this paper aims to test the quality of the final solution at the end of a fixed number of function evaluations (FEs). The maximum number of FEs was set to 3×10^5 for 30-D functions with the population size $S = 50$. The error function values were given in the Table 1 according to $(F(x) - F(x^*))$ for evaluating the success of five algorithms, where $\vec{x}$ is the best value of the bacterium in a run and $\vec{x}^*$ is the global best of the test function. "Mean Error" and "Std Dev" in Table 1 indicate the average and the standard deviation of the error values obtained in 25 runs.

The CEC'14 test functions are as follows: F_1=Rotated high conditioned Elliptic, F_2=Rotated Bent Cigar, F_3=Rotated discus, F_4= Shifted and rotated Rosenbrock, F_5=Shifted and rotated Ackley, F_6=Shifted and rotated Weierstrass, and F_7=Shifted and rotated Griewank (Liang et al., 2013).

Comparison with three State-of-the-art DE and one non-DE

The performance of the ICDEOA algorithm was compared with OptBees (Maia et al. 2013), FERDE (Qu et al. 2014), RSDE (Xu et al. 2014), and CDEOA (Yıldız et al. 2015).

1) Unimodal Functions $F_1 - F_3$.

As presented in Table 1, overall, ICDEOA is better than that of four methods on these three unimodal functions. It outperforms OptBees on 2, FERDE on 2, RSDE on 1, and CDEOA on 2 test functions. In contrast, FERDE and RSDE perform better than ICDEOA on test function F_1 . ICDEOA also exhibits similar performance with OptBees, RSDE, and CDEOA on test function F_2 .

2) Multimodal Functions $F_4 - F_7$.

On these four multimodal test functions, ICDEOA outperforms OptBees on 3, FERDE on 2, RSDE on 2, and CDEOA on 2 test functions. FERDE, RSDE, and CDEOA exhibit better performance than ICDEOA on test function F_7. Overall, ICDEOA performs better than OptBees, FERDE, RSDE, and CDEOA.

Table 1. Comparison of OptBees, FERDE, RSDE, CDEOA, and ICDEOA

Functions		OptBees Mean Error Std Dev		FERDE Mean Error Std Dev		RSDE Mean Error Std Dev		CDEOA Mean Error Std Dev		ICDEOA Mean Errror Std Dev
Unimodal Functions	F_1	8.57E+04 6.96E+02	-	5.41E+02 6.40E+02	+	1.50E+03 1.70E+03	+	4.64E+04 4.26E+04	-	7.32E+03 2.45E+03
	F_2	0.00E+00 6.40E-02	≈	2.39E03 3.24E-03	-	**0.00E+00** **0.00E+00**	≈	**0.00E+00** **0.00E+00**	≈	**0.00E+00** **0.00E+00**
	F_3	8.41E03 2.94E+00	-	1.13E03 7.36E-04	-	4.74E02 1.16E-01	-	1.05E05 6.68E-05	-	1.20E07 3.38E-07
Simple Multimodal Functions	F_4	1.64E+01 2.88E+00	-	6.25E01 1.46E+00	-	3.05E+00 1.34E+01	-	4.46E+00 2.40E+01	-	1.49E02 1.27E-02
	F_5	2.00E+01 1.29E-04	≈	2.00E+01 7.17E-05	≈	2.03E+01 9.88E-02	-	2.00E+01 5.42E-06	≈	2.00E+01 2.00E-03
	F_6	1.64E+01 1.28E+00	-	1.82E+01 1.72E+00	-	5.16E+00 2.01E+00	≈	6.30E+00 2.33E+00	-	5.36E+00 1.86E+00
	F_7	3.75E02 1.40E-01	-	0.00E+00 0.00E+00	+	8.46E04 1.59E-03	+	5.99E03 8.69E-03	+	8.66E03 1.34E-02
-		5		4		3		4		
+		0		2		2		1		
≈		2		1		2		2		

"-", "+", and "≈" indicates that the success of the corresponding technique is worse than, better than, and similar to ICDEOA, respectively.

CONCLUSION

ICDEOA, proposed in this paper, aims to tackle with the premature convergence problem of reproduction operator of CDEOA. We employed *probabilistic repositioning* operator to balance the intensification and diversification of search space. ICDEOA was compared with three state-of-the-art DE counterparts, i.e., FERDE, RSDE, and CDEOA and one non-DE counterpart. Simulation results show that overall performance of ICDEOA[1] was superior to, or comparable to, that of the four competitors.

REFERENCES

Biswas, A., Dasgupta, S., Das, S., & Abraham, A. (2007a). A synergy of differential evolution and bacterial foraging optimization for global optimization. *Neural Network World*, 17(6), 607.

[1] The Python source code of the ICDEOA can be accessed from Y. Emre Yıldız's page: https://sites.google.com/site/yeyildiz12/

Biswas, A., Dasgupta, S., Das, S., & Abraham, A. (2007b). Synergy of PSO and bacterial foraging optimization—a comparative study on numerical benchmarks. In *Innovations in Hybrid Intelligent Systems* (pp. 255–263). Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-74972-1_34

Kim, D. H., Abraham, A., & Cho, J. H. (2007). A hybrid genetic algorithm and bacterial foraging approach for global optimization. *Information Sciences*, 177(18), 3918–3937.

Liang, J. J., Qu, B. Y., & Suganthan, P. N. (2013). Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. Computational Intelligence Laboratory. Retrieved from http://web.mysites.ntu.edu.sg/PublicSite/Shared Documents/CEC-2014/Definitions of CEC2014 benchmark suite Part A.pdf

Liu, Y., & Passino, K. M. (2002). Biomimicry of Social Foraging Bacteria for Distributed Optimization: Models, Principles, and Emergent Behaviors. *Journal of Optimization Theory and Applications*, 115(3), 603–628. http://doi.org/10.1023/A:1021207331209

Maia, R. D., Castro, L. N. de, & Caminhas, W. M. (2013). OptBees-A Bee-Inspired Algorithm for Solving Continuous Optimization Problems. In *BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC)*, 142–151. IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6855842

Mishra, S. (2005). A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation. *IEEE Transactions on Evolutionary Computation*, 9(1), 61–73.

Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems*, 22(3), 52–67.

Qu, B. Y., Liang, J. J., Xiao, J. M., & Shang, Z. G. (2014). Memetic differential evolution based on fitness Euclidean-distance ratio. In *IEEE Congress on Evolutionary Computation (CEC)*, 2266–2273. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber= 6900476

Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4), 341–359.

Tripathy, M., Mishra, S., Lai, L. L., & Zhang, Q. P. (2006). Transmission loss reduction based on FACTS and bacteria foraging algorithm. In *Parallel Problem Solving from Nature-PPSN IX* (pp.222–231).Springer. Retrieved from http://link.springer.com/chapter/10.1007/11844297_23

Xu, C., Huang, H., & Ye, S. (2014). A Differential Evolution with Replacement Strategy for Real-Parameter Numerical Optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, 1617–1624. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber= 6900468

Yıldız, Y. E., & Altun, O. Hybrid Achievement Oriented Computational Chemotaxis in Bacterial Foraging Optimization: A Comparative Study on Numerical Benchmark. *Soft Computing*. Springer. http://doi.org/10.1007/s00500-015-1687-4