INTERNATIONAL SCIENTIFIC AND VOCATIONAL JOURNAL (ISVOS JOURNAL)

Vol.: 5 Issue: 2 Date: 31.12.2021 Received: 01.12.2021 Accepted: 13.12.2021 Final Version: 31.12.2021

ISVOS Journal, 2021, 5(2): 164-170 - https://doi.org/10.47897/bilmes.1031011

Opposite Based Crow Search Algorithm for Solving Optimization Problems

Burhanettin DURMUŞ[a,1]

[a,1]Kütahya Dumlupınar Üniversitesi, Faculty of Engineering, Electrical-Electronics Engineering Department, Kütahya, Türkiye
ORCID ID: 0000-0002-8225-3313

Abstract

This article proposes an opposite based learning (OBL) enhanced crow search algorithm (CSA) version for solving optimization problems. The proposed method, named the opposite based CSA (ObCSA), starts searching with individuals with higher fitness in the initial phase of the evolutionary process. In this way, it is aimed to improve the convergence performance of the basic CSA. To validate the proposed method, a set of benchmark test suit of different of features is chosen. Its convergence characteristic and statistical results are compared with the basic CSA. The results obtained show that the proposed method improves the convergence performance of the basic CSA. And the statistical results indicate that it manages to reach the near optimal solution and increases the quality of the solution.

Keywords: "Crow search algorithm, opposite based learning, optimization"

1. Introduction

Over the past 20 years, behavioral patterns of swarm-based organisms have inspired the development of many optimization methods. While the teachings of living things individually and in the swarm determine their behavioral strategies in maintaining their vital activities, these strategies are modeled and turn into an effective search tool in reaching the optimal solution of a problem. Ant colony optimization (ACO), particle swarm optimization (PSO), artificial bee colony (ABC), lion optimization algorithm (LOA) and cuckoo search algorithm (CS) are the most famous of these [1-5]. These metaheuristics generally have similar evolutionary processing phases, although the behavioral patterns of the swarm or creature they represent may differ. They use computational phases such as the random distribution of search agents in the search space, the creation of new agents, random selections, and the continuation of the search through iterations.

On the other hand, various approaches to improve the search skills and performance of metaheuristics are adapted to their processing phases. The approaches such as dynamic population [6], weighted accelerations [7], chaotic mapping [8] are the preferred methods. One of the places where performance improvement is made is the initialization phase. In this phase, search agents are randomly distributed to the search space. They are evaluated according to their fitness value and transferred to the next processing phases. However, since their initial distribution is random, individuals with low fitness also participate in the computational process. Instead, starting the evolution with individuals with high fitness in the initial phase will undoubtedly accelerate the global solution and increase the convergence performance of the algorithm. In this context, one of the suggested approaches is the opposite-based learning strategy (OBL) [9]. This strategy evaluates the opposite position in the search space of a generated position. Because a solution based on opposite location may be closer to the global solution than a randomly generated solution [9]. The results indicating that the OBL approach increases the algorithm performance are presented in various studies in the literature. To improve the performance of the PSO, the OBL approach is applied to the PSO and it is reported to produce good results [10]. Wang et al. [11] are modified the harmony search algorithm (HSA) with OBL and applied to global optimization problems. The obtained results show that the developed approach improves the convergence performance and the solution quality. In another study, Dinkar and Deep [12] are developed the antlion optimizer (ALO) version using OBL. The performance was observed by applying the developed method to the control systems. Zhao et al. [13] are derived the OBL-ABC algorithm using the OBL approach. It is noted that the derived algorithm is suitable for solving complex functions. In addition, it has been presented

[1] Sorumlu Yazar. Tel.: +90- 274 - 443 -4223-4223
E-posta adresi: burhanettin.durmus@dpu.edu.tr

in other studies that OBL improves algorithm performance [14-16]. As a result, the OBL approach speeds up the search process and improves the convergence performance.

The CSA algorithm is one of the new metaheuristics developed on the intelligence behavior of crows [17]. It manipulates the crows' ability to hide and remember their food. It is a population-based metaheuristic that uses random selection and has few self-parameters. In this paper, the OBL strategy is adapted to the CSA algorithm to improve its convergence performance. Two ObCSA methods are derived by integrating different OBL approaches into CSA. The developed methods are applied to high-dimensional test functions and their performances are compared.

2. Basic CSA

The CSA is a new metaheuristic inspired by the intelligence behavior of crows [17]. It mimics the behavior of crows to hide and find their food. Its evolution process starts with the random distribution of crows in the search space. For an *n*-dimensional problem, the initialization phase is defined as:

$$x_{i,ite\_number} = \left[x^1_{i,ite\_number}, x^2_{i,ite\_number}, \ldots x^n_{i,ite\_number} \right] \tag{1}$$

Here, x_{i,ite_number} represents the position of the i^{th} crow and the *ite_number* represents the number of iterations.

Each crow position is evaluated according to the objective function and their fitness values are calculated. This fitness value actually represents the solution quality of the crow. All positions visited by the crows during the search process are recorded in their own memory as in Eq. (2).

$$M_{i,ite\_number} = \left[m^1_{i,ite\_number}, m^2_{i,ite\_number}, \ldots m^n_{i,ite\_number} \right] \tag{2}$$

The next position of a crow is generated as in Eq. (3), depending on the awareness probability (*AP*) and flight length (*fl*) parameters. The new location produced is evaluated according to the objective function and the fitness value is calculated. Then, the selection process is made for the new location as in Eq. (4).

$$X_{i,ite\_number+1} = \begin{Bmatrix} x_{i,ite\_number} + r_i . fl.(m_{i,ite\_number} - x_{i,ite\_number}) & r_i > AP \\ random & \text{otherwise} \end{Bmatrix} \tag{3}$$

$$m_{i,ite\_number+1} = \begin{Bmatrix} \text{if} \quad f\left(x_{i,ite\_number+1}\right) > f\left(m_{i,ite\_number}\right) & x_{i,ite\_number+1} \\ \text{else} & m_{i,ite\_number} \end{Bmatrix} \tag{4}$$

Evolutionary operations are continued until the number of maximum iterations (*ite*_max) is reached. The best solution is returned when the stopping criterion is met.

3. Opposite based CSA (ObCSA)

In the classical CSA algorithm, the initialization positions of the crows are randomly distributed over the search space. In this case, the computational process is started with individuals with a lower fitness value. As a result, the convergence performance of the algorithm may decrease. According to the OBL approach proposed by Tizhoosh [9], a solution generated based on opposition is closer to the global solution than a randomly generated solution. In this way, the process of reaching a global solution can be accelerated.

In this paper, two different OBL approaches are used for the proposed opposite based CSA algorithms. In the first one (ObCSA-1), half of the population is randomly generated, while the other half is determined as the opposing positions of these individuals.

For the second algorithm (ObCSA-2), the initial individuals are generated both randomly and with their opposite positions. Afterwards, they are selected according to their fitness values and the position with the better fit is included in the initialization phase. Thus, the evolutionary calculation begins with individuals with higher fitness values. The pseudocode of the proposed ObCSA methods is shown in Figure 1.

4. Simulation Results

In this section, the proposed ObCSA methods are applied to the benchmark set given in Table 1 to demonstrate their performance. These test functions are classified into two groups as unimodal (UM) and multimodal (MM). The first four functions are UM and have a single global optimum. The remaining functions are MM and have too many local optima. The two-dimensional graphs of the test functions used are shown in Figures 2 and 3.

```
**Initialize**
Define the PopSize, ite_max, fl, AP, x_min, x_max and dimension
        //No Opposite
                for m = 1 : PopSize
                        xPop(m,:) = x_min + ( x_max – x_min ) .* rand(1,Dim);
                end
        //Opposite == 1
                Lenght = round(PopSize/2);
                for m = 1 : Lenght
                        xPop(m,:) = x_min + ( x_max – x_min ) .* rand(1,Dim);
                        xPop(m+PopSize-Lenght,:) = x_min + x_max - xPop(m,:);
                end
        //Opposite == 2
                for m = 1 : PopSize
                        x_Temp = x_min + ( x_max – x_min ) .* rand(1,Dim);
                        xOps = x_min + x_max – x_Temp;
                        y_Temp = Objective_Function(x_Temp);
                        yOps = Objective_Function(xOps);
                        if y_Temp <= yOps
                        xPop(m,:) = x_Temp;
                        else
                        xPop(m,:) = xOps;
                        end
ite_number=1
**Evaluate**
yPop(m) = Objective_Function(xPop(m,:));
**Generate** the crow's new position
**While** ite<ite_max
Update the crow's new position according to Eq(3)
Update the memory of crows in Eq(2)
ite_number=ite_number+1
**End**
Return the position of best crow
```

Figure 1. Pseudo code of ObCSA

Table 1. The benchmark set used in experiments

Type	Definition	Name	*S*
UM	$f_1 = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	Schwefel's problem 1.2	$[-100,100]^n$
UM	$f_2 = \sum_{i=1}^{n-1}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	Rosenbrock	$[-30,30]^n$
UM	$f_3 = \sum_{i=1}^{n}\left(\lvert x_i + 0.5\rvert\right)^2$	Step	$[-100,100]^n$
UM	$f_4 = \sum_{i=1}^{n} i x_i^4 + random(0,1)$	Quartic noise	$[-1.28,1.28]^n$
MM	$f_5 = \sum_{i=1}^{n}\left[x_i^2 - 10Cos(2\pi x_i) + 10\right]$	Rastrigin	$[-5.12,5.12]^n$
MM	$f_6 = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	Ackley	$[-32,32]^n$
MM	$f_7 = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Generalized Griewank	$[-600,600]^n$
MM	$f_8 = \frac{1}{10}\left\{\begin{array}{l}\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2\left[1 + \sin^2(3\pi x_{i+1})\right] \\ +(x_n - 1)^2\left[1 + \sin^2(2\pi x_{i+1})\right]\end{array}\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	Penalized2	$[-50,50]^n$

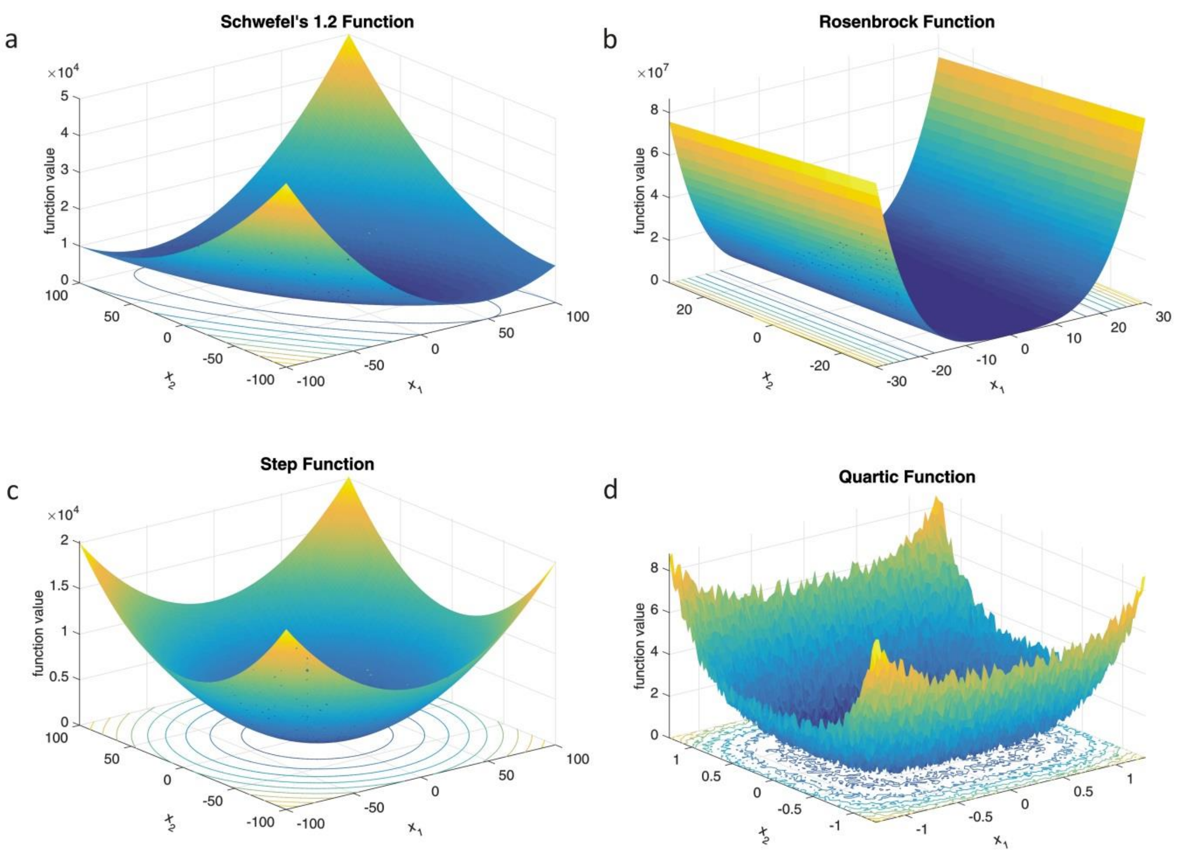

Figure 2. The illustration two-dimensional UM functions

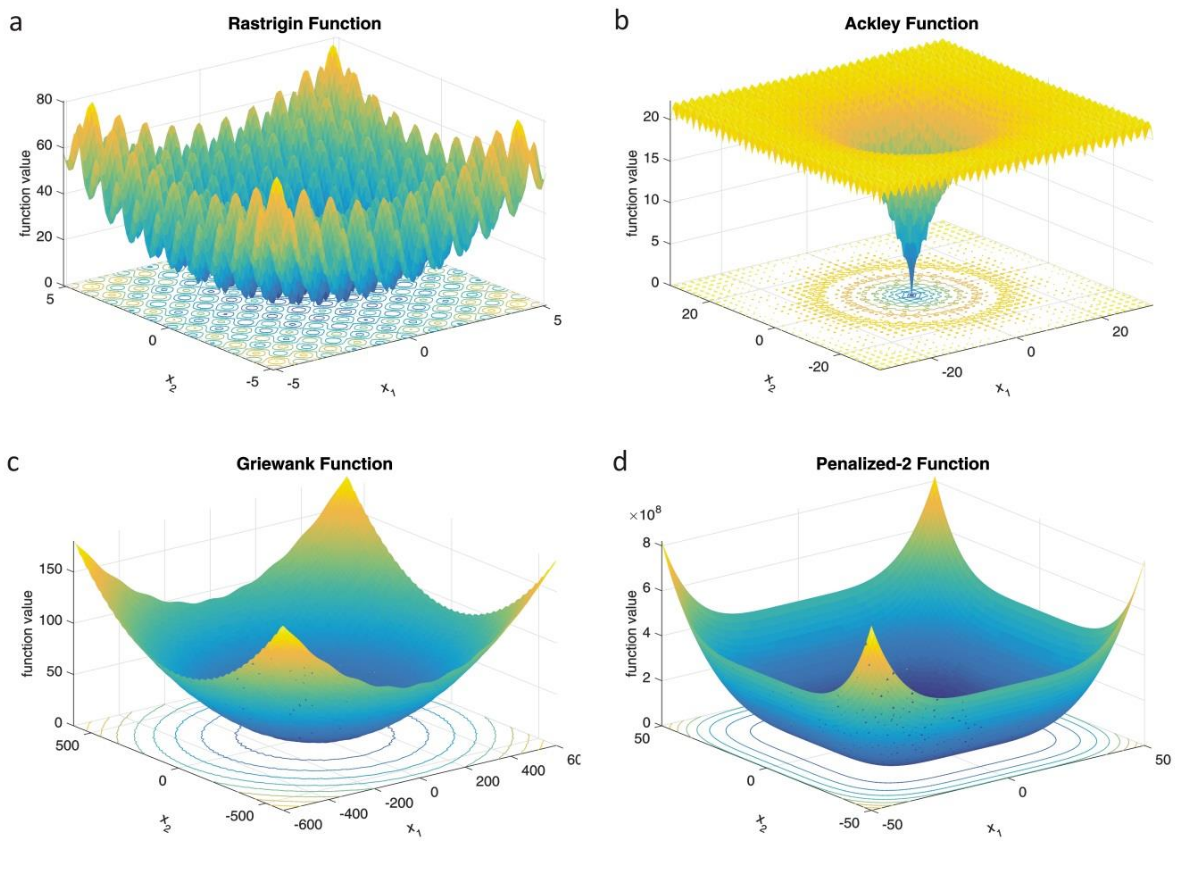

Figure 3. The illustration two-dimensional MM functions

Simulation studies have been performed on PC with the specifications of core i7 8GB. All methods are run 50 times for 50-dimensional functions. The settling parameters of CSA and ObCSA are follows: *PopSize*=25, *ite_max*=500, *AP*=1 and *fl*=1. Statistical results obtained for UM functions are given in Table 2 and convergence curves for the best solution are given in Figure 4, respectively.

According to Table 2, ObCSA-1 produces the best solutions considering Best value for all UM functions. The convergence curves in Figure 4 also support these results. The ObCSA-1 algorithm has a higher convergence rate than the others. Considering the Mean values, ObCSA methods are more successful than basic CSA in f_1 and f_4 functions.

Table 2. The statistical results of UM functions

f	Methods	Best	Mean	Worst	Std.
f_1	CSA	1.245876e-05	1.208577e+02	9.494972e+02	2.789620e+02
	ObCSA-1	5.548655e-08	3.972850e+01	2.411449e+02	8.133670e+01
	ObCSA-2	3.948795e-02	6.913407e+02	3.394790e+03	1.059401e+03
f_2	CSA	1.772417e-08	9.587184e-02	8.357679e-01	2.471808e-01
	ObCSA-1	6.547443e-09	3.552517e-01	1.638986e+00	5.800647e-01
	ObCSA-2	9.913657e-08	2.320926e-01	1.308630e+00	3.941654e-01
f_3	CSA	9.015058e-11	1.775855e-02	1.111651e-01	3.677735e-02
	ObCSA-1	5.099702e-13	2.671550e-02	1.854854e-01	5.464461e-02
	ObCSA-2	4.467390e-11	3.731923e-01	1.630211e+00	6.031711e-01
f_4	CSA	2.847369e-04	1.971604e-03	6.521443e-03	2.247900e-03
	ObCSA-1	1.215682e-04	1.960128e-03	8.490555e-03	2.576609e-03
	ObCSA-2	2.819505e-04	6.346028e-04	1.470637e-03	3.346326e-04

Statistical results obtained for MM functions are given in Table 3 and convergence curves for the best solution are given in Figure 5, respectively. According to Table 3, ObCSA-1 produces the best solutions in f_6, f_7 and f_8 functions. In f_5 function, ObCSA-2 achieves better result than others. Considering the Mean values, ObCSA methods are more successful than basic CSA in f_5 and f_8 functions.

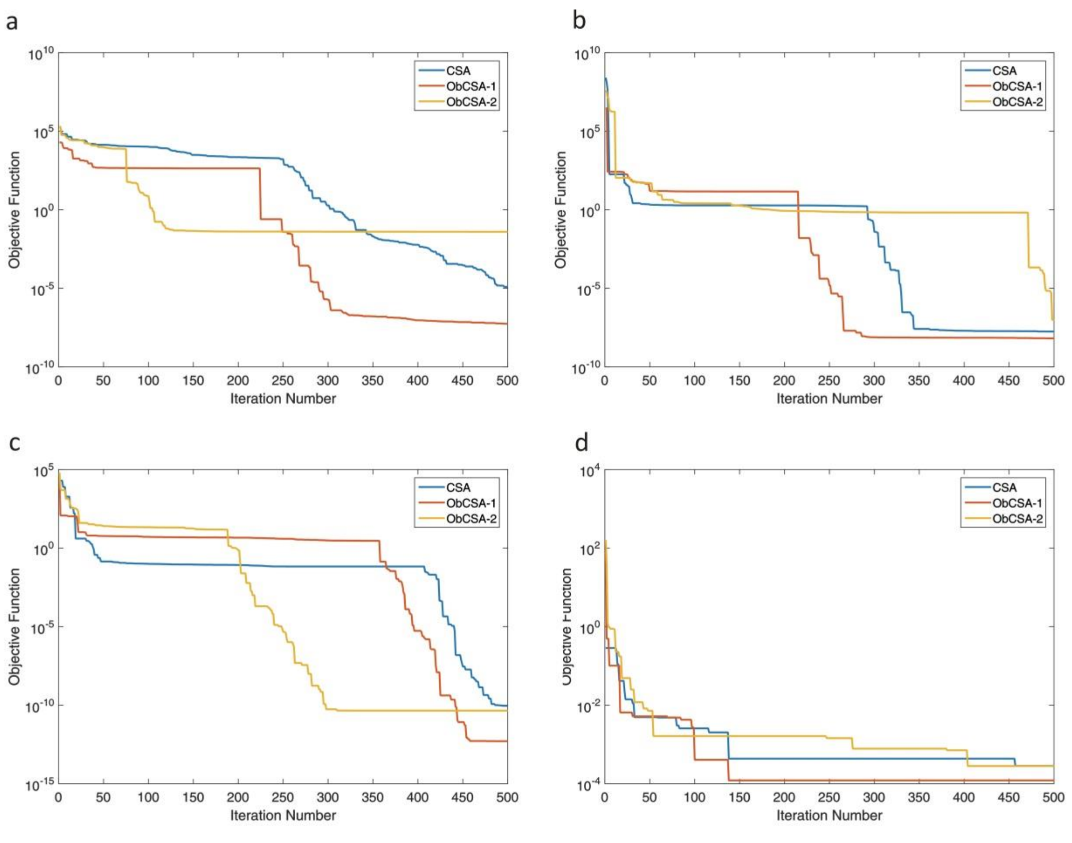

Figure 4. The convergence curves of UM functions (a) f_1; (b); f_2; (c) f_3; (d) f_4

In MM functions, the final value is more important than the convergence rate. Because the problem has too many local optima, the method used gets stuck to the local optimum. In Fig. 5, although the convergence rate of the basic CSA is high, it produces larger error values. This means that the CSA gets stuck in the local optima. The proposed ObCSA methods reach lower error values than basic CSA.

Table 3. The statistical results of MM functions

f	Methods	Best	Mean	Worst	Std.
f_5	CSA	1.705853e-10	7.967219e-02	7.551901e-01	2.252569e-01
	ObCSA-1	7.332623e-11	7.726171e-03	5.483126e-02	1.621960e-02
	ObCSA-2	8.951062e-12	1.210252e-01	6.427200e-01	1.861955e-01
f_6	CSA	4.163119e-06	2.011804e-02	1.286980e-01	4.190452e-02
	ObCSA-1	1.026417e-07	4.053685e-02	1.720274e-01	6.485713e-02
	ObCSA-2	1.138841e-07	7.631362e-02	5.845917e-01	1.723064e-01
f_7	CSA	1.020848e-10	9.164123e-02	5.909100e-01	1.871892e-01
	ObCSA-1	1.098233e-12	1.463629e-01	7.858924e-01	2.349170e-01
	ObCSA-2	1.162426e-11	1.763992e-01	7.500073e-01	2.853353e-01
f_8	CSA	1.553088e-11	3.635947e-03	1.928421e-02	6.162485e-03
	ObCSA-1	3.365973e-16	7.754907e-03	4.285018e-02	1.292313e-02
	ObCSA-2	2.581291e-12	2.592147e-03	1.399344e-02	4.931962e-03

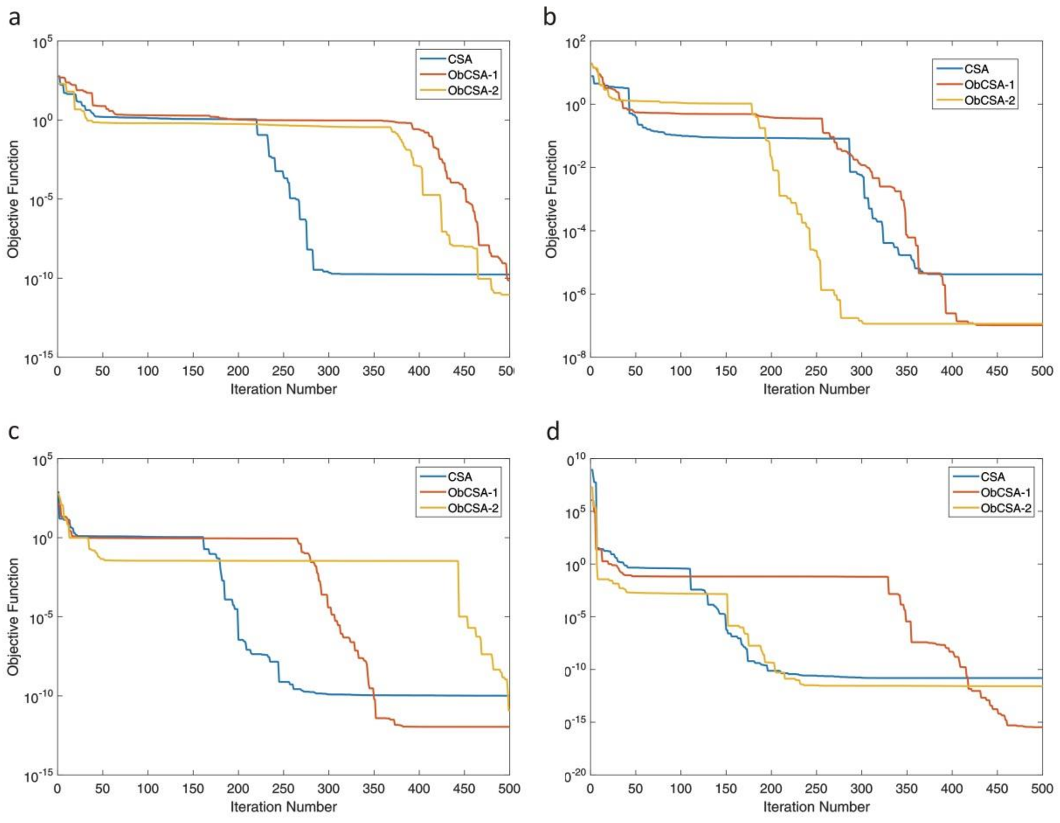

Figure 5. The convergence curves of MM functions (a) f_5; (b); f_6; (c) f_7; (d) f_8

5. Conclusions

The fact that the search algorithms start to search with a uniformly distributed population in the search space may cause a decrease in performance. Using opposite position instead of sequentially generated random values can produce solutions with better fitness values. Thus, the evolutionary computation starts with solution candidates with higher fitness values, and as a result, the process of reaching the global solution is accelerated. In this paper, opposite-based initialization strategies are integrated into the CSA algorithm. Two ObCSA methods were derived and their performances were compared with the basic CSA. The results show that the proposed approach increases the convergence rate and solution accuracy. In future studies, it is aimed to adapt the oppositional approach to new metaheuristics and to apply them to different engineering problems.

References

[1] M. Dorigo and G. Di Caro, The Ant Colony Optimization Metaheuristic, New Ideas in Optimization. New York: McGraw-Hill, 1999.

[2] J. Kennedy and R. Eberhart, “Particle swarm optimization,” In Proc. IEEE International Conference on Neural Networks, vol. IV, 1995, pp. 1942-1948.

[3] D. Karaboğa and B. Baştürk, “A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm,” Journal Global Optimization, vol. 39, no. 3, pp. 459-471, 2007.

[4] M. Yazdani and F. Jolai, “Lion optimization algorithm (LOA): A nature-inspired metaheuristic algorithm,” Journal of Computational Design and Engineering, vol. 3, no. 1, pp. 24-36, 2016.

[5] R. Rajabioun, “Cuckoo optimization algorithm,” Applied Soft Computing, vol. 11, pp. 5508–5518, 2011.

[6] M. A. Montes de Oca, T. Stützle, K. Van den Enden, and M. Dorigo, “Incremental social learning in particle swarms. IEEE Transactions on Systems Man and Cybernetics Part B Cybernetic, vol. 41, no. 2, pp. 368-384, 2011.

[7] C. Banerjee and R. Sawal, “PSO with dynamic acceleration coefficient based on multiple constraint satisfaction,” In Proc. International Conference on Advances in Electronics Computers and Communications ‘14, 2014, pp. 1-5.

[8] B. Durmuş, “Chaotic map based tree seed algorithm,” Süleyman Demirel University Journal of Natural and Applied Sciences, vol. 23, no. 2, pp. 601-610, 2019.

[9] H. R. Tizhoosh, “Opposition-based learning: A new scheme for machine intelligence,” In Proc. International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), 2005, pp. 695-701.

[10] M. G. H. Omran and S. Sharhan. “Using opposition-based learning to improve the performance of particle swarm optimization,” In Proc. IEEE Swarm Intelligence Symposium, 2008, pp. 1-6.

[11] H. Wang, H. Ouyang, L. Gao, and W. Qin, “Opposition-based learning harmony search algorithm with mutation for solving global optimization problems,” In Proc. 26th Chinese Control and Decision Conference, 2014, pp. 1090-1094.

[12] S. K. Dinkar and K. Deep, “Accelerated opposition-based antlion optimizer with application to order reduction of linear time-invariant systems,” Arabian Journal for Science and Engineering, vol. 44, pp. 2213-2241, 2019.

[13] J. Zhao, L. Lv, and H. Sun, “Artificial bee colony using opposition-based learning,” in Genetic and Evolutionary Computing, H. Sun, C. Y. Yang, C. W. Lin, J. S. Pan, V. Snasel, A. Abraham, Eds. Nanchang: Springer Cham, 2015, pp. 3-10.

[14] J. Li, T. Chen, T. Zhang, and Y. X. Li, “A cuckoo optimization algorithm using elite opposition-based learning and chaotic disturbance,” Journal of Software Engineering, vol. 10, no. 1, pp. 16-28, 2016.

[15] H. Ming, W. Mingxu, and L. Xu, “An improved genetic algorithm using opposition-based learning for flexible job-shop scheduling problem,” In Proc. 2nd International Conference on Cloud Computing and Internet of Things, 2016, pp. 8-15.

[16] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, “Opposition-based differential evolution algorithms,” In Proc. IEEE Congress on Evolutionary Computation, 2006, pp. 2010-2017.

[17] A. Askarzadeh, “A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm,” Computer & Structures, vol. 169, June, pp. 1-12, 2016.