2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Opposition-based Crossover Operation for Differential Evolution Algorithm

1st Sevda Ebrahimi
Department of Electrical, Computer, and Software Engineering
Ontario Tech University
Oshawa, ON, Canada
sevda.ebrahimi@ontariotechu.ca

2nd Shahryar Rahnamayan
Engineering Department
Brock University
St. Catharines, ON, Canada
srahnamayan@brocku.ca

3ed Azam Asilian Bidgoli
Faculty of Science
Wilfrid Laurier University
Waterloo, ON, Canada
abidgoli@wlu.ca

***Abstract*—Differential Evolution (DE) is widely recognized as an effective, robust, and gradient-free global optimization algorithm. However, the DE algorithm's search strategy has certain limitations that present opportunities for further improvement. Opposition-based Learning (OBL) as one of the efficient computational concepts provides the optimizer with the capability of exploring the search space in opposite directions. This research paper introduces a novel crossover scheme for the DE algorithm based on OBL concept. Unlike existing approaches in the literature, which primarily focus on utilization of OBL in population level, proposed scheme takes the advantage of OBL in operation level. In proposed scheme, the crossover operator generates two trial vectors in opposite directions, enhancing the exploration capability of the search strategy and taking a cautious approach by regularly examining the opposite directions during crossover. To evaluate the effectiveness of the proposed method, a series of experiments are conducted using the CEC-2017 benchmark functions with two different numbers of dimensions: 30 and 50. The results demonstrate a significant improvement in performance of the DE algorithm through the proposed method.**

***Index Terms*—Evolutionary Algorithm (EA), Differential Evolution (DE), Opposition-based Learning (OBL), Opposition-based Crossover, Operation-level Opposition**

I. Introduction

Evolutionary Algorithms (EAs) are stochastic metaheuristic search methods based on Darwinian evolution and phenomena of natural selection of the fittest [1]. In the past few decades, EAs have been employed as a useful approach to tackle a wide range of difficult problems and find solutions effectively [2]. Among these algorithms, Differential Evolution (DE) has received a significant attention [3]. Just like other EAs, DE starts its optimization process with a population initialization. The population consists of several individuals defined as the potential solutions for a given optimization problem. Individuals are evolved within evolutionary processes including: mutation, crossover, and selection operations which consequently lead to generating new offspring which are expected to become closer to the optimal solution generation-by-generation. With the efficient search strategy and at times a prior knowledge about the optimal solutions is available, the algorithm can achieve a promising performance. However, in the lack of prior knowledge, which is the most common scenario, DE starts the optimization with random candidate solutions which may reduce the chance of finding better solutions through evolution.

The concept of Opposition-based Learning (OBL) is first introduced in [4]. The primary motivation of OBL is to search for potential solutions in a search space by simultaneously exploring both the opposite and primary directions of the investigation process. OBL is used in various searching, learning, and optimization algorithms including Reinforcement Learning (RL) [5], Artificial Neural Networks (ANN) [6], Fuzzy systems [7], and optimization algorithms. OBL proved to be effective in improving search capability of several metaheuristics including evolutionary computations such as Genetic Algorithm (GA) [8], Differential Evoultion (DE) [9], Ant Colony System Optimization [10], beluga whale Optimization Algorithm [11], Harmony Search [12], Simulated Annealing [13], Swarm Intelligence [14], and Multi-Objective Optimization. The main idea behind OBL is simultaneously evaluation of a candidate solution and its opposite in the search space in order to have an effective investigation for an optimal solution. This in fact, increases the likelihood of discovering optimal solutions and empowers the exploration ability of the algorithms.

In recent years, multiple research studies have been conducted to expand the novel OBL approaches to enhance DE algorithm. Opposition-based Differential Evolution algorithm (ODE) is one of the most successful and well-known methods developed by embedding OBL in population initialization and generation jumping [9]. Different variants of DE have been developed by utilizing the OBL concept with various application. Quasi Oppositional DE (QODE) benefit from the first variant of OBL in DE [15]. Generalized OBL is another scheme embedded in DE algorithm developing Generalized opposition-based DE (GODE) [16]. In [17], different strategy of utilizing OBL concept is embedded in DE in mutation operation. Centroid Opposition-based Differential Evolution (CODE), defines the centroid-opposite point for the candidate solution which gravity center of the population is used for generating opposite candidate solution instead of using minimum and maximum boundary of candidate solution [18]. Another different scheme of OBL concept is introduced to DE in [19], when triple comparison method is conducted in selection step including a random candidate solution, the target, trial, and their corresponding opposites. These opposite solutions are calculated based on principle definition for the opposite of

978-0-7381-4408-5/23/$31.00 ©2023 IEEE

a point in OBL using minimum and maximum boundaries. A comparative study conducted on performance of eight different OBL schemes imbedded in DE in [20], to mention some; a comprehensive survey on OBL was published in [21], [22].

In this paper, the concept of opposition-based optimization is applied to DE algorithm in operation level, when performing the crossover operation. The primary objective is to enhance the algorithm's capacity of exploring the optimal solution. This technique allows the algorithm to intelligently explore the search space with current candidate solution generated by crossover and its opposite. Consequently, the evolutionary algorithm will benefit from a broad view when searching for the optimal solution. During the exploration, the opposite candidate solution will support a high diversity for the exploration, and during the fine-tuning it will contribute on a finer exploitation depending on optimization phase.

The paper is structured as follows: Section II provides a brief overview of the DE algorithm and the concept of OBL. In section III, the proposed approach of customized opposition-based crossover for the DE algorithm is discussed in detail. In section IV, the experiments on the test functions, parameter settings, and the comparison of the proposed method with the classical DE algorithm (DE/rand/1/bin) are explained. Finally, the paper is summarized and concluded in section V.

II. A BRIEF REVIEW OF DIFFERENTIAL EVOLUTION AND OPPOSITION-BASED LEARNING

In this section, DE and opposition-based optimization are explained in detail as the background review section.

A. Differential Evolution Algorithm (DE)

DE is a popular optimization method which is simple, effective and has been successfully applied to many optimization problems. Complex problems which include linear and nonlinear, unimodal and multimodal solution landscapes can be successfully solved by applying this algorithm. In DE, when no prior information or solution is available, the process of searching for optimal solution, starts with initializing random vectors as individuals of the population. Then, the algorithm tries to improve the quality of candidate solutions in the population, generation by generation, to reach an optimal solution. The algorithm is able to search for a better candidate solution by applying the specific operations including mutation, crossover, and selection. These three operations are the major building blocks of the DE algorithm.

Suppose that the population $\mathbf{P}(G) = \{\mathbf{X}_{1,G}, ..., \mathbf{X}_{N_p,G}\}$ consists of N_p vectors in generation G, where $\mathbf{X}_{i,G}(i = 1, 2, ..., N_p)$ is the ith individual in population $\mathbf{P}(G)$. Each $\mathbf{X}_i$ is a D-dimensional vector defined as $\mathbf{X}_i = \{X_{i,1}, ..., X_{i,D}\}$. A simple DE algorithm includes three major operations: mutation, crossover, and selection. Mutation and crossover are responsible for generating mutant and trial vectors respectively and selection operation determines which individual must be selected into the next generation.

Mutation–In this step for each vector $\mathbf{X}_{i,G}$ in generation G, three vectors are uniform randomly selected from the population such that $i_1 \neq i_2 \neq i_3 \neq i$ where $i \in \{1, ..., N_p\}$. For each vector $\mathbf{X}_i$, the mutant vector is generated as follow [3]:

$$\mathbf{V}_{i,G} = \mathbf{X}_{i_1,G} + F(\mathbf{X}_{i_2,G} - \mathbf{X}_{i_3,G}), \tag{1}$$

The population size N_p should fulfill the condition of $N_p \geq 4$ since i, i_1, i_2, and i_3 are different. $F \in [0, 2]$ is a real constant number defined as mutation factor, scaling factor or differential weight that controls the amplification of the differential variation of $(\mathbf{X}_{i_2,G} - \mathbf{X}_{i_3,G})$.

Crossover-In DE, crossover operation is responsible for creating trial vectors by shuffling the parent vector and mutant vector. The trial vector is calculated as follows [3]:

$$U_{i,j,G} = \begin{cases} V_{i,j,G}, & rand_j(0,1) \leq Cr \text{ or } j_{rand} = j, \\ X_{i,j,G}, & \text{otherwise.} \end{cases} \tag{2}$$

where $j \in \{1, 2, ..., N_p\}$, $Cr \in$ [0,1] is the crossover rate, and $rand_j(0, 1)$ is a random number in interval [0,1] for jth dimension. Finally, the trial vector is defined as follow [3]:

$$\mathbf{U}_{i,G} = (U_{i,1,G}, U_{i,2,G}, ..., U_{i,D,G}), \tag{3}$$

where $i \in \{1, 2, ..., N_p\}$.

Selection-This operation evaluates the $\mathbf{U}_{i,G}$ and $\mathbf{X}_{i,G}$, and compares them with respect to their fitness values. The selection mechanism is defined as follows [3]:

$$\mathbf{X}_{i,G} = \begin{cases} \mathbf{U}_{i,G}, & f(\mathbf{U}_{i,G}) \leq f(\mathbf{X}_{i,G}), \\ \mathbf{X}_{i,G}, & \text{otherwise.} \end{cases} \tag{4}$$

where $f(X)$ is an objective function to be minimized or maximized based on the optimization problem definition. Considering the minimization problems in this paper and in the selection mechanism defined in "(9)", if the fitness value of the trial vector $\mathbf{U}_{i,G}$ is equal to or lower than the parent vector $\mathbf{X}_{i,G}$, then the $\mathbf{X}_{i,G}$ will be set as the $\mathbf{U}_{i,G}$, otherwise the $\mathbf{X}_{i,G}$ will remain unchanged.

B. Opposition-Based Learning (OBL)

In this section, the concept of OBL is briefly discussed from Type-I and min-max-based opposition viewpoints [18], [23]. These two main schemes of OBL concept have been utilized in development of proposed novel OBL scheme which is an opposition-based crossover operation in DE. Whilst evolutionary processes in EAs, the main purpose of OBL is to help these algorithms explore the search space in a better way by integrating the diversity to possible candidate solutions and inclusive exploration and selection. Consequently, this approach can increase convergence rate of the optimizer. Opposite number and opposite point are defined as follows.
Opposite number-Let $x \in [a, b]$ be a real number. The opposite of x is defined by [4]:

$$\breve{x} = b + a - x \tag{5}$$

Opposite point in the n-dimensional space-Let us assume $P(x_1, x_2, ..., x_n)$ is a candidate solution in an n-dimensional

space with $x_i \in [a_i, b_i]$, $\forall i \in \{1, 2, ..., n\}$. Based on opposite point definition, $\breve{P}(\breve{x}_1, \breve{x}_2, ..., \breve{x}_n)$ is the opposite of $P(x_1, x_2, ..., x_n)$ such that [4]:

$$\breve{x}_i = b_i + a_i - x_i \tag{6}$$

where a_i and b_i are lower boundary and upper boundary for the ith dimension respectively. Different variants of OBL schemes exist in the literature, however the proposed scheme in this study inspired by min-max-based opposition and other variants are somehow out of scope of this study.

III. THE PROPOSED ALGORITHM

In this section, OBL scheme in DE algorithm is introduced in a way that employs OBL concept in operation-level of the algorithm, resulting a novel crossover scheme for DE. Most approaches introduced in the literature typically focus on population-level scheme of the opposition concept to enhance the performance of an evolutionary algorithm. However, our proposed method indicates that OBL when integrated in crossover operation, significantly improves the algorithm with minimal algorithm modification and without additional control parameter.

In conventional DE algorithm a trial vector is generated by applying crossover operator to parent and mutant vectors. The generated trial vector is then compared with parent vector and then the fittest among trial and parent will participate in new generation. This study, introduces opposite of trial vector to the algorithm that can be generated by crossover operator at the same time as generating trial vector. The primary idea is to keep the valuable information that Cr discards in classic DE when trial vector is being generated. Proposed crossover operator calculates trial vector in "(2)" and opposite of trial vector (opposite-trial vector) as follows:

$$\breve{U}_{i,j,G} = \begin{cases} X_{i,j,G}, & rand_j(0,1) \leq Cr \text{ or } j_{rand} = j, \\ V_{i,j,G}, & \text{otherwise.} \end{cases} \tag{7}$$

Finally, the opposite-trial vector is defined as:

$$\breve{\mathbf{U}}_{i,G} = (\breve{U}_{i,1,G}, \breve{U}_{i,2,G}, ..., \breve{U}_{i,D,G}) \tag{8}$$

where same as "(3)" $i \in \{1, 2, ..., N_p\}$ and, G is the generation index. The way that opposite-trial vector is generated is quite simple and the impact that it has on enhancement of the algorithm is significant. For each gene, there are two options to take the value from the parent or mutant vector. So, depending on the selection of the gene values from the parent or mutant vectors, the opposition can be defined. This scheme generates extra point or vector in the search space by making use of a hyper-rectangle. This hyper-rectangle is made using parent and mutant vectors. Fig. 1 illustrates the cuboid generated using a sample parent vector x, and a sample mutant vector v for three-dimensional search space. In Fig. 1, After performing the proposed crossover operation, trial vector u will be generated referring to one of these vertices in the cuboid. It is worth mentioning that, based on the value that Cr takes in the algorithm, trial vector will be closer to, or

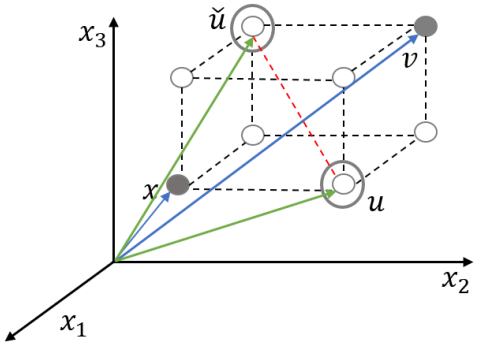

Fig. 1: The visual illustration of sample parent (x), trial (u), opposite-trial vectors ($\breve{u}$) in a cuboid generated by parent and mutant vector (v) in three-dimensional search space. Trial vector and its opposite always refer to opposite vertices in the generated hyper-rectangle.

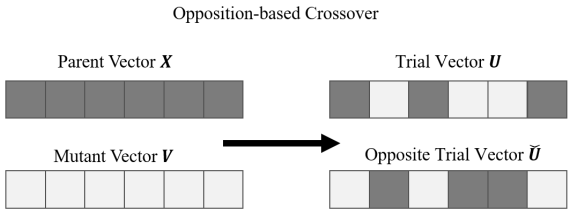

Fig. 2: The visual illustration of generated trial vector and its opposite by shuffling the mutant and parent vector.

far from the mutant vector. At the same time, opposite-trial vector $\breve{u}$ will be generated referring to the opposite vertex of which trial vector is referring to. Finally, after performing crossover and in selection step, the fittest among x, u, $\breve{u}$ will be chosen to be present in the next generation. We have called this algorithm Opposition-based Crossover Differential Evolution (Op-DE) algorithm. Fig. 2 shows the trial vector U and its opposite $\breve{U}$ for better illustration of the reverse selection. Thus, for Op-DE algorithm, beside calculating primary trial vector U, simultaneously, the crossover operation calculates the opposite of the primary trial vector $\breve{U}$. The opposite trial vector is generated just by reversely selecting the genes that are discarded for selection in generating the trial vector. This can increase the chance of finding better candidate solutions and consequently exploring the search space by evaluating both vectors.

During the selection process, the $U_{i,G}$, $\breve{U}_{i,G}$ and $X_{i,G}$ vectors are evaluated and the vector with the superior fitness value is chosen as the parent vector for the subsequent generation. Given a minimization problem, the selection procedure in determining the fittest vector is formulated as follows:

$$\mathbf{X}_{i,G} = \begin{cases} \mathbf{U}_{i,G}, & f(\mathbf{U}_{i,G}) \leq f(\mathbf{X}_{i,G}) \wedge f(\mathbf{U}_{i,G}) \leq f(\breve{\mathbf{U}}_{i,G}), \\ \breve{\mathbf{U}}_{i,G}, & f(\breve{\mathbf{U}}_{i,G}) < f(\mathbf{U}_{i,G}) \wedge f(\breve{\mathbf{U}}_{i,G}) < f(\mathbf{X}_{i,G}), \\ \mathbf{X}_{i,G}, & \text{otherwise.} \end{cases} \tag{9}$$

where $f(X)$ indicates the fitness value of vector $\mathbf{X}$. Algorithm 1 demonstrates the way that OBL is applied to the crossover operation of classical DE algorithm. The mutation and selection are same as original DE.

In the following section, experimental results also validate that the proposed method in this paper improves DE algorithm. Experiments indicate that opposite trial vector, which has been disregarded prior to this study, has valuable information that can assist the algorithm towards achieving an optimal solution.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, in order to investigate the effect of the proposed Op-DE algorithm, the algorithm is evaluated with limited computational budget and in comparison with DE/rand/1/bin variant of classic DE algorithm. To this end, the CEC-2017 is selected as the set of benchmark functions. Experiments on CEC-2017 benchmark functions, have been conducted on two different dimensions, 30 and 50, to evaluate the performance of the proposed method. In the next subsection, the parameter setting, and the benchmark functions set are defined. Then, the comparison strategies and metrics for evaluating performance of the proposed algorithm are explained.

A. Benchmark functions and parameters setting

The set of benchmark functions used to evaluate and compare the performance of DE algorithm and proposed Op-DE algorithm, consist of 29 minimization problems. The benchmark functions include 2 unimodal functions ($f_1 - f_2$), 7 simple multimodal functions ($f_3 - f_9$), 10 hybrid functions ($f_{10} - f_{19}$), and 10 composition functions ($f_{20} - f_{29}$). The parameter settings are described in Table I. Parameters are identical for both DE and Op-DE algorithms. The maximum number of function evaluations has been set to 3000*D for both algorithms and the dimension of the problems in these experiments has been set to 30 and 50. Due to the stochasticity of evolutionary algorithms, we run each algorithm 31 times. T-statistic test with the significance level of α = 0.05 is also performed and applied to the results to confirm any significant differences between the results of the two algorithms.

B. Results Analysis

To compare the performance of the DE and Op-DE algorithms, the mean value of the fitness functions and the standard deviation (Std) of the fitness function values over 31 runs are compared. Table II presents the results for D=30 and Table III presents the results for D=50. In Table II and Table III, the results of the algorithm with the better fitness value are shown in bold.

1) Comparison of DE and proposed Op-DE on D=30: In Table II, the results regarding the mean values, Std values, the rejection or acceptance of null hypothesis (H_0) and the p-value show that the DE and the proposed Op-DE have different performance on 16 functions of the benchmark and for the rest of 13 functions, both algorithms present similar performance and there is no significant difference between these two algorithms. On 16 functions with significantly different outcomes, proposed Op-DE algorithm outperforms the DE algorithm on 12 functions and only on 4 functions, DE shows a better performance. In addition, for these 13 functions the Std values are significantly lower in the proposed Op-DE algorithm, which is an indicator for a robust behavior of the proposed algorithm. In Fig. 3, performance plots for functions 1, 16, and 21 are presented. In functions 1 and 21, Op-DE has better performance than DE algorithm. However, in function 16, DE algorithm outperforms Op-DE algorithm. Fig. 4 shows the contribution rate of the parent, trial, and opposite trial vectors in selection stage in each iteration of the proposed algorithm. Graphs show that the trial vectors have have valuable information and consequently contribution in generating new offspring because they are selected as parent for the new population in each iteration. It is clear that with Cr of 0.9, the ratio of selected genes from mutant vector in trial vector is higher than those from parent vector. This is vice versa in the opposite vector and may result in lower contribution of opposite vector compared to trial vector. However, ignoring the opposite vector causes the loss of that

Algorithm 1 Opposition-based Crossover DE

1: Parameter setting $(F, Cr, D, NFC_{Max}, N_P)$
2: Initialize individuals in the population $\mathbf{P}(G)$ randomly
3: Compute the fitness value for each $\mathbf{X_{i,G}}$
4: **While** $(NFC \leq NFC_{Max})$ **do**
`/* Continue performing the classical DE */`
5: **for** i = 1 to N_P **do**
6: Select 3 random vectors like $\mathbf{X}_{i_1,G}, \mathbf{X}_{i_2,G}, \mathbf{X}_{i_3,G}$ from $\mathbf{P}(G)$ where $(i_1 \neq i_2 \neq i_3 \neq i)$;
7: $\mathbf{V}_{i,G} = \mathbf{X}_{i_1,G} + F(\mathbf{X}_{i_2,G} - \mathbf{X}_{i_3,G})$;
`/* apply Opposition-based Crossover * /`
9: **for** j = 1 to D **do**
10: **if** $rand(0,1) \leq Cr$ **then**
11: $U_{i,j,G} = V_{i,j,G}$;
12: $\breve{U}_{i,j,G} = X_{i,j,G}$;
13: **else**
14: $U_{i,j,G} = X_{i,j,G}$;
15: $\breve{U}_{i,j,G} = V_{i,j,G}$;
16: **end if**
17: **end for**
18: Evaluate $\mathbf{U}_{i,G}$ and $\breve{\mathbf{U}}_{i,G}$ as $f(\mathbf{U}_{i,G})$ and $f(\breve{\mathbf{U}}_{i,G})$;
19: NFC=NFC+2;
20: $\mathbf{X}_{i,G}$=Select the fittest individual from the set $\{\mathbf{X}_{i,G}, \mathbf{U}_{i,G}, \breve{\mathbf{U}}_{i,G}\}$
21: **end for**
22: G=G+1;
23: **end while if criterion meets**

TABLE I
PARAMETER SETTING FOR ALL EXPERIMENTS, FOR BOTH DE AND Op-DE ALGORITHMS.

Parameter	Description	Value
N_P	Population size	50
F	Differential weight	0.5
Cr	Crossover rate	0.9
NFC_{Max}	Maximum number of function call	3000*D
N_{Run}	Number of runs	31

TABLE II
PERFORMANCE COMPARISON OF DE AND PROPOSED Op-DE FOR D=30 ON CEC-2017 BENCHMARK FUNCTION. THE FITTEST VALUE FOR EACH FUNCTION IS HIGHLITED IN BOLDFACE.

F	DE		Op-DE		T-test	
	mean	std	mean	std	p-value	H_0
1	3462.17	21961457	**101.37**	58.67	0.0001	1
2	**3673.93**	8963294	17081.75	33682085	1.0e-16	1
3	485.59	327.33	487.31	19.26	0.609	0
4	680.85	202.37	**611.82**	114.67	5.6e-30	1
5	600.00	7.19e-08	600.00	8.02e-10	0.73	0
6	917.99	137.99	**851.44**	74.04	7.6e-34	1
7	972.29	792.02	**916.28**	67.81	1.9e-15	1
8	900.25	0.17	**900.00**	1.15e13	0.0009	1
9	7866.46	1000133	**5789.08**	74014	2.8e-16	1
10	1130.95	1037.03	**1158.81**	825.34	0.0006	1
11	42979	1355260700	53841	1827336455	0.28	0
12	4193.89	93882291	1942.60	3426757.38	0.208	0
13	1465.96	145.63	**1452.35**	56.16	1.5e-06	1
14	1544.62	1001.28	1533.00	109.45	0.056	0
15	2243.81	224476	2326.90	49411	0.38	0
16	**1836.22**	9266	1891.88	12280	0.03	1
17	7655.37	36635444	9072.40	57056980	0.41	0
18	**1914.25**	27.44	1925.36	25.03	5.8e-12	1
19	2072.17	10828	2105.11	5091	0.15	0
20	2472.54	105.08	**2413.81**	85.18	3.6e-32	1
21	6201.25	12994198	**3125.69**	3672684	9.1e-05	1
22	2797.55	3685.12	**2760.98**	54.96	0.001	1
23	2995.30	896.86	**2940.74**	92.09	7.8e-14	1
24	2886.95	1.05	2886.90	0.017	0.82	0
25	4699.80	569087	4755.35	12042	0.68	0
26	3204.26	90.52	3201.33	53.81	0.17	0
27	3191.92	1432	3201.13	1097	0.31	0
28	**3400.50**	21293	3565.06	7405	1.1e-06	1
29	5906.83	1876553	5640.87	281710	0.31	0
w/t/l			12/13/4			

much contribution which is beneficial for reaching the better candidate solutions.

2) Comparison of DE and proposed Op-DE on D=50: To assess the performance of the proposed algorithm in higher dimensions, the same experiment is conducted for D=50. Table III presents the results of the experiment, including the mean, Std, and t-statistical test p-value. Based on these metrics, the DE and proposed Op-DE algorithms demonstrate similar performance on 8 benchmark functions. Amongst other 21 functions of the benchmark, the proposed Op-DE algorithm outperforms the DE algorithm on 17 functions, whereas DE exhibits a superior fitness value on only 4 functions. Except function number 27, the Std values are also significantly lower

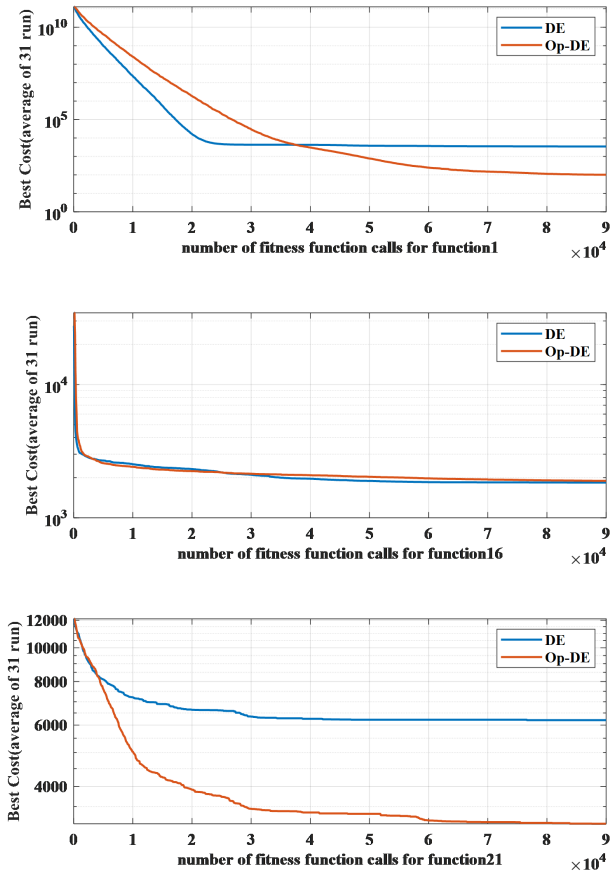

Fig. 3: Mean value of best so far solution (over 31 runs) versus number of function calls. Performance comparison between DE algorithm and proposed Op-DE algorithm for D=30.

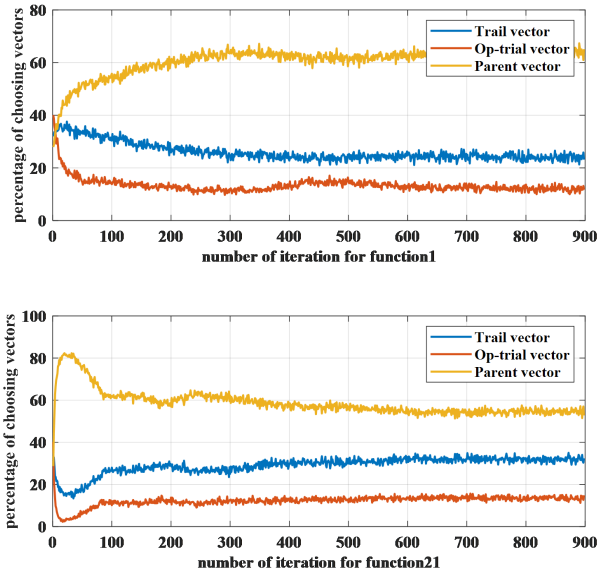

Fig. 4: Selection rate of parent, trial, and the opposite trial vector in each iteration in proposed Op-DE algorithm for D=30.

TABLE III
PERFORMANCE COMPARISON OF DE AND PROPOSED Op-DE FOR D=50 ON CEC-2017 BENCHMARK FUNCTION. THE FITTEST VALUE FOR EACH FUNCTION IS HIGHLITED IN BOLDFACE.

F	DE		Op-DE		T-test	
	mean	std	mean	std	p-value	H_0
1	7294.62	70583385	5764.53	60393255	0.459	0
2	**126272**	571550647	173488	978416825	8.8e-09	1
3	514.53	3284	510.72	1107	0.75	0
4	835.80	4377	**777.10**	231	1.0e-05	1
5	600.01	0.00078	**600.00**	4.5e-09	0.001	1
6	1117.63	294	**1043.69**	178	5.1e-27	1
7	1152.56	828	**1078.05**	256	1.6e-18	1
8	943.46	10113	**901.85**	15.79	0.024	1
9	14041.50	2135443	**10928.85**	98129	5.8e-17	1
10	**1169.31**	1767	1201.89	512	0.0003	1
11	364974	96239305610	444278	119879230765	0.346	0
12	12013.19	71512350	**6421.64**	29719418	0.002	1
13	2733.77	3907766	2128.45	934963	0.130	0
14	3649.11	7525914	**2176.05**	1540661	0.008	1
15	3749.55	944580	**3290.52**	76923	0.014	1
16	2868.20	188284	2723.64	59085	0.110	0
17	46434.32	1286887953	34999.43	616883575	0.149	0
18	5432.68	53949715	2664.26	8279479	0.055	0
19	3100.80	111496	**2946.08**	35431	0.02	1
20	2643.17	3085	**2585.92**	172	6.0e-07	1
21	15654.78	267581	**12648.23**	136580	1.1e-34	1
22	2999.35	13847	3009.10	220	0.64	0
23	3258.83	835.18	**3189.07**	506.22	2.2e-15	1
24	3026.84	1663.52	**3000.26**	1224.92	0.007	1
25	**5103.75**	1496720	6468.20	29196	6.8e-08	1
26	3352.54	8034	**3249.99**	1506	2.2e-07	1
27	3301.23	397.13	**3270.84**	402.85	1.3e-07	1
28	**3458.03**	85405	3917.83	34670	5.4e-10	1
29	709846	15848330568	**633805**	2071056953	0.002	1
w/t/l	**17/8/4**					

in these 17 functions where the proposed Op-DE algorithm outperforms the DE algorithm. In Fig. 5, performance plots for functions 9, 19, and 26 are presented for better investigation of algorithms' efficiency for D=50. Fig. 6 also represents the contribution rate of the parent, trial, and opposite trial vectors in selection stage for D=50.

Almost in all plots for the selection rate of parent, trial, and the opposite trial, it is seen: a) during the exploration, we observe a monotonically increasing trend for parent ratio plot and monotonically decreasing behavior for both trial and opposite trial, b) during the exploitation they stay with their almost flat ratios (i.e., curves). As seen, even with a low contribution ratio for trial and opposite trial vectors, utilizing them can improve

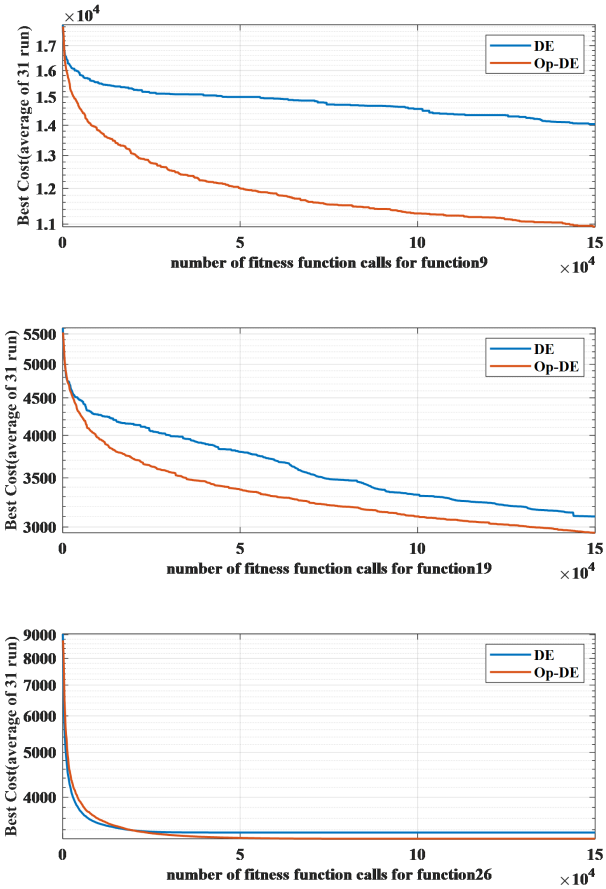

Fig. 5: Mean value of best so far solution (over 31 runs) versus number of function calls. Performance comparison between DE algorithm and proposed Op-DE algorithm for D=50.

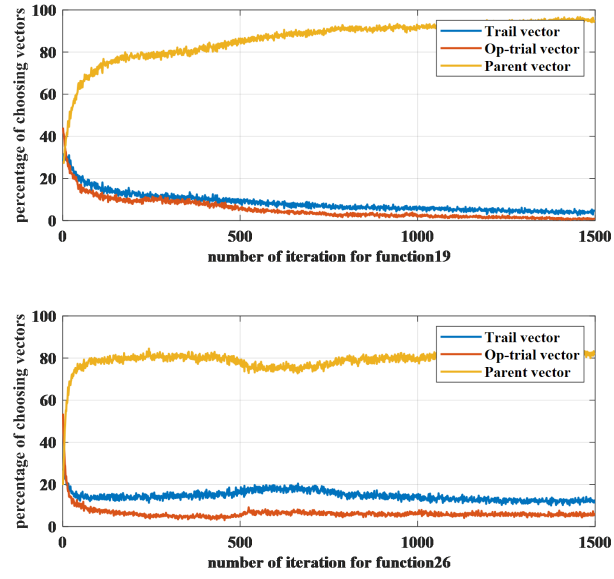

Fig. 6: Selection rate of parent, trial, and the opposite trial vector in each iteration in proposed Op-DE algorithm for D=50.

the convergence rate of the proposed algorithm significantly.

V. CONCLUSION REMARKS

This paper introduces the application of Opposition-based Learning (OBL) to the DE algorithm at the operation level, specifically within the crossover operation, with the aim of improving its performance. By incorporating the OBL concept, the DE algorithm benefits from enhanced exploration and increased search capability. This is achieved by introducing an opposite trial vector alongside the original trial vector after the crossover operation is applied to the mutant and parent vectors. By leveraging the valuable information present in the opposite trial vector, the algorithm explores the search space more effectively and efficiently. Including both possible trials provide the optimizer with additional candidate solutions to improve the population within the same iteration.

To evaluate and compare the performance of the proposed Op-DE algorithm against the classic DE algorithm, the CEC-2017 benchmark functions with varying dimensions were utilized. The results demonstrate that the Op-DE algorithm outperforms the classic DE algorithm for different dimensions (30 and 50). Specifically, the proposed algorithm exhibits accelerated convergence in reducing the fitness value compared to the classical DE algorithm. Notably, the proposed opposition-based crossover scheme is straightforward to implement and can be applied to other variants of DE. It proves effective in addressing the premature convergence issue in solving deceptive and highly multi-modal optimization problems.

This paper presents a promising direction for defining opposition-based operations without the need for additional control parameters. As future research, the proposed method will be further investigated in solving large-scale and many-objective optimization problems.

REFERENCES

[1] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proceedings of the 2004 congress on evolutionary computation (IEEE Cat. No. 04TH8753)*, vol. 2, pp. 1980–1987, IEEE, 2004.

[2] D. Dasgupta and Z. Michalewicz, *Evolutionary algorithms in engineering applications*. Springer Science & Business Media, 2013.

[3] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, pp. 341–359, 1997.

[4] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06)*, vol. 1, pp. 695–701, IEEE, 2005.

[5] H. R. Tizhoosh, "Opposition-based reinforcement learning," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 10, no. 3, 2006.

[6] M. Ventresca and H. R. Tizhoosh, "Improving the convergence of backpropagation by opposite transfer functions," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pp. 4777–4784, IEEE, 2006.

[7] H. R. Tizhoosh, "Opposite fuzzy sets with applications in image processing.," in *IFSA/EUSFLAT Conf.*, pp. 36–41, Citeseer, 2009.

[8] J. J. Grefenstette, "Genetic algorithms and machine learning," in *Proceedings of the sixth annual conference on Computational learning theory*, pp. 3–4, 1993.

[9] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Opposition-based differential evolution algorithms," in *2006 IEEE international conference on evolutionary computation*, pp. 2010–2017, IEEE, 2006.

[10] D. Zhao, L. Liu, F. Yu, A. A. Heidari, M. Wang, H. Chen, and K. Muhammad, "Opposition-based ant colony optimization with all-dimension neighborhood search for engineering design," *Journal of Computational Design and Engineering*, vol. 9, no. 3, pp. 1007–1044, 2022.

[11] S.-L. Xu, Q.-W. Chai, W.-M. Zheng, J.-S. Pan, and P. Hu, "An opposition-based beluga whale optimization," in *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 363–373, Springer, 2022.

[12] A. A. Alomoush, A. A. Alsewari, H. S. Alamri, K. Z. Zamli, W. Alomoush, and M. I. Younis, "Modified opposition based learning to improve harmony search variants exploration," in *Emerging Trends in Intelligent Computing and Informatics: Data Science, Intelligent Information Systems and Smart Computing 4*, pp. 279–287, Springer, 2020.

[13] M. Ventresca and H. R. Tizhoosh, "Simulated annealing with opposite neighbors," in *2007 IEEE Symposium on Foundations of Computational Intelligence*, pp. 186–192, IEEE, 2007.

[14] M. Agarwal and G. M. S. Srivastava, "Opposition-based learning inspired particle swarm optimization (opso) scheme for task scheduling problem in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 10, pp. 9855–9875, 2021.

[15] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Quasi-oppositional differential evolution," in *2007 IEEE congress on evolutionary computation*, pp. 2229–2236, IEEE, 2007.

[16] H. Wang, S. Rahnamayan, and Z. Wu, "Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems," *Journal of Parallel and Distributed Computing*, vol. 73, no. 1, pp. 62–73, 2013.

[17] J. Xie and J. Yang, "Improved differential evolution for global optimization," in *2010 2nd IEEE International Conference on Information Management and Engineering*, pp. 651–654, IEEE, 2010.

[18] S. Rahnamayan, J. Jesuthasan, F. Bourennani, H. Salehinejad, and G. F. Naterer, "Computing opposition by involving entire population," in *2014 IEEE congress on evolutionary computation (CEC)*, pp. 1800–1807, IEEE, 2014.

[19] Y. Pei and H. Takagi, "Triple and quadruple comparison-based interactive differential evolution and differential evolution," in *Proceedings of the twelfth workshop on Foundations of genetic algorithms XII*, pp. 173–182, 2013.

[20] W. Wang, H. Wang, H. Sun, and S. Rahnamayan, "Using opposition-based learning to enhance differential evolution: A comparative study," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 71–77, IEEE, 2016.

[21] S. Mahdavi, S. Rahnamayan, and K. Deb, "Opposition based learning: A literature review," *Swarm and evolutionary computation*, vol. 39, pp. 1–23, 2018.

[22] M. Pant, H. Zaheer, L. Garcia-Hernandez, A. Abraham, *et al.*, "Differential evolution: A review of more than two decades of research," *Engineering Applications of Artificial Intelligence*, vol. 90, p. 103479, 2020.

[23] H. Salehinejad, S. Rahnamayan, and H. R. Tizhoosh, "Type-ii opposition-based differential evolution," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1768–1775, IEEE, 2014.