Partial Opposition-based Learning Using Current Best Candidate Solution

Sedigheh Mahdavi
Department of Electrical, Computer, and Software Engineering
University of Ontario Institute of Technology (UOIT)
Oshawa, Canada
Email: Sedigheh.Mahdavi@uoit.ca

Shahryar Rahnamayan, SMIEEE
Department of Electrical, Computer, and Software Engineering
University of Ontario Institute of Technology (UOIT)
Oshawa, Canada
Email: Shahryar.Rahnamayan@uoit.ca

Kalyanmoy Deb, FIEEE
Department of Electrical and Computer Engineering
Michigan State University
East Lansing, USA
Email: kdeb@egr.msu.edu

***Abstract*—Opposition based learning (OBL) has been gaining significant attention in machine learning, specially, in meta-heuristic optimization algorithms to take OBL's advantage for enhancing their performance. In OBL, all variables are changed to their opposites while some variables are currently holding proper values which are discarded and converted to worse values by performing opposite. The partial opposition scheme was developed to change randomly some variables to its opposites but they do not pay enough attention to identify and keep variables which have proper values. In this paper, we propose a novel partial opposition scheme, which is generated based on the current best candidate solution. It tries to generate new trial solutions by using the candidate solutions and their opposites such that some variables of a candidate solution are remain unchanged and other variables are changed to their opposites in the trial solution (i.e., gene/variable based optimization). Variables in the trial solution are identified as close or far, according to their Euclidean distance from the corresponding variables/genes in the current best candidate solution. The proposed scheme uses the opposite of variables, which are closer to the current best solution. Only the new trial solutions are included in the next generation which are closer to corresponding opposite solution. As a case study, we employ the proposed partial opposition scheme in the DE algorithm and the partial opposition-based DE is evaluated on CEC-2014 benchmark functions. Simulation results confirm that the partial opposition-based DE obtains a promising performance on the majority of the benchmark functions. The proposed algorithm is compared with the Opposition-based DE (ODE) and random partial opposition-based DE algorithm (DE-RPO); the results show that our new method is better than or at least comparable to other competitors.**

I. Introduction

Metaheuristic algorithms are an efficient class of the optimization methods for solving the complex real-world problems in engineering and science applications. All these algorithms have the common basic framework: they start with some random candidate solutions as the initial population. Then, in the evolutionary iteration they apply some operations to generate new trial candidate solutions. The generated trial candidate solutions compete to survive. Opposition based learning (OBL) was proposed [1], [2], [3] to improve DE algorithm with considering the current candidate solutions and their corresponding opposites simultaneously during initialization and evolutionary processes.

Generally speaking, the OBL scheme considers the current estimate (guess) and its corresponding opposite estimate (anti-guess) simultaneously to converge to an optimal solution for a given objective function in the machine learning algorithms. Many machine learning algorithms have been enhanced by utilizing this concept, such as, Reinforcement Learning (RL) [4], Artificial Neural Networks (ANN) [5], fuzzy systems [6], and variant optimization methods [1], [3], [7], like Genetic Algorithms (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO), Biogeography-Based Optimization (BBO), Harmony Search (HS), Ant Colony System (ACS), Artificial Bee Colony (ABC), Simulated Annealing (SA), discrete and combinatorial optimization, etc.

In recent years, several schemes of OBL have been developed, such as, quasi-reflection opposition [8], comprehensive opposition [9], centroid point-opposition [10], convex or concave opposition with the beta distribution [11], rotated-based learning [12], fitness-based opposition [13], partial opposition-based learning [14], opposite-center learning [15], type-II OBL [16], generalized opposition-based learning [17] and super-opposition [18]. In the mentioned schemes of OBL, all variables are selected to change into their opposite therefore it may cause that some good variables with the proper values in the original candidate solution are destroyed. The partial opposition scheme was proposed in a few research works [14], [11] to keep some variables of a candidate solution. In [14], a partial opposition-based learning (POBL) scheme was proposed by computing a set of partial opposite points for a candidate solution. In the partial opposite point, the opposition values for only some dimensions of a candidate solution are computed. In [11], they changed the subset of dimensions to their opposition values. These methods just select randomly some variables to compute their opposite. Because these partial schemes are based on a random strategy to choose variables to be coverted to their opposite, they can not guaranty that variables with proper values are still kept. In fact, all variables have an equal chance to change into their opposites. Thus, it would be interesting to develop some partial opposition schemes which can increase the change chance of some variables with the improper values into their opposite while can decrease the risk of changing the good variables with the proper values.

In this paper, we propose a new concept of the partial opposition according to identifying the closeness of variables in a

candidate solution to the variables of the current best candidate solution. In the first step, for each candidate solution of the current population, a new trial solution is generated which the close variables of a candidate solution to the current best candidate solution are unchanged and other remaining variables are selected to be converted to their opposites. After generating all new trial candidate solutions, those trial candidate solutions are included in the opposite population which their changed variables are more their unchanged variables. Then, the fittest solutions are selected from the union of two populations (i.e., the current population and the opposite population). The performance of the proposed algorithm is evaluated on CEC-2014 benchmark functions. Simulation results confirm that the proposed algorithm obtains a promising performance on the majority of functions.

The organization of the rest of the paper is as follows. Section II presents a background review. Section III describes the details of the proposed method. Section IV presents the experimental results. Finally, the paper is concluded in Section V.

II. BACKGROUND REVIEW

A. Opposition-Based Learning

Concept of opposite numbers were defined in [2] as follows.

Definition 1 (The opposite number) [2]**.** Let $x \in [a, b]$ be a real number. Its opposite number, $\breve{x}$, is defined as follow:

$$\breve{x} = a + b - x. \tag{1}$$

Fig. 1 indicates the number x and its opposite, $\breve{x}$. Similarly, the opposition concept can be extended for higher dimensions as follows.

Definition 2 (The opposite point in the D space) [2]**.** The opposition of the point $X(x_1, \ldots, x_D), x_i \in [a_i, b_i], i = 1, 2, \ldots, D$ was defined in the D-dimensional space as follow: [2]:

$$\breve{x_i} = a_i + b_i - x_i. \tag{2}$$

The fundamental goal of utilizing OBL in the opposition-based methods is the increasing of convergence rate of finding fittest solutions for a black-box problem.

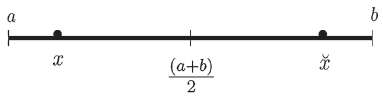


Fig. 1: The number x and its opposite $\breve{x}$ [7].

B. Opposition-based Differential Evolution (ODE)

For the first time, Rahnamayan et al. (2006) [3], [19] proposed the Opposition-based DE (ODE) which utilizes the OBL concept in the DE algorithm to improve its performance. In the initialization and evolutionary phases in the classic DE algorithm, OBL is applied; the initial population is randomly generated and simultaneously the opposite of the initial population is constructed by computing the opposite of each candidate solution. Then, the fittest solutions are selected as the initial population from the union of two populations (the initial population and its opposite). In the evolutionary process, the opposition of the current population is dynamically computed according to a jumping rate by employing the minimum and maximum values of each variable in the current population. Dynamic opposition for the candidate solution x in the evolutionary process is calculated as follow:

$$\breve{x}_{i,j} = a_i + b_i - x_{i,j}, \tag{3}$$

where a_i and b_i are current maximum and minimum values of each variable in the current population. In [7], comprehensive experiments were conducted to analyze the influence of dimensionality, opposite points, population size, various strategies of mutation, and jumping rates. In [**?**], [20], it is mathematically proved that considering the random candidate solution and its opposite has higher chance to be closer to an unknown optimal solution than two random candidate solutions.

C. Random Partial Opposition Scheme

In [14], a POBL-based adaptive DE was proposed in which for each candidate solution, a set of partial opposite solutions is generated. Then, the opposite of the candidate solution and some of partial solutions are randomly selected to compete for replacing with the original candidate solution. Park et al. [11] proposed a stochastic OBL using a beta distribution which integrated with the selection switching and the partial dimensional changing schemes. They perform the proposed stochastic OBL randomly on some dimensions to preserve some variables of the candidate solution. The random POBL gives an equal chance for all variables to change into their opposites; therefor it would be still possible to destroy variables with proper values by computing their opposites.

In [14], [11], a kind of OBL, partial opposition scheme, was proposed which in a candidate solution, some randomly selected variables are replaced with their opposites and and all other variables remain untouched. When some variables are randomly selected, the number of candidate trial solutions which can be generated from the original candidate solution and its opposite would be equal to $2^n - 2$. The values of n variables can be selected from the original candidate solution and its opposite, therefore there are two possible values for each variable. Fig. 2 indicates all possible trial solutions for the original candidate solution and its opposite in the three-dimensional space. In Fig. 2, the point $X8$ is the opposite of the point $X1$ and points $X2 - X7$ are the possible partial opposite points which can be generated by X1 and X2 as partial opposite.

III. THE PROPOSED PARTIAL OPPOSITION-BASED DE

The opposition schemes computes the opposition for all variables of a candidate solution while some variables can have close values to the corresponding variables in the optimal solution. Therefore, if the opposition is computed for all variables in a candidate solution, some variables with having proper values may be converted to their opposite values which it can take them far away from the optimal region. In the previous partial opposition schemes, they changed randomly a subset of variables into their opposite values so it is still possible that some near optimal variables become worse; in fact, the partial opposition scheme is not intelligent enough to

Algorithm 1 :**DE-POB** (NP, JR, $NFCMAX$, D)

1: //NP, JR, $NFCMAX$, and D are the population size, jumping rate, the maximum number of function evaluations, and the dimension of problem.
2: //Computing opposition-based population initialization.
3: Generating the initialization population randomly, pop.
4: Calculating the opposition population, $opop$ by using equation 1.
5: Picking NP fittest solutions from $pop \bigcup opop$ as the initialization population.
6: NFC=1.
7: **while** NFC $<$ NFCMAX **do**
8: //Run one cycle of an optimization algorithm by performing mutation, crossover, and selection operations.
9: $pop1 \Leftarrow$ **optimizer**(pop)
10: $(bestsolution, best\_val) \Leftarrow$ **evaluate**$(pop1)$
11: //Run the partial opposition according to the jumping rate.
12: $Par\_opp = []$
13: **if** $rand(0,1) < JR$ **then**
14: **for** $k = 1$ to NP **do**
15: $Iter\_Var = 0$
16: $Iter\_Var\_opp = 0$
17: $New\_trial = []$
18: **for** $j = 1$ to D **do**
19: Calculate the opposition of jth variable as $opp(j)$ by Equation (2)
20: Categorize jth variable to close and far classes based on its Euclidean distance of jth variable in the $bestsolution$
21: **if** jth variable is close **then**
22: $New\_trial(j) = pop(k, j)$
23: $Iter\_Var = Iter\_Var + 1$
24: **else**
25: $New\_trial(j) = opp(j)$
26: $Iter\_Var\_opp = Iter\_Var\_opp + 1$
27: **end if**
28: **end for**
29: //Check a solution to include the partial opposition population, Par_opp,
30: **if** $Iter\_Var < Iter\_Var\_opp$ **then**
31: $Par\_opp = pop \bigcup New\_trial$
32: **end if**
33: **end for**
34: Picking NP fittest solutions from $pop \bigcup Par\_opp$ as the current population.
35: **end if**
36: $NFC = NFC + 1$
37: **end while**

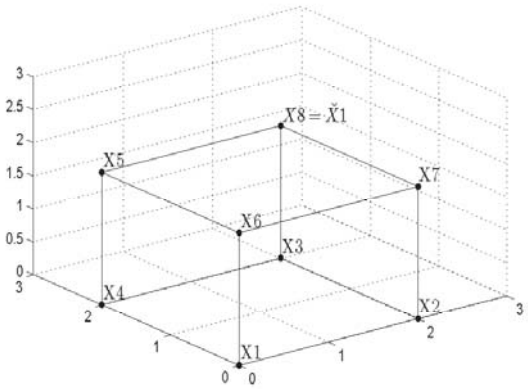


Fig. 2: The possible generated trial solutions from the original candidate solution ($X1$) and its opposite ($X8$).

target just far variables to be replaced with their opposites. As the dimension of a problem is increased, by randomly performing the opposition for only some variables the partial opposite solution would be so similar (i.e., close) to the original candidate solution.

The main goal in the proposed partial opposition, based on the current best candidate solution, is maintaining the values of those variables which are currently holding proper values; i.e., they have closer values to the unknown optimal solution than its opposite values. In fact, we keep those untouched variables which computing their opposition cannot offer the better values than their current values. In optimization steps, the obtained best solution is as a guide, i.e., the assumption is that the obtained best candidate solution so far moves toward the optimal solution (we know that it is not guaranteed), so the proposed partial opposition uses the guidance of the obtained best candidate solution. For example, in the Fig. 3 if the point X_{opt2} is supposed as the optimal solution, the original candidate solution $X1$ would be much closer than its opposition $\breve{X}1(X8)$ to X_{opt2}. In this case, the trial partial opposite points $X2 - X4$ are more likely would be selected due to their closeness to the optimal solution X_{opt2}. Also, if the point X_{opt1} is supposed as the optimal solution, the opposition point $\breve{X}1(X8)$ would be much closer than the original candidate solution $X1$ to X_{opt1}. In this case, the trial partial opposite points $X5 - X7$ are more likely will be selected due to their closeness to the optimal solution X_{opt1}.

In the following, the partial opposition-based algorithm based on the current best candidate solution is described in details (Algorithm 1). First, the initial population is randomly generated (line 3) and the opposite initial population is computed (line 4). Then, the fittest candidate solutions are selected as the initial population from the union of initial population and its opposite population (line 5). In the evolutionary process, after running the evolutionary operations, the partial opposite population is computed according to a jumping rate (lines 13-35). Variables of each candidate solution in the population, are categorized in two classes based on their closeness; close and far variables (line 20). To identify closeness of variables, first the opposite of the candidate solution $\breve{x} = (\breve{x_1}, \ldots, \breve{x_n})$ is

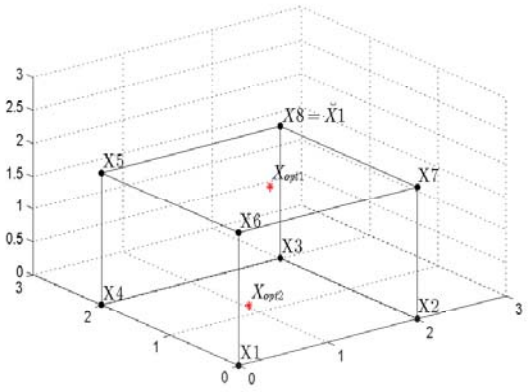


Fig. 3: The trial partial opposite points and the optimal solution.

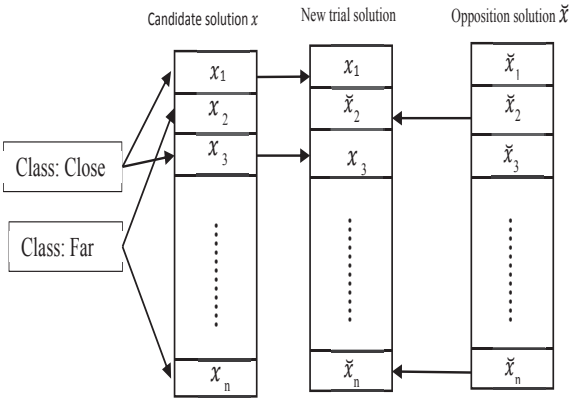


Fig. 4: The new partial opposite trial candidate solution from the original candidate solution and its opposite.

calculated which the minimum and maximum values of each dimension, a_i and b_i, in the current population are employed for calculating it:

$$\breve{x_i} = a_i + b_i - x_i. \tag{4}$$

Then, the Euclidean distance of each variable with the corresponding variable in the obtained best current candidate solution, $x_{best} = (x_{best1}, \ldots, x_{bestn})$, is calculated in the original candidate solution x and its opposite $\breve{x}$. If this Euclidean distance of each variable in the original candidate solution $dist_{x_i}$ is greater than the Euclidean distance of the corresponding variable in the opposite solution $dist_{\breve{x_i}}$ then this variable is assigned as close variable; otherwise, it is assigned as far variable. A partial opposite trial solution is created from a candidate solution x such that the values of the variables in the class close are remain unchanged while all variables in the class far are replaced with their corresponding opposite values (lines 21-27). The new partial opposite trial solution $x_t(x_{t1}, x_{t2}, \ldots, x_{tn})$ is generated as follows:

$$x_{ti} = \begin{cases} x_i & \text{if the variable } x_i \text{ is close variable,} \\ \breve{x_i} & \text{if the variable } x_i \text{ is far variable,} \end{cases} \tag{5}$$

Fig. 4 shows an example to indicate how a generated new trial candidate solution takes values for its variables from the original candidate solution and its opposite according to the class of variables; far and close. Fig. 5 indicates two partial opposite solutions for two cases; assuming the optimal solution can be X_{opt2} or X_{opt1}. In the first case, the original candidate solution $X1$ is closer to the optimal solution X_{opt2} while in the second case, its opposite $\breve{X1} = X2$ is closer to the optimal solution X_{opt1}. As it can be seen from Fig. 5, in the first case, the new trial solution $\breve{X_{p2}} = X4$ has variables which their values are taken from $X1$ while for the second case the variables of the new trial solution $\breve{X_{p1}} = X3$ are more closer to $\breve{X1}$. Partial opposite trial candidate solutions are computed for all the candidate solutions in the current population; but for constructing the partial opposite population, it includes those trial solutions which the number of their variables with the opposite values is greater than the variables with the original values (lines 29-32). The reason of selecting some partial

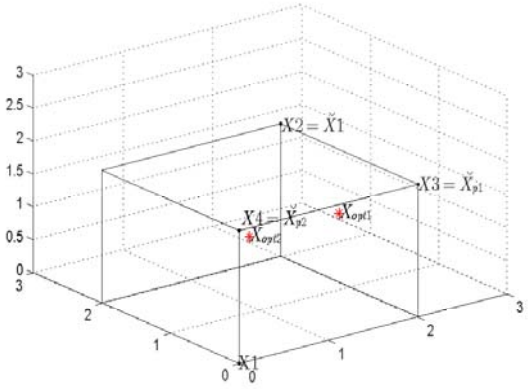


Fig. 5: Two sample cases of the partial opposite solutions.

opposite trial solutions is that new trial solutions including a significant difference with the original candidate solution are more potential to be an effective generated solution for exploring search space. In fact, the partial opposite population is created in such a way that it contains more solutions which their position is more closer to the opposite solutions. When the most of variables in the new trial solutions are similar with the candidate solutions so there is the proper representatives of them, the candidate solutions, in the population. Then, the fittest solutions are selected from the union of two populations (the current population and the partial opposite population). The proposed partial opposition scheme can be applied to every other population-based optimization algorithm, we used the classical DE (DE/1/bin) algorithm as the parent algorithm for our case study.

IV. EXPERIMENTAL RESULTS

A. Setup of experiments

To analyze the performance of the proposed partial opposition-based algorithm, we compare the partial opposition-based DE (DE/rand/bin) based on the current best candi-

TABLE I: Results of the DE-POB algorithm DE-RPO and ODE algorithms on the CEC-2014 benchmark functions. Symbols '†', '‡', and '≈' denote DE-POB are worse than, better than, or similar to the compared algorithms, respectively.

Function		ODE	DE-RPO	DE-POB
f_1	Mean	3.72561e+09‡	5.21279e+08‡	4.48823e+08
	Std	1.20999e+09	1.08989e+08	1.25750e+08
f_2	Mean	1.69937e+11‡	3.66937e+07‡	3.54716e+06
	Std	3.41760e+10	1.40192e+07	1.48829e+07
f_3	Mean	3.29004e+05‡	2.74494e+04‡	2.39510e+04
	Std	3.08450e+04	5.81691e+03	5.32654e+03
f_4	Mean	3.63397e+04‡	7.12830e+02≈	7.32972e+02
	Std	1.29091e+04	3.74495e+01	5.42602e+01
f_5	Mean	5.21350e+02≈	5.21359e+02≈	5.21343e+02
	Std	2.69298e-02	2.55112e-02	4.61353e-02
f_6	Mean	7.52724e+02‡	7.08083e+02‡	6.68462e+02
	Std	5.80407e+00	3.12403e+01	2.80166e+01
f_7	Mean	2.42592e+03‡	7.01297e+02‡	7.00503e+02
	Std	3.97986e+02	1.65061e-01	3.26848e-01
f_8	Mean	2.02808e+03‡	1.67194e+03‡	1.45921e+03
	Std	9.22851e+01	3.84690e+01	1.08019e+02
f_9	Mean	2.28349e+03‡	1.83019e+03‡	1.79912e+03
	Std	7.42268e+01	2.38865e+01	3.13257e+01
f_{10}	Mean	3.00357e+04‡	2.83363e+04‡	2.67972e+04
	Std	1.95148e+03	7.79638e+02	1.34480e+03
f_{11}	Mean	3.19373e+04≈	3.25289e+04‡	3.19433e+04
	Std	9.91834e+02	5.54688e+02	1.34986e+03
f_{12}	Mean	1.20448e+03‡	1.20444e+03≈	1.20414e+03
	Std	2.46607e-01	2.47009e-01	7.88565e-01
f_{13}	Mean	1.30678e+03‡	1.30073e+03‡	1.30062e+03
	Std	7.45499e-01	6.31482e-02	6.41723e-02
f_{14}	Mean	1.88890e+03‡	1.40046e+03‡	1.40042e+03
	Std	8.91698e+01	7.50780e-02	1.10073e-01
f_{15}	Mean	6.78229e+06‡	1.58633e+03†	1.59018e+03
	Std	4.48444e+06	2.73409e+00	5.87956e+00
f_{16}	Mean	1.64718e+03‡	1.64732e+03‡	1.64686e+03
	Std	2.73415e-01	2.46334e-01	3.38786e-01
f_{17}	Mean	4.29543e+08‡	2.52154e+07≈	2.40825e+07
	Std	1.63567e+08	8.37978e+06	7.89659e+06
f_{18}	Mean	7.67282e+09‡	3.70596e+03≈	4.14493e+03
	Std	3.64883e+09	1.69874e+03	2.60496e+03
f_{19}	Mean	3.23660e+03‡	2.00306e+03‡	2.00061e+03
	Std	4.01010e+02	4.43069e+00	4.96403e+00
f_{20}	Mean	8.20847e+05‡	5.01147e+04‡	3.80302e+04
	Std	3.51597e+05	1.39618e+04	1.33209e+04
f_{21}	Mean	1.83396e+08‡	5.27921e+06‡	4.43295e+06
	Std	6.73970e+07	1.90572e+06	1.26511e+06
f_{22}	Mean	9.73620e+03‡	6.84544e+03≈	6.76021e+03
	Std	2.44188e+03	2.82725e+02	2.72120e+02
f_{23}	Mean	3.85675e+03‡	2.64962e+03‡	2.64906e+03
	Std	2.82778e+02	5.89315e-01	6.16393e-01
f_{24}	Mean	3.04557e+03‡	2.80204e+03†	2.80628e+03
	Std	7.92355e+01	6.93415e+00	6.86070e+00
f_{25}	Mean	2.80733e+03†	2.85706e+03‡	2.82457e+03
	Std	3.78630e+01	2.99550e+01	3.16336e+01
f_{26}	Mean	2.82915e+03≈	2.85149e+03≈	2.84015e+03
	Std	4.03246e+01	1.44670e+02	9.68158e+01
f_{27}	Mean	6.93098e+03‡	4.53032e+03‡	4.02844e+03
	Std	1.90095e+02	3.14355e+02	3.28947e+02
f_{28}	Mean	1.26760e+04‡	5.90889e+03≈	6.11552e+03
	Std	2.33232e+03	3.86039e+02	5.63850e+02
f_{29}	Mean	1.35003e+08‡	4.85583e+07‡	2.53401e+07
	Std	2.18485e+07	4.63642e+07	4.16633e+07
f_{30}	Mean	2.45012e+07‡	3.20163e+04≈	2.90922e+04
	Std	1.33566e+07	7.82685e+03	5.10849e+03
	w/t/l	26/3/1	19/9/2	-

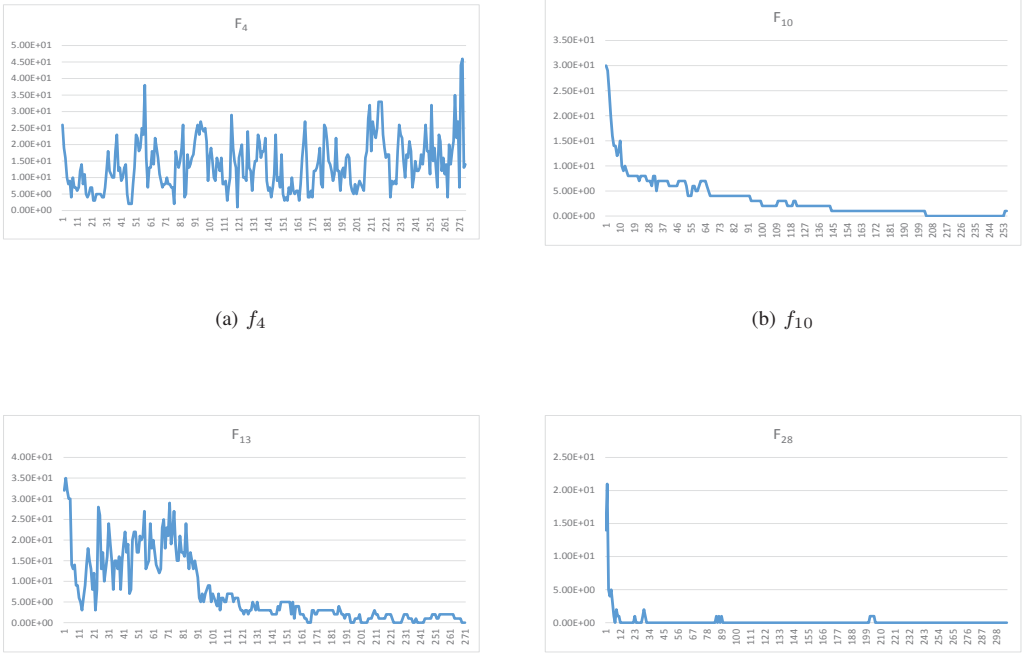
date solution, namely DE-POB, against the Opposition-based DE (ODE) [7] and the random partial opposition-based DE algorithm, namely DE-RPO, which randomly selects some variables with the probability of 0.5 to convert into their opposite values. Experiments are conducted on the CEC-2014 benchmark functions [21] with D=100. In this study, the maximum number of fitness evaluations was set to $D \times 1000$, the population size was set to 100, and the jumping rate was set to 0.3. All algorithms are evaluated over 51 independent runs and the results are recorded. A two-sided Wilcoxon statistical test with a confidence interval of 95% is performed among the compared algorithms and DE-POB algorithm. Symbols '†', '‡', and '≈' denote the compared algorithm are worse than, better than, or similar to DE-POB algorithm, respectively. *"w/t/l"* in the last row in tables means that the compared frameworks wins in *w* functions, ties in *t* functions, and loses in *l* functions, compared with DE-POB algorithm.

B. Numerical Results

The mean and the standard deviation of the obtained error values by DE-POB, ODE and DE-RPO algorithms are summarized in the Table I. As it can be seen from the Table I, DE-RPO perform better than DE-POB and ODE on 26 (f_1-f_4,$f_{21} - f_{24}$, and $f_{27} - f_{30}$) and 19 (f_1-f_3,$f_6 - f_{11}$,$f_{13} - f_{14}$,f_{16},f_{19}-f_{21},f_{23},f_{25} and f_{29}) functions, respectively. The results of DE-RPO are worse than DE-POB and ODE on 1 (f_{25}), and 2 (f_{15} and f_{24}) functions, respectively. DE-RPO achieve the same results in comparison with DE-POB and ODE on 5 and 15 other functions. To gain a better understanding of the behavior of the DE-POB algorithm and DE-RPO and ODE algorithms, we plot the convergence graph on six selected problems in Fig.7. From the results, it can be seen that DE-RPO algorithm is significantly better than other algorithms. In Fig.6, we plot the number of the survive partial opposite solutions based on the current best candidate solution in cycles which the partial opposition is performed for some functions. As it can been seen from Fig.6, the plots of functions f_{10} and f_{13} indicate that the number of survive trial solutions is large in the earlier cycles and it is decreased in the last cycles.

V. Conclusions and Future Directions

In this paper, the partial opposite concept by using current best candidate solution was introduced. In the partial opposition-based algorithm, variables are divided in two groups, close and far, according to their Euclidean distance with the corresponding variables in the obtained best solution so far. The partial opposition scheme maintains the close variables while the far variables are changed to their opposite values. After generating new trial solutions, some trial solutions are consider to evaluate; those solutions with having more changed variables than the unchanged variables. The performance of the partial opposition-based algorithm was evaluated on CEC-2014 benchmark functions with D=100. The proposed algorithm was compared with the Opposition-based DE (ODE) and random partial opposition-based DE algorithm (DE-RPO). The performance of the proposed algorithm is superior to or at least comparable with the compared algorithms. In future, we are planning to design the new schemes of the partial opposition to use the advantages of other types of OBL such as quasi-reflection and centroid opposition types. Furthermore, we will investigate scalability analysis of this proposed scheme to solve large-scale problems.



(a) f_4 (b) f_{10}

(c) f_{13} (d) f_{28}

Fig. 6: The number of the survive partial opposite solutions based on the current best candidate solution in corresponding iterations. The vertical axis is the number of the survive partial opposite solutions.

REFERENCES

[1] S. Rahnamayan, "Opposition-based differential evolution," Ph.D. dissertation, 2007.

[2] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *null*. IEEE, 2005, pp. 695–701.

[3] S. Rahnamayan, H. R. Tizhoosh, and M. Salama, "Opposition-based differential evolution algorithms," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*. IEEE, pp. 2010–2017.

[4] H. R. Tizhoosh, "Reinforcement learning based on actions and opposite actions," in *International conference on artificial intelligence and machine learning*, vol. 414, 2005, pp. 906–918.

[5] M. Ventresca and H. R. Tizhoosh, "Improving the convergence of backpropagation by opposite transfer functions," in *Neural Networks, 2006. IJCNN'06. International Joint Conference on*. IEEE, 2006, pp. 4777–4784.

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

[7] S. Rahnamayan, H. R. Tizhoosh, and M. Salama, "Opposition-based differential evolution," *Evolutionary Computation, IEEE Transactions on*, vol. 12, no. 1, pp. 64–79, 2008.

[8] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Quasi-oppositional differential evolution," in *2007 IEEE Congress on Evolutionary Computation*. IEEE, 2007, pp. 2229–2236.

[9] Z. Seif and M. B. Ahmadi, "Opposition versus randomness in binary spaces," *Applied Soft Computing*, vol. 27, pp. 28–37, 2015.

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

[11] S. Park and J. Lee, "Stochastic opposition-based learning using a beta distribution in differential evolution." *IEEE transactions on cybernetics*, 2015.

[12] H. Liu, Z. Wu, H. Li, H. Wang, S. Rahnamayan, and C. Deng, "Rotation-based learning: A novel extension of opposition-based learning," in *PRICAI 2014: Trends in Artificial Intelligence*. Springer, 2014, pp. 511–522.

[13] M. Ergezer and D. Simon, "Probabilistic properties of fitness-based quasi-reflection in evolutionary algorithms," *Computers & Operations Research*, vol. 63, pp. 114–124, 2015.

[14] Z. Hu, Y. Bao, and T. Xiong, "Partial opposition-based adaptive differential evolution algorithms: evaluation on the cec 2014 benchmark set for real-parameter optimization," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 2259–2265.

[15] H. Xu, C. D. Erdbrink, and V. V. Krzhizhanovskaya, "How to speed up optimization? opposite-center learning and its application to differential evolution," *Procedia Computer Science*, vol. 51, pp. 805–814, 2015.

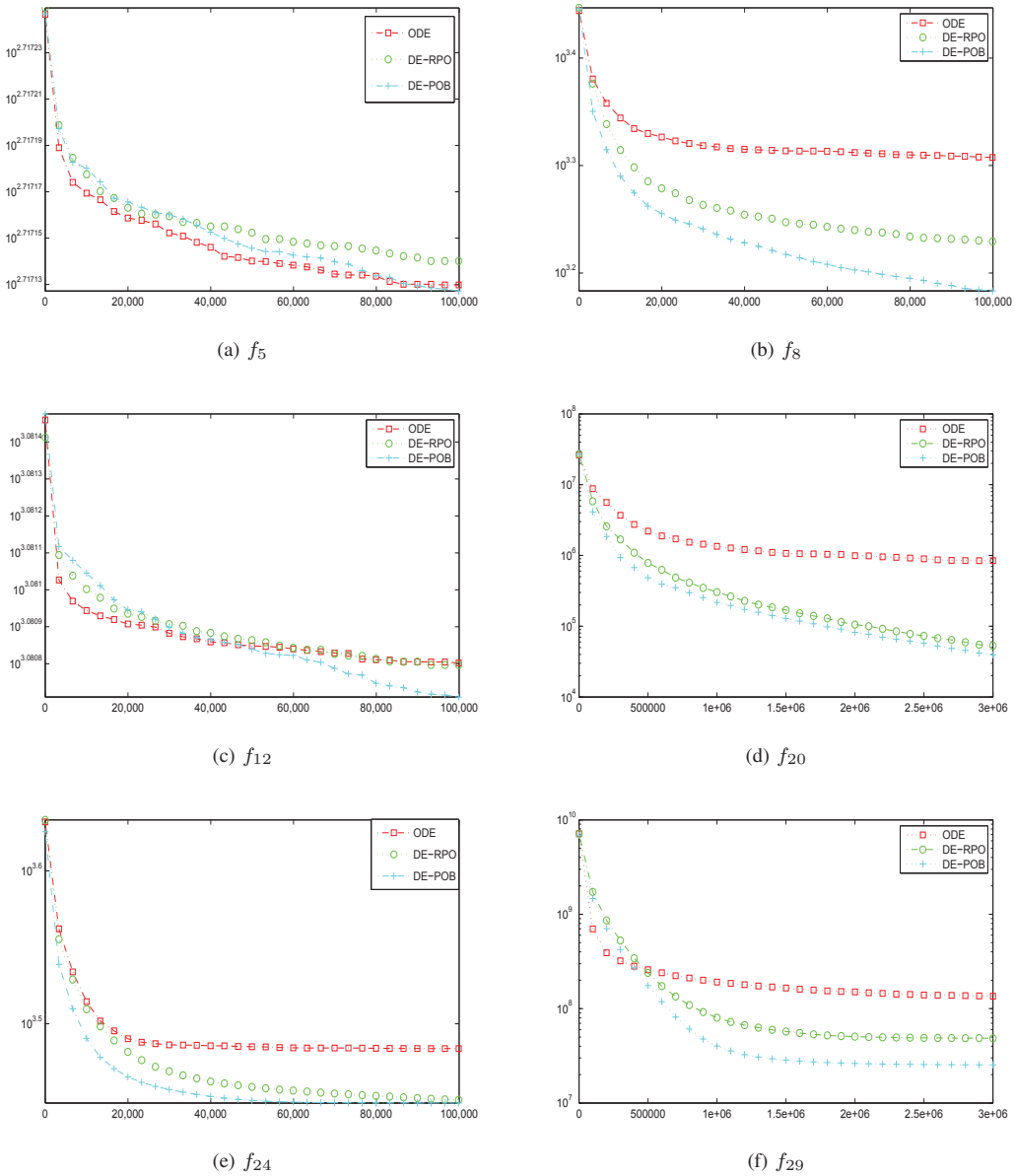
[16] H. Salehinejad, S. Rahnamayan, and H. R. Tizhoosh, "Type-ii opposition-based differential evolution," in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 1768–1775.

[17] H. Wang, Z. Wu, and S. Rahnamayan, "Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems," *Soft Computing*, vol. 15, no. 11, pp. 2127–2140, 2011.

[18] H. Tizhoosh and M. Ventresca, "Oppositional concepts in computational intelligence," *Studies in Computational Intelligence. Berlin, Germany: Springer*, vol. 155, 2008.

[19] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "A novel population initialization method for accelerating evolutionary algorithms," *Computers & Mathematics with Applications*, vol. 53, no. 10, pp. 1605–1614, 2007.

[20] S. Rahnamayan, G. G. Wang, and M. Ventresca, "An intuitive distance-



(a) f_5 (b) f_8

(c) f_{12} (d) f_{20}

(e) f_{24} (f) f_{29}

Fig. 7: Convergence plots of f_5, f_8,f_{12},f_{20},f_{24}, and f_{29} of the CEC-2014 benchmark function. The results were averaged over 51 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations.

based explanation of opposition-based sampling," *Applied Soft Computing*, vol. 12, no. 9, pp. 2828–2839, 2012.

[21] J. Liang, B. Qu, and P. Suganthan, "Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization," *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, 2013.