Received November 20, 2018, accepted December 11, 2018, date of publication January 2, 2019, date of current version January 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2890402

Exploring the Reasons Behind the Good Performance of Opposition-Based Learning

QINGZHENG XU[1,2], NA WANG[1], FENG ZOU[3], AND JUNGANG YANG[1]

[1]College of Information and Communication, National University of Defense Technology, Xi'an 710106, China
[2]School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798
[3]School of Physics and Electronic Information, Huaibei Normal University, Huaibei 235000, China

Corresponding author: Qingzheng Xu (xuqingzheng@hotmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572224, in part by the China Scholarship Council under Grant 201703170064, in part by the Scientific Research Foundation of the National University of Defense Technology under Grant ZK18-03-43, and in part by the Anhui Provincial Natural Science Foundation under Grant 1708085MF140.

ABSTRACT In the face of these diverse forms of opposition existed widely in real-world contexts, as a novel concept in computational intelligence, opposition-based learning (OBL) was originally introduced to accelerate the population-based algorithm. In addition, its superiority has been proved mathematically and experimentally. Two key elements (function prototype and algorithm design) that influence the algorithm performance, however, has not yet fully discussed in the past decade. In this paper, two OBL strategies are reexamined in respect of function prototype and algorithm design. In the first part of this paper, considering the position relationship between the optimal solution and the center point, some well-known benchmark functions are divided into three categories. Then, quasi-opposition-based differential evolution (QODE) is investigated by two approaches: solving several benchmark functions of various function types and solving the same functions with a different optimal solution. The numerical experiments reveal that ''smart'' matching between the benchmark functions and the QOBL is an important factor to the good performance of QODE. In the second part of this paper, a novel individual-based embedding method is proposed to coincide with the classical definition of opposition-based optimization. Then, two opposition-based differential evolution algorithms are compared to discuss the differences between the two embedding methods. The experimental results confirm that the convergence differences stem from the embedding method chosen in the OBL scheme rather than the utilization rate of opposite points. Furthermore, the impacts caused by various function types and jumping rate are also discussed.

INDEX TERMS Opposition-based learning, differential evolution, function type, embedding method, opposite point, convergence speed.

I. INTRODUCTION

Inspired from the opposition concept in real-world contexts, such as opposite particle in physics, antonym in language, and subject/object in philosophy, the basic concept of opposition-based learning (OBL) was originally introduced and simply defined by Tizhoosh [1]. The key idea of this optimization strategy is, considering an estimate and its corresponding opposite estimate simultaneously can be beneficial to find a better candidate solution for an optimizer. As a novel research field, it has already attracted a recognizable interest in machine learning and artificial intelligence since 2005 [2], [3].

In the past decade, it was extended rapidly and several new and potent oppositional strategies were proposed to improve solution quality, such as quasi-opposition-based learning (QOBL) [4], quasi-reflection opposition-based learning (QROBL) [5], center-based sampling [6], generalized opposition-based learning (GOBL) [7], opposition-based learning using the current optimum (COOBL) [8], partial opposition-based learning (POBL) [9], rotated-based learning (RBL) [10], opposite-center learning (OCL) [11] and comprehensive opposition (CO) [12]. Meanwhile, these OBL strategies were further successfully applied in swarm and evolutionary computing algorithms to solve various science and engineering problems, such as power system, pattern recognition and image processing, identification, bioinformatics and medicine, etc [3].

A lot of research works have mathematically and experimentally proved that utilizing the opposition concept in the framework of swarm and evolutionary computing algorithms

can enhance (maintain at least) their performance. For a long time, some consensus have been built in this field. First of all, it is the opposition-based strategy that can improve the performance of swarm and evolutionary computing algorithms [4]–[13]. More important, there is continuing improvement in algorithm performance, with more opposite points survived to the child population during the whole evolution process [8], [13], [14]. The key goal of this paper is to better understand the true reasons behind the good performance and disambiguate some misunderstanding of OBL. Thus OBL and QOBL strategies are reexamined in respects of function prototype and algorithm design in this paper.

The contributions of this paper to this field can be listed as follows. First, a function classification principle is introduced and applied to some well-know benchmark functions. Second, the real reason for the good performance of QOBL is revealed. We think that it is the ''smart'' matching between it and benchmark functions, rather than the strategy itself. Third, an individual-based embedding method is proposed as a base to develop opposition-based swarm and evolutionary computing algorithms. Finally, a more plausible explanation of the convergence differences between two opposition-based differential evolution (ODE) is given to reclaim the previous knowledge. We believe it is the embedding method chosen in OBL scheme, rather than the utilization rate of opposite points.

The remainder of this paper is organized as follows. The concept of opposition-based learning and opposition-based differential evolution are reviewed in Section II. Three benchmark function categories are introduced and the influences of function type and optimal solution on algorithm convergence are discussed in Section III. Section IV contains a new embedding method of OBL scheme and further experimental results of two ODE algorithms over 58 optimization problems. Finally, Section V concludes this paper.

II. PRELIMINARIES

A. CONCEPT OF OPPOSITION-BASED LEARNING

The definitions of OBL and QOBL studied in this paper are given as follows.

Let $P = (p_1, p_2, \ldots, p_D)$ be an arbitrary point in D-dimensional space, where $p_1, p_2, \ldots, p_D \in \boldsymbol{R}$ and $p_i \in [a_i, b_i]$, $\forall i \in \{1, 2, \ldots, D\}$. The opposite point $\breve{P} = (\breve{p}_1, \breve{p}_2, \ldots, \breve{p}_D)$ [13] and quasi-opposite point $\breve{P}_q = (\breve{p}_{q1}, \breve{p}_{q2}, \ldots, \breve{p}_{qD})$ [4] of P can be completely defined by its coordinates, respectively

$$\breve{p}_i = a_i + b_i - p_i \tag{1}$$

$$\breve{p}_{qi} = \text{rand}(c_i, \breve{p}_i) \tag{2}$$

where c_i (calculated as $\frac{a_i+b_i}{2}$) is the center of the search interval $[a_i, b_i]$, and $\text{rand}(c_i, \breve{p}_i)$ is a random number uniformly distributed between c_i and $\breve{p}_i$.

As an example, now we can define opposition-based optimization as follows. Similar to it, quasi-opposition-based optimization can also be defined by using the concept of QOBL given above.

Opposition-Based Optimization: Let $P = (p_1, p_2, \ldots, p_D)$ be a point in an D-dimensional space and $\breve{P} = (\breve{p}_1, \breve{p}_2, \ldots, \breve{p}_D)$ be an opposite point of point P. Assume $f(\cdot)$ is a fitness function which is used to measure the candidate's fitness. Now, if $f(\breve{p}) < f(p)$, then point P can be replaced with $\breve{P}$; otherwise we continue with P. Hence, the optimization process continues with the fitter one.

Step 1 Opposition-based Population Initialization
Set generation number $G = 0$, and randomly initialize a population $P_0 = \{X_{1,0}, X_{2,0}, \cdots, X_{Np,0}\}$
FOR $i = 1$ to N_p
  $OP_{i,0} = a+b-X_{i,0}$
END FOR
Select N_p fittest individuals from current population (P_0) and opposition-based population (OP_0)
Step 2 WHILE stopping criterion is not satisfied DO
  Step 2.1 Mutation
  /* Generate a mutated vector $V_{i,G}$ for each target vector $X_{i,G}$ */
  FOR $i = 1$ to N_p
    Select three parents $X_{i1,G}$, $X_{i2,G}$, and $X_{i3,G}$ randomly from current population where $i \neq i_1 \neq i_2 \neq i_3$
    $V_{i,G} = X_{i1,G} + F \times (X_{i2,G} - X_{i3,G})$
  END FOR
  Step 2.2 Crossover
  /* Generate a trial vector $U_{i,G}$ for each target vector $X_{i,G}$ */
  FOR $i = 1$ to N_p
    $j_{rand} = rand(0,1) \times D$
    FOR $j = 1$ to D
      IF ($rand_j(0,1) < C_R$) or ($j = j_{rand}$)
        $U_{i,j,G} = V_{i,j,G}$
      ELSE
        $U_{i,j,G} = X_{i,j,G}$
      END IF
    END FOR
  END FOR
  Step 2.3 Selection
  FOR $i = 1$ to N_p
    Evaluation the trial vector $U_{i,G}$
    IF $f(U_{i,G}) < f(X_{i,G})$
      $X_{i,G+1} = U_{i,G}$
    ELSE
      $X_{i,G+1} = X_{i,G}$
    END IF
  END FOR
  Step 2.4 Opposition-based Generation Jumping
  IF $rand(0, 1) < J_r$
    FOR $i = 1$ to N_p
      $OP_{i,G} = MIN_G + MAX_G - X_{i,G}$
    END FOR
    Select N_p fittest individuals from current population (P_G) and opposition-based population (OP_G)
  END IF
  $G = G + 1$
END WHILE

FIGURE 1. Pseudo code for opposition-based differential evolution (ODE).

B. OPPOSITION-BASED DIFFERENTIAL EVOLUTION

From statistical results in [2], differential evolution (DE) was widely used as a basic meta-heuristic algorithm to reveal good performance of OBL strategies. Simple design and easy

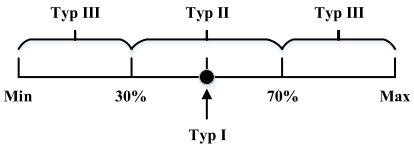


FIGURE 2. Categories of benchmark functions.

implementation, two main preferences of DE than other soft computing algorithms, are a basic reason for widespread application of DE in the last decade.

In [13], OBL was embedded in the classical DE, DE/rand/1/bin, and then opposition-based differential evolution was developed to accelerate its convergence speed. The corresponding pseudo code is given in Fig. 1. Note that two steps (population initialization and generation jumping) are enhanced using OBL scheme. What is important is that alternative oppositional DE can also be developed easily by embedding other opposition-based strategies in two steps. The details are not yet covered here.

III. EXPERIMENTAL ANALYSIS OF QUASI-OPPOSITION-BASED LEARNING IN RESPECT OF FUNCTION PROTOTYPE

A. MOTIVATION

The properties of benchmark functions, such as function notation, dimension of function spaces, domain and range, concave/convex, turning points and extremal points, as we all know, have great influence on algorithm performance. If the tested algorithm is an excellent match for the function prototype, its performance may be overestimated. It is unfair to other algorithms, which do not fit these function characters and may fit others well, and ultimately leads to questionable results.

A benchmark function set (or its part) in the pioneer work [13] was widely employed to measure performance in most of opposition-based research works [4], [5], [8], [11]. The utilized test suite includes 58 well-known unimodal as well as highly multimodal optimization problems. The dimensionality varying from 2 to 30 covers a wide range of problem complexity. The detailed definition of these benchmark functions are presented in Appendix A of paper [13], together with their global optimums, that can be used directly readers.

After careful observation of 58 optimization problems, we found that some optimal solutions are just located at the center of the function domain, and some close to the center and some keep away from the central zone. In our view, the position relationship between optimal solution and center point of function spaces is a very important, but yet often overlooked property of benchmark functions. In this paper, according to the position of the optimal solution in function domain, these benchmark functions can be divided into three categories as illustrated in Fig. 2: locating at the center (Type I), closing to the center (Type II) and keeping away from the central zone (Type III). If the optimal solution of multidimensional function closes to the center only for some dimension, the function is

TABLE 1. Name and type of optimization problems.

Function	Name	Dimension	Type
F1	Sphere model	30	I
F2	Axis parallel hyperellipsoid	30	I
F3	Schwefel's problem 1.2	20	I
F4	Rosenbrock's valley	30	III
F5	Rastrigin's function	10	I
F6	Griewangk's function	30	I
F7	Sum of different power	30	I
F8	Ackley's path function	30	I
F9	Beale function	2	II (part)
F10	Colville function	4	II (whole)
F11	Easom function	2	II (whole)
F12	Hartmann function 1	3	II (part)
F13	Hartmann function 2	6	II (part)
F14	Six Hump Camel back function	2	II (whole)
F15	Levy function	30	II (whole)
F16	Matyas function	2	I
F17	Perm function	4	III
F18	Michalewicz function	10	II (part)
F19	Zakharov function	30	III
F20	Branins's function	2	III
F21	Schwefel's problem 2.22	30	I
F22	Schwefel's problem 2.21	30	I
F23	Step function	30	I
F24	Quartic function	30	I
F25	Kowalik's function	4	II (whole)
F26	Shekel's Family (5)	4	II (part)
F27	Shekel's Family (7)	4	II (part)
F28	Shekel's Family (10)	4	II (part)
F29	Tripod function	2	II (part)
F30	De Jong's function 4 (no noise)	30	I
F31	Alpine function	30	I
F32	Schaffer's function 6	2	I
F33	Pathological function	3	I
F34	Inverted cosine wave function (Masters)	5	I
F35	Aluffi-Pentini's Problem	2	II (whole)
F36	Becker and Lago Problem	2	III
F37	Bohachevsky 1 Problem	2	I
F38	Bohachevsky 2 Problem	2	I
F39	Camel Back-3 Three Hump Problem	2	I
F40	Dekkers and Aarts Problem	2	II (part)
F41	Exponential Problem	10	I
F42	Goldstein and Price Problem	2	II (part)
F43	Gulf Research Problem	3	II (part)
F44	Helical Valley Problem	3	II (whole)
F45	Hosaki Problem	2	III

F46	Levy and Montalvo 1 Problem	3	II (whole)
F47	McCormick Problem	2	III
F48	Miele and Cantrell Problem	4	II (part)
F49	Multi-Gaussian Problem	2	II (whole)
F50	Neumaier 2 Problem	4	II (part)
F51	Odd Square Problem	10	II (part)
F52	Paviani's Problem	10	III
F53	Periodic Problem	2	I
F54	Powell's Quadratic Problem	4	I
F55	Price's Transistor Modeling Problem	9	II (part)
F56	Salomon Problem	10	I
F57	Schaffer 2 Problem	2	I
F58	Wood's Function	4	II (whole)

considered as Type II (part); otherwise it belongs to Type II (whole). Thus these optimization problems are concluded in Table 1.

Rahnamayan *et al.* [4] proved mathematically that, for a black-box optimization problem, quasi-opposite point has a higher chance than opposite point to be closer to the unknown optimal solution. Furthermore, experimental results clearly illustrate that quasi-opposition-based differential evolution (QODE) outperforms ODE and DE.

By definition of (2), quasi-opposite point is a uniform random point generated between center point and opposite point. As we can imagine, quasi-opposite point must be closer to the optimal solution which is the center exactly (or near the center) of function domain. Even it is possible for quasi-opposition-based algorithm to find the optimal solution of optimization problem directly (or almost directly).

Coincidentally, for most of benchmark functions tested, their optimal solutions are the center exactly (Type I), or near the center (Type II) as shown in Table 1. QODE algorithm can make full use of this function prototype incidentally or accidentally. As a result, this is nothing to be surprised at the good performance of QODE to solve these optimization problems. We have reason to suspect that it is ''smart'' matching between benchmark functions and QOBL embedded in DE, that always leads to the anticipant result.

The main purpose of this section is to check our conjecture above. Obviously, sharply changing the function property may achieve ''stupid'' matching, resulting in poor algorithm performance. We have two approaches to verify what we suspect specifically. Firstly, when solving different types of benchmark functions, QODE may show the biased performance. Secondly, when solving the same function with different optimal solutions, it may degrade greatly with diverging from the domain center to the domain vertices.

B. EXPERIMENTAL SETUP

In the next two subsections, QODE will be reexamined on several benchmark function sets and the only difference between these function sets is the position of their optimal solutions. All basic optimization problems are borrowed from [13].

By translating the function definition with respect to each dimension, the position of optimal solution will vary accordingly. For $f_1(x) = \sum_1^n x_i^2$, the new function is defined as $f_1^*(x) = \sum_1^n (x_i - T_r(MAX_i - MIN_i))^2$, in which T_r is a translation factor, and MAX_i and MIN_i are minimum and maximum values for ith dimension of $f_1(x)$, respectively. At this time, the position of optimal solution is translated from $[0, 0, \cdots, 0]$ to $[T_r(MAX_1 - MIN_1), T_r(MAX_2 - MIN_2), \cdots, T_r(MAX_n - MIN_n)]$. As you can see from Fig. 2 and Table 1, for most of optimization problems, optimal solution falls in between 30% to 70% of function domain. In order to avoid crossing the boundary, translation factor α is set as a constant value less than 30% in this study.

As listed below, parameter settings are also borrowed from [4] and [13] and the values stay the same for all conducted experiments. The termination criterion is to find a value smaller than the value-to-reach (VTR) before reaching the maximum number of function evaluations (MAX_{NFE}).

- Population size, $N_p = 100$
- Differential amplification factor, $F = 0.5$
- Crossover probability, $C_r = 0.9$
- Jumping rate, $J_r = 0.05$
- Mutation strategy: DE/rand/1/bin (classic version of DE)
- Maximum NFE, $\text{MAX}_{\text{NFE}} = 10^6$
- Value to reach, $\text{VTR} = 10^{-8}$

In this study, convergence speed is measured by counting the number of function evaluations (NFE) until the algorithm reaches the predetermined VTR, which is the most commonly used metric in literatures. A smaller NFE means a higher convergence speed.

Success rate (SR) used to evaluate the algorithm reliability is the ratio of runs where the algorithm successfully reaches the VTR. Low success rate means that more runs are needed for algorithm to get a significant result.

Utilization rate of opposite points (UR) is another comparison site in this study. It is the ratio of opposite points inherited to the next generation through fitness selection. Large UR means that opposite points have more effects on survival of candidate solutions from parent population to child population.

It is noted that, in order to minimize the effect of stochastic nature on each measured metric, the reported result for each function and algorithm is the average over 100 trials.

C. EXPERIMENT SERIES 1: SOLVING BENCHMARK FUNCTIONS WITH DIFFERENT TYPES

Experimental results (NFE and SR) of applying DE, ODE, and QODE to solve 58 optimization problems are given

in Table 2. To make the table concise, the self-explanatory SR (one or zero) is omitted here. The best NFE for each case are highlighted in boldface.

According to statistical data listed in the last row of Table 2, QODE, ODE and DE get 138, 14 and 31 best NFEs for all cases, respectively. It can be concluded that, as stated

TABLE 2. Experimental comparison of DE, ODE, and QODE.

Function	Type	$T_r = 0$			$T_r = 10\%$			$T_r = 20\%$			$T_r = 30\%$		
		DE	ODE	QODE	DE	ODE	QODE	DE	ODE	QODE	DE	ODE	QODE
F1	I	73502	66648	**37001**	73596	66971	**37210**	74029	67753	**39232**	75808	69765	**40974**
F2	I	81417	73248	**41421**	81587	74014	**41932**	81945	74953	**43949**	83525	76866	**45585**
F3	I	153992	155951	**104901**	154459	156557	**108746**	153845	156335	**109109**	152550	155184	**110447**
F5	I	326824 (0.94)	**120489** (0.93)	142545 (0.95)	301624 (0.83)	**123947** (0.93)	152311	303983 (0.9)	**121062** (0.89)	160957 (0.96)	288999 (0.98)	**116198** (0.87)	192050 (0.95)
F6	I	77280	69962	**39441**	77344	69923	**39339**	77990	71032	**41314**	79811	73165	**43418**
F7	I	19159	14277	**6383**	19200	15021	**6500**	20317	15959	**7502**	22139	17766	**10444** (0.98)
F8	I	143902	131290	**74135**	144457	130942	**74832**	144514	132834	**76659** (0.98)	146913	135235	**79516**
F16	I	2686	2717	**2480**	2690	2772	**2544**	2729	2761	**2598**	2767	2823	**2692**
F21	I	155837	156449	**86377**	156048	158655	**90218**	157643	160247	**97490**	163933	167923	**106684**
F22	I	-	**173843** (0.79)	-	-	**173233** (0.85)	-	-	**175446** (0.82)	-	-	**176817** (0.79)	-
F23	I	19712	18128	**8581**	19485	18383	**10198**	20221	19027	**10859**	21682	21174	**14064**
F24	I	-	-	-	-	-	-	-	-	-	-	-	-
F30	I	41952	32694	**17138**	42022	32668	**18161**	42861	33717	**21691**	44539	35830	**26039**
F31	I	345115	326449	**246268**	349664	342311	**241549**	345431	345770	**246175**	345968	348663	**273361**
F32	I	4952	4788	**4186**	4894	4800	**4291**	4979	4819	**4470**	4944	4899	**4619**
F33	I	25696	**23761**	26063	25655	**24121**	25947	25565	**23530**	25833	25423	**23605**	25531
F34	I	31255	27523	**24719**	30673	28816	**24645**	29863	30204	**28180** (0.98)	30725	30620	**29815**
F37	I	4129	4185	**3811**	4126	4212	**3854**	4181	4198	**3894**	4106	4219	**4011**
F38	I	4088	4172	**3831**	4078	4163	**3857**	4115	4156	**3970**	4126	4161	**4011**
F39	I	2909	2935	**2558**	2848	2901	**2683**	2889	2960	**2748**	2899	2944	**2860**
F41	I	16042	15832	**10976**	16068	15791	**10911**	16144	15759	**11594**	16270	16055	**11990**
F53	I	6707	6004	**4999**	6516	6487	**6410**	7066	7068	**6994**	7935	**7556**	7754
F54	I	7407	7362	**6652**	7368	7448	**6699**	7407	7517	**6940**	7326	7482	**7112**
F56	I	-	-	-	-	-	-	-	-	-	-	-	-
F57	I	15472	15579	**14387**	14471	14444	**13460**	14233	14300	**13439**	14282	14097	**13532**
F10	II(whole)	**13731**	14203	13852	**13813**	14217	14514	**13421**	13895	14487	**13649**	13832	13968
F11	II(whole)	5348	5356	**4779**	5299	5407	**5079**	**5407**	5567	5439	**5595**	5687	5683
F14	II(whole)	4866	4973	**4841**	**4705**	4790	4898	4885	4864	**4825**	**4824**	4993	4940
F15	II(whole)	81126 (0.94)	74185	**41506** (0.82)	81219 (0.95)	74875	**42413**	82694	76222	**44820**	85065	79849	**49751** (0.86)
F25	II(whole)	9753	9997	**9660**	10356 (0.98)	10909 (0.98)	**10199**	**13267** (0.09)	13439 (0.23)	13320 (0.05)	-	-	-
F35	II(whole)	3032	3082	**2827**	2989	3070	**2798**	3048	3086	**3000**	3053	3066	**3019**
F44	II(whole)	5501	5539	**4912**	5529	5554	**5068**	5527	5514	**5223**	5583	5660	**5312**
F46	II(whole)	4781	4905	**4209**	4770	4847	**4263**	4836	4847	**4250**	4825	4894	**4194**
F49	II(whole)	4430 (0.91)	4633 (0.95)	**4245** (0.98)	4391 (0.95)	4570 (0.92)	**4199**	4345 (0.93)	4396 (0.95)	**4312** (0.9)	4119 (0.86)	4218 (0.91)	**4094** (0.87)
F58	II(whole)	14092	13789	**13744**	**13737**	14221	14322	**14091**	14140	14308	**13570**	13986	13992
F9	II(part)	3209	3398	**3188**	**3194**	3370	3208	**3229**	3323	3237	**3194**	3277	3286
F12	II(part)	4113	4178	**3979**	4124	4168	**3964**	4091	4158	**4004**	4115	4166	**4045**
F13	II(part)	11237 (0.95)	11436 (0.89)	**9168**	10809 (0.95)	11004 (0.95)	**9334**	10836 (0.94)	11029 (0.96)	**9319** (0.98)	10774 (0.95)	11436 (0.97)	**9337**
F18	II(part)	184888 (0.5)	187213 (0.54)	**181206** (0.49)	190934 (0.32)	186409 (0.44)	**181138**	**179317** (0.52)	180616 (0.57)	183344 (0.52)	161198 (0.48)	163226 (0.5)	**157158** (0.4)

F26	II(part)	9456	9720	**7823**	9457	9692	**7292**	9049	9101	**7376**	**8889**	9048	8328
F27	II(part)	8932	9069	**7653**	8759	8871	**7133**	8524	8618	**7364**	8307	8492	**7825**
F28	II(part)	9026	9013	**7702**	8853	9039	**7149**	8686	8790	**7384**	8424	8429	**7774**
F29	II(part)	8315 (0.98)	8608	**7892**	8380 (0.97)	8670	**8179**	8311	8431	**8107**	**7746**	7907	7839
F40	II(part)	**7190**	7437	7263	**7234**	7424	7378	**6985**	7395	7514	**7070**	7350	7524
F42	II(part)	3567	3630	**3450**	3529	3612	**3471**	3523	3614	**3441**	3518	3570	**3437**
F43	II(part)	**5517**	5716	5646	**5463**	5640	5587	**5340**	5588	5512	**5227**	5524	5406
F48	II(part)	**2496**	2525	2519	**2437**	2543	2555	**2490**	2573	2534	2545	2522	**2503**
F50	II(part)	240239	**236650**	252883	**235366**	248633	238174	242689	234347	**234236**	**206445**	227053	231797
F51	II(part)	-	-	-	-	-	-	-	-	-	-	-	-
F55	II(part)	-	-	-	-	-	-	-	-	-	-	-	-
F4	III	**356748**	357083	947650 (0.06)									
F17	III	-	-	-									
F19	III	362274	358601	**229581**									
F20	III	6250	6844	**6013**					None				
F36	III	**7489**	7736	8200									
F45	III	2558	2659	**2494**									
F47	III	2706	2672	**2596**									
F52	III	21962	21929	**17860**									
Type I		0	3	20	0	3	20	0	3	20	0	4	19
Type II(whole)		1	0	9	3	0	7	4	0	6	4	0	5
Type II(part)		3	1	9	5	0	8	5	0	8	6	0	7
Type III		(2)	(0)	(5)					(None)				
Total		4	4	38	8	3	35	9	3	34	10	4	31

in [4], QODE performs better than ODE and DE on most benchmark functions in convergence speed. Furthermore, for most problems, except F22, these algorithms can obtain the global optimal solution in limited function evaluations by the similar success rate. As a result, the dominant difference between them is in convergence speed.

In order to distinguish the convergence speed intuitively and quantitatively, acceleration rate (AR) is defined as follows, based on their NFEs for two algorithms:

$$\mathrm{AR}_{1:2} = \frac{\mathrm{NFE}_2 - \mathrm{NFE}_1}{\min\{\mathrm{NFE}_1, \mathrm{NFE}_2\}} \tag{3}$$

where $\mathrm{AR}_{1:2} > 0$ means algorithm 1 can solve the problem with less NFE (e.g. high convergence speed).

Please note that the definition of AR is not the same as that in [13]. In the previous paper, AR is straightly defined as the ratio between the NFEs for two algorithms. In some cases, it is possible to produce contradictory results. For instance, the results on two functions are recorded in Table 3.

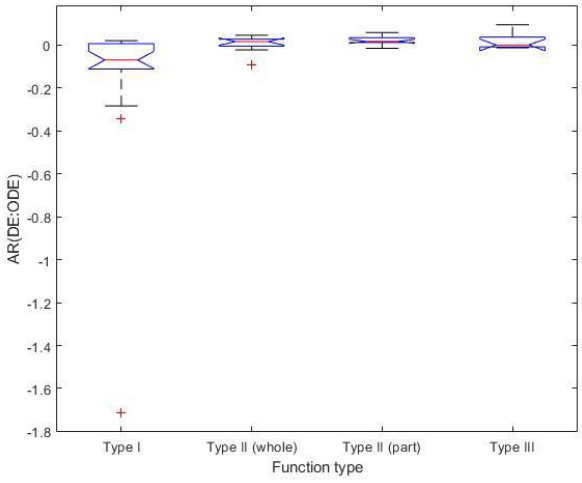
From Table 3, algorithm 1 converges faster on F2, while algorithm 2 converges faster on F1 with the same rate. Ignoring other factors, two algorithms should have the same convergence speed as a whole. Unfortunately, the value of $\mathrm{AR}_{1:2}$ defined in [13] cannot support this obvious conclusion. On the contrary, $\mathrm{AR}_{1:2}$ proposed in this paper is equal to zero, which means two algorithms can solve problems with the same convergence speed.

TABLE 3. An example of contradictory experimental results.

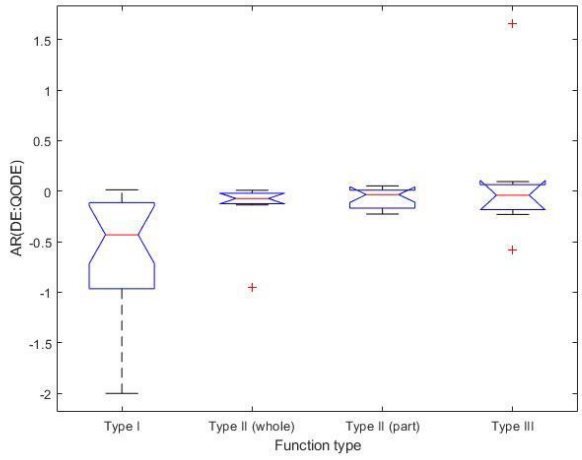
Function	Algorithm 1	Algorithm 2	$\mathrm{AR}_{1:2}$ in [13]	Proposed $\mathrm{AR}_{1:2}$
F1	10000	8000	1.25	-0.25
F2	80000	100000	0.8	0.25
Ave.			1.025	0

With the help of this new definition of AR, improvement degree in convergence speed can be discussed for ODE and QODE. The boxplots of $\mathrm{AR}_{\mathrm{DE:ODE}}$ and $\mathrm{AR}_{\mathrm{DE:QODE}}$ for all kinds of functions are presented below in Fig. 3.

According to classification principle mentioned above, the deviation degree of optimal solution to the center point increases in the order of Type I, Type II (whole), Type II (part) and Type III. The average $\mathrm{AR}_{\mathrm{DE:QODE}}$ of four function types are equal to −0.6073, −0.1630, −0.0636 and 0.1195, respectively. That is to say, convergence speed of QODE varies from 60% (progression) to −12% (regression). From Fig. 3(b), shape and position of four boxplots have also significant difference. These results both indicate that QODE produce biased performance in function type. If the function property is matched well with QOBL (such as very close distance between optimal solution and center point for Type I), QOBL is very helpful to improve algorithm performance. On the other hand, for solving Type III functions, QODE may run counter to conventional views of good algorithm performance.



(a)



(b)

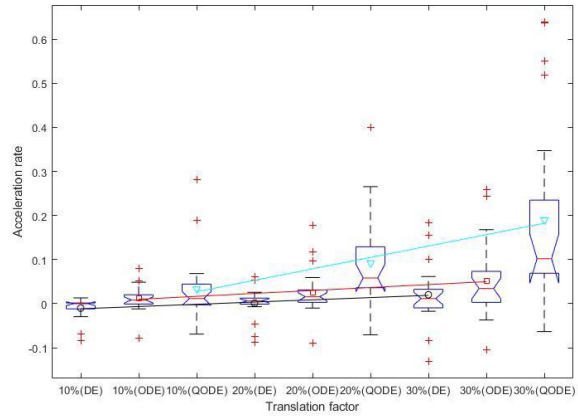
FIGURE 3. Acceleration rate between DE and ODE (QODE). (a) AR between DE and ODE. (b) AR between DE and QODE.

D. EXPERIMENT SERIES 2: SOLVING THE SAME FUNCTIONS WITH DIFFERENT OPTIMAL SOLUTIONS

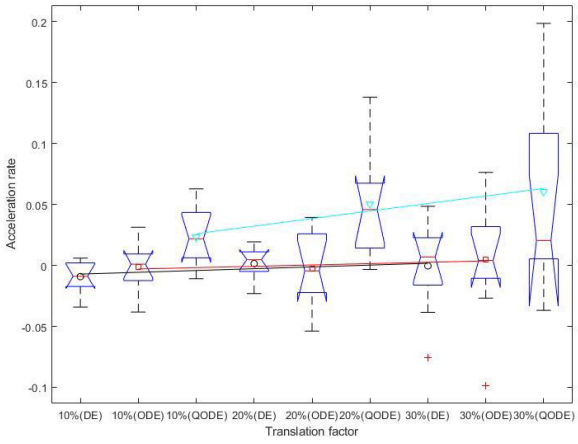
Next, the influence of the translation factor on algorithm convergence is studied to check our conjecture. On the whole, the number of best NFE obtained by QODE becomes smaller (from 38 to 31) and the number by DE becomes larger (from 4 to 10) when the translation factor increases from zero to 30%. From this, QODE may be very effective, especially to the functions, which their optimal solutions near the domain center.

Acceleration rates between each translation factor and zero are calculated based on Table 2 and (3). The resulted notched boxplots for Type I, Type II (whole) and Type II (part) functions are displayed below in Figs. 4(a), 4(b) and 4(c), respectively.

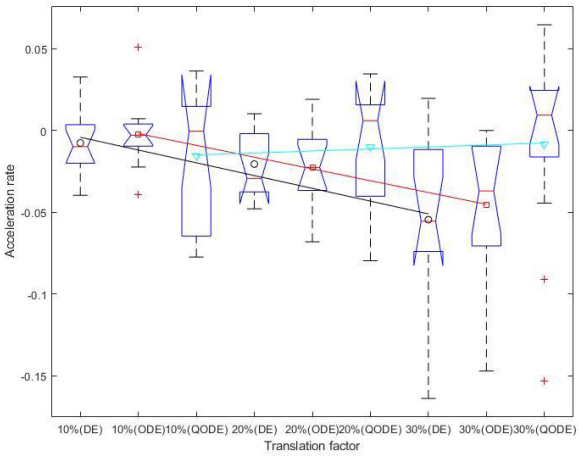
It can be clearly seen, having compared Figs. 4(a), 4(b) and 4(c), that acceleration rate for Type I functions is the largest among all function types. Note that the scale of ordinate is different among three subfigures. Perhaps the most extreme



(a)



(b)



(c)

FIGURE 4. Acceleration rate between different translation factors for three function types. (a) Type I functions. (b) Type II (whole) functions. (c) Type II (part) functions.

example is, when QODE solve F23 (Type I) with 30% translation factor, $AR_{0:30\%}$ is equal to 0.6390 ($\frac{14064-8581}{8581}$). That is to say, QODE needs more 63.90% times of function evaluations, only because the optimal solution of F23 shifts from 50%

(domain center) to 80% of function domain. From this point, QODE algorithm is very sensitive to the position of optimal solution. In fact, from its definition of (2), QOBL always apts to explore an area near the domain center. If the optimal solution is luckily situated within the area, QOBL will help DE to converge quickly. However, its effect is not evident for other positions of optimal solution.

In addition, the extreme value of AR are 0.1986 and −0.16369 for Type II (whole) and Type II (part) functions, respectively. The linear fitting results for DE, ODE and QODE are also presented in Fig. 4. It is obvious that, except for DE and ODE algorithms to solve Type II (part) functions, the slope of fitting line is larger than zero. This result means three algorithms need more NFEs to solve the optimization problems with the increasing translation factor. These fully illustrate function character (e.g. the position relationship between optimal solution and center point in this paper) has distinct impact on algorithm convergence.

On more careful observation, it is found that the slope of fitting line is near zero for DE and ODE. For QODE, it is much larger than zero, which can be treated as another evidence supporting the idea that ''smart'' matching between benchmark functions and QOBL is an important factor to the good performance of QODE.

The length of boxplot from lower tail to upper tail shows the spread of the data distribution. Comparatively speaking, in most instances, three algorithms can be ranked as follows, in descend order of data dispersion: QODE, ODE and DE. It indicates that DE can solve all kinds of optimization problems in a stable way. Thus we conclude that, the given function property has a very limited influence on DE and a great influence on QODE.

E. ADDITIONAL REMARKS

Two experiment evidences suggest that the good performance of QODE is closely interrelated to the position relationship between optimal solution and center point. Of course, we do not deny the value of QOBL to speed up the convergence.

On the other hand, in engineering application, it is reasonable for most cases to assume that the optimal solution of optimization problem is located near the domain center. If not, we can translate the relative position of optimal solution by changing the definition domain of optimization problem. In fact, during the evolution process, the range of candidate solution population is compressing and moving to global optimum, in which QOBL plays a bigger role. This process can be considered as another realization of changing the definition domain.

IV. EXPERIMENTAL ANALYSIS OF OPPOSITION-BASED LEARNING IN RESPECT OF ALGORITHM DESIGN

A. NEW DESIGN OF EMBEDDING METHOD

As shown in Fig. 1, ODE is developed easily by embedding OBL scheme in two steps of the classical DE. An interesting fact about algorithm structure is that OBL scheme is utilized on the whole population, not on the special individuals. More specifically, all opposite solutions generated by OBL definition constitutes an opposition-based population (OP_G) firstly. Then the same population size of fittest individuals are selected from the joint population (P_G and OP_G) and remained to the child generation. It is named population-based embedding method in this paper. When using this method, it is possible to pitch on (or give up) both the candidate solution and its corresponding opposite solution simultaneously.

Furthermore this population-based embedding method of OBL is completely different to opposition-based optimization defined in Section II.A. From the definition, its original intention is to compare the solution and its corresponding opposite solution and then select the fitter one. By this means, only one of two points can in any case be survived to the child generation and another will be discarded.

To follow along this idea, a new approach named individual-based embedding method is proposed and the corresponding opposition-based DE algorithm is developed in this paper. This algorithm is called as ODE(Ind) to distinguish it with previous ODE(Pop) in [13]. The pseudo code of ODE(Ind) is presented in Fig. 5. In ODE(Ind), two key procedures (opposition-based population initialization and opposition-based generation jumping) convey the idea of opposition-based optimization entirely by new individual-based embedding method.

In Section IV.B, two opposition-based DE algorithms are compared firstly. Then we propose a more plausible explanation of the performance differences between them, which can reclaim the previous misunderstanding of OBL. Furthermore, the effects of function type and jumping rate are discussed in Sections IV.C and IV.D, respectively.

B. HOLISTIC COMPARISON OF TWO EMBEDDING METHODS

In this subsection, two embedding methods are compared firstly in terms of convergence speed and algorithm robustness. Experimental results (NFE and SR) of DE, ODE(Pop) and ODE(Ind) solving 58 optimization problems are given in Table 4. To make the table concise, the self-explanatory SR (one or zero) is also omitted here. In addition, acceleration rates between two opposition-based algorithms using different embedding methods are calculated. Then the average acceleration rates on each function type are recorded at the end of the corresponding function type in Table 4, and total average on four function types are also shown in the last row of the table.

Comparatively speaking, DE algorithm has relatively high convergence speed on 28 problems, and ODE(Pop) has on other 25 problems. Based on the total average $AR_{DE:ODE(Pop)}$, the convergence speed of ODE(Pop) increases 5.26% than DE algorithm. Unfortunately, the improvement degree of performance is significantly lower than expected. It was reported in [13] that ODE is on average 44% faster than DE. The reason behind this phenomenon is that the performance of

Step 1 Opposition-based Population Initialization
Set generation number $G = 0$, and randomly initialize a population $P_0 = \{X_{1,0}, X_{2,0}, \cdots, X_{Np,0}\}$
FOR $i = 1$ to N_p
  $OP_{i,0} = a+b-X_{i,0}$
  Evaluation the opposite solution $OP_{i,0}$
  IF $f(OP_{i,0}) < f(X_{i,0})$
    $X_{i,0} = OP_{i,0}$
  END IF
END FOR
Step 2 WHILE stopping criterion is not satisfied DO
  Step 2.1 Mutation
  /* Generate a mutated vector $V_{i,G}$ for each target vector $X_{i,G}$ */
  FOR $i = 1$ to N_p
    Select three parents $X_{i1,G}$, $X_{i2,G}$, and $X_{i3,G}$ randomly from current population where $i \neq i_1 \neq i_2 \neq i_3$
    $V_{i,G} = X_{i1,G} + F \times (X_{i2,G} - X_{i3,G})$
  END FOR
  Step 2.2 Crossover
  /* Generate a trial vector $U_{i,G}$ for each target vector $X_{i,G}$ */
  FOR $i = 1$ to N_p
    $j_{rand} = rand(0,1) \times D$
    FOR $j = 1$ to D
      IF ($rand_j(0,1) < C_R$) or ($j = j_{rand}$)
        $U_{i,j,G} = V_{i,j,G}$
      ELSE
        $U_{i,j,G} = X_{i,j,G}$
      END IF
    END FOR
  END FOR
  Step 2.3 Selection
  FOR $i = 1$ to N_p
    Evaluation the trial vector $U_{i,G}$
    IF $f(U_{i,G}) < f(X_{i,G})$
      $X_{i,G+1} = U_{i,G}$
    ELSE
      $X_{i,G+1} = X_{i,G}$
    END IF
  END FOR
  Step 2.4 Opposition-based Generation Jumping
  IF $rand(0, 1) < J_r$
    FOR $i = 1$ to N_p
      $OP_{i,G} = MIN_G+MAX_G-X_{i,G}$
      Evaluation the opposite solution $OP_{i,G}$
      IF $f(OP_{i,G}) < f(X_{i,G})$
        $X_{i,G} = OP_{i,G}$
      END IF
    END FOR
  END IF
  $G = G + 1$
END WHILE

FIGURE 5. Pseudo code for opposition-based differential evolution (Individual) (ODE(Ind)).

ODE(Pop) vary for different jumping rates. It is set as 0.05 in this paper as mentioned in Section II.B, while it is equal to 0.3 in [13]. In Section IV.C, the effect of jumping rate will be more fully discussed.

Obviously, the success rates are almost the same for ODE(Pop) and ODE(Ind) from Table 4. However, ODE(Ind) has the slowest convergence speed of all algorithms assessed here. In general, it needs more 4.63% NFEs than DE algorithm. For 46 problems among all tested problems, NFE required by ODE(Ind) is larger than that by ODE(Pop).

TABLE 4. Experimental comparison of DE, ODE(Pop) and ODE(Ind).

Function	Type	DE	ODE(Pop)		ODE(Ind)		$AR_{Pop:Ind}$
			NFE(SR)	UR	NFE(SR)	UR	
F1	I	73502	**66648**	34.95	75690	17.09	0.1357
F2	I	81417	**73248**	35.19	83855	17.21	0.1448
F3	I	**153992**	155951	8.01	162006	3.19	0.0388
F5	I	326824 (0.94)	**120489** (0.93)	13.99	357338 (0.97)	2.63	1.9657
F6	I	77280	**69962**	35.03	79897	17.37	0.1420
F7	I	19159	**14277**	46.40	19183	33.80	0.3436
F8	I	143902	**131290**	32.86	149104	16.28	0.1357
F16	I	**2686**	2717	45.27	2875	12.30	0.0582
F21	I	**155837**	156449	15.46	164222	4.22	0.0497
F22	I	-	**173843** (0.79)	45.58	307593	38.21	0.7694
F23	I	19712	**18128**	29.81	20709	11.60	0.1424
F24	I	-	-	-	-	-	-
F30	I	41952	**32694**	45.61	41920	32.54	0.2822
F31	I	345115	**326449**	11.68	488588	3.43	0.4967
F32	I	4952	**4788**	42.69	5101	15.74	0.0654
F33	I	25696	**23761**	23.59	25512	11.37	0.0737
F34	I	31255	**27523**	24.09	31302	11.96	0.1373
F37	I	**4129**	4185	45.48	4316	18.19	0.0313
F38	I	**4088**	4172	44.46	4306	18.20	0.0321
F39	I	**2909**	2935	46.02	3074	15.00	0.0474
F41	I	16042	**15832**	42.78	16438	25.67	0.0383
F53	I	6707	**6004**	43.94	6931	19.13	0.1544
F54	I	7407	**7362**	38.84	7625	18.03	0.0357
F56	I	-	-	-	-	-	
F57	I	**15472**	15579	37.63	16061	23.80	0.0309
Ave.							0.2327
F10	II(whole)	**13731**	14203	21.18	14437	15.39	0.0165
F11	II(whole)	**5348**	5356	20.09	5532	13.12	0.0329
F14	II(whole)	**4866**	4973	42.06	4933	17.19	-0.0081
F15	II(whole)	81126 (0.94)	**74185**	32.70	84331	16.53	0.1368
F25	II(whole)	**9753**	9997	22.32	10156	17.34	0.0159
F35	II(whole)	**3032**	3082	45.24	3082	41.70	0
F44	II(whole)	**5501**	5539	45.16	5625	39.08	0.0155
F46	II(whole)	**4781**	4905	43.49	5021	36.39	0.0236
F49	II(whole)	**4430** (0.91)	4633 (0.95)	31.23	4671 (0.93)	29.72	0.0082
F58	II(whole)	14092	**13789**	22.63	14481	16.12	0.0502
Ave.							0.0291
F9	II(part)	**3209**	3398	37.10	3317	34.68	-0.0244
F12	II(part)	**4113**	4178	41.95	4217	37.18	0.0093
F13	II(part)	**11237** (0.95)	11436 (0.89)	34.67	11298 (0.95)	28.47	-0.0122
F18	II(part)	184888 (0.5)	**187213** (0.54)	4.13	192396 (0.5)	2.81	0.0277
F26	II(part)	**9456**	9720	37.10	9994	30.76	0.0282
F27	II(part)	**8932**	9069	37.73	9235	30.02	0.0183
F28	II(part)	9026	**9013**	38.64	9268	31.79	0.0283

F29	II(part)	8315 (0.98)	**8608**	31.66	8691 (0.98)	26.31	0.0096
F40	II(part)	**7190**	7437	40.92	7472	19.52	0.0047
F42	II(part)	**3567**	3630	41.85	3659	37.34	0.0080
F43	II(part)	**5517**	5716	25.49	5629	23.23	-0.0155
F48	II(part)	**2496**	2525	39.69	2574	36.91	0.0194
F50	II(part)	240239	**236650**	5.29	245414	4.85	0.0370
F51	II(part)	-	-	-	-	-	-
F55	II(part)	-	-	-	-	-	-
Ave.							0.0107
F4	III	**356748**	357083	31.86	434762	23.20	0.2175
F17	III	-	-	-	-	-	-
F19	III	362274	**358601**	4.07	376365	2.28	0.0495
F20	III	**6250**	6844	27.90	6742	22.48	-0.0151
F36	III	**7489**	7736	38.81	8299	17.51	0.0728
F45	III	**2558**	2659	44.42	2586	40.81	-0.0282
F47	III	2706	**2672**	44.73	2698	43.06	0.0097
F52	III	21962	**21929**	32.38	22521	23.94	0.0270
Ave.							0.0476
Type I		7	16		0		
Type II (whole)		8	2		0		
Type II (part)		9	4		0		
Type III		4	3		0		
Total		28	25		0		0.0800

The average $AR_{Pop:Ind}$ is larger than zero for all function types and the total average $AR_{Pop:Ind}$ is equal to 8.00%. These results confirm that ODE(Pop) has a higher convergence speed comparable with ODE(Ind), owing to the proposed embedding method.

In these experiments, almost all parameters are the same, except the embedding method chosen in OBL scheme. In order to explain the performance difference between two opposition-based algorithms, we first try to look for sound reason from the contribution of opposite points.

The goal of embedding opposition-based learning is to achieve a better approximation of true solution and to accelerate convergence of population-based algorithm. We may take it for granted that algorithm performance will be better and better the more opposite points survived to the child population. It is clear from Table 4 that, except for very few cases, most cases confirm this hypothesis. For example, when solving function F1, NFE required by ODE(Pop) decreases by 12%, which means good convergence, and at the same time, the average UR increases by 105% compared with ODE(Ind).

However this hypothesis cannot explain why the performance of ODE(Ind) deteriorates sharply despite of embedding some opposite points. For example, ODE(Ind) needs more 2188 function evaluations than the classical DE, while 17.09% opposite points are utilized on average. We believe that the true reason for good outcomes of opposition-based algorithm is just the embedding method, rather than the utilization rate of opposite points. In order to show their differences, average UR of two opposition-based algorithms are presented for four typical functions in Fig. 6. Note that the number of times that opposition-based learning is embedded in 100 experiments becomes less and less. Then the average AR may fluctuate violently in the last part of the curve.

Here, we will discuss the simplest case first, namely opposition-based population initialization. For a good candidate solution $X_{k,0}$ $(k = 1, 2, \ldots, N)$ in initial population P_0, it is very possible that the corresponding opposite point $OX_{k,0}$ also has a similar fitness value and good performance in opposition-based population OP_0. For population-based embedding method, the pairing solutions with good and similar quality, $X_{k,0}$ and $OX_{k,0}$, are both remained to the child generation. At this time, other solution with bad performance (and the corresponding opposite point) will be discarded inevitably. The uniform distribution is usually assumed, without prior knowledge of optimization problem. Thus a natural inference is that half of initial opposition population can remained to child generation, and the value of UR is near 50% as shown in Fig. 6. As a result, this population-based embedding method is biased to high quality solutions and strongly reduces population diversity.

While for new individual-based embedding method, the pairing solutions, $X_{k,0}$ and $OX_{k,0}$, are compared directly. Yet, no matter how good the other is, only one of them can be survived to the child generation. At this time, some solutions may be replaced with better opposite points, and population diversity has not be affected markedly by opposition-based learning. In addition, if the benchmark function is symmetrical about the center of the interval of its definition, the function fitness of the paring solutions are the same, and the opposite point cannot be superior than the original solution, which leads to UR = 0% as shown in Fig. 6(a). Contrary to the symmetrical case, their function fitness are different with each other and there is half chance to remain in child generation as illustrated in Figs 6(b), 6(c) and 6(d).

Next opposition-based generation jumping will be studied. The fitness of solutions, together with the evolution of population, decrease during the course of minimization. Compared with solution after mutation, crossover and selection operations, the performance of their opposite point often become worse remarkably. For two embedding methods of OBL, in opposition-based generation jumping, the chance to be chosen of opposite points is much less than 50% as shown in Fig. 6.

On conclusion, opposition-based learning can be considered as a mutation operation to some extent. The noticeable difference between OBL and conventional mutation operation is that the distance and direction of mutation vector. A solution can hop to opposite half-plane from the definition of OBL, while it may move astatically to a slightly different point for mutation operation. In this way, we can explain why ODE(Pop) has high convergence speed as follows. For population-based embedding method, as mentioned above, it may reduce population diversity like mutation operation. On the other hand, it can also maintain more solutions with

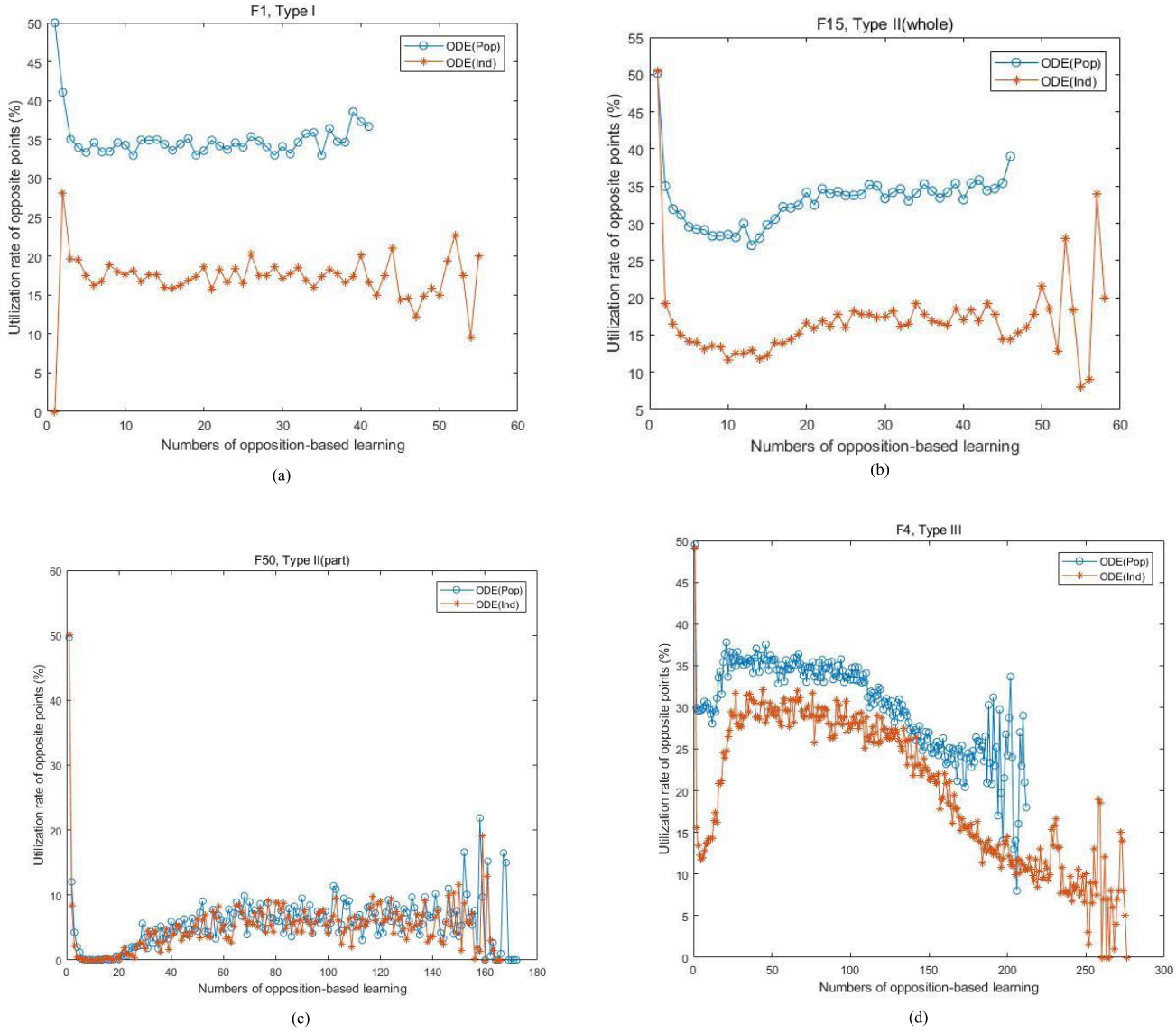


FIGURE 6. Average UR of two opposition-based algorithms for four typical functions. (a) Type I function. (b) Type II (whole) function. (c) Type II (part) function. (d) Type III function.

good performance, which leads to a high convergence speed of ODE(Pop).

In fact, the case is just the opposite for new individual-based embedding method. The candidate sulutions are updated one by one with the corresponding opposite solutions with good quality. Thus ODE(Ind) has high population diversity during the whole iteration process. In fairness to DE, extra fitness evaluation required for the opposite points both in population initialization and generation jumping phases are counted in all experiments. As a result, more NFEs are needed to solve optimization problems, although some opposite points are generated and evolved for ODE(Ind).

To sum up, the performance differences between two opposition-based algorithms cannot be explained by the utilization rate of opposite points. A possible explanation is, in our opinions, the embedding method of OBL scheme.

C. EFFECT FOR VARIOUS FUNCTION TYPES

In order to investigate the effect for different function types, the boxplots of $AR_{ODE(Pop):ODE(Ind)}$ for four types are displayed below in Fig. 7.

From Fig. 7, it can be seen that the embedding method chosen in OBL scheme has more affection for Type I functions than other types. For example, ODE(Pop) can solve function F5 (Type I) in 120489 function evaluations, while ODE(Ind) needs 357338 function evaluations on average. There is a distinct difference between them as shown in Fig. 7.

The optimal solution is located at the domain center for Type I functions. Therefore, it can be reasonably assumed that their fitness distribution is symmetrical about the center. By the definition of population-based embedding method, relatively more opposite points with good quality can be included in child population to speed up convergence. As a

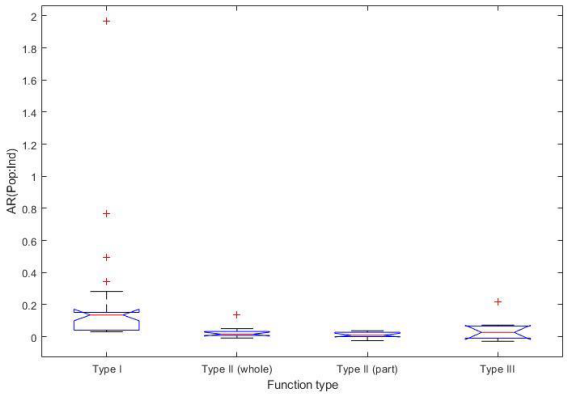


FIGURE 7. $AR_{ODE(Pop):ODE(Ind)}$ for four function types.

result, seen from the last five rows in Table 5, ODE(Pop) has the highest convergence speed on Type I functions among all types. Hence the difference between ODE(Pop) and ODE(Ind) on Type I functions looks big compared to other function types.

D. PROPER SETTING OF JUMPRING RATE

The jumping rate is an important control parameter which, if optimally set, can achieve even better results. The value of jumping rate for our current study is borrowed from [4] and is set to 0.05 without any effort to find an optimal value. In this subsection, we will discuss the detailed effect on ODE(Ind) and try to find an optimal jumping rate from a discrete set of jumping rate values.

Now, we repeat the conducted experiments for $J_r \in [0.05, 0.3]$ with step size of 0.05. The results (NFE) for different jumping rates are presented in Table 5. The total average AR between two ODE(Ind) algorithms with different jumping rates are summarized in the last row of the table to ease the comparison.

It is clearly that, except for very few cases, NFE value becomes larger and larger with the increasing jumping rate. The total average AR increased from 0 to 0.150, and the increasing trend is approximately linear with the value of J_r. For those few cases, they can simply be considered as random error in 100 independent experiments.

By contrast, for ODE(Pop) in [13], the general recommendation of jumping rate are 0.3 or 0.6, and the average optimal value is equal to 0.37 for the same benchmark function set. As mentioned above, opposition-based learning in ODE(Ind) can be considered as a mutation operation. Thus the optimal jumping rate (e.g. mutation probability) must be very small as shown by the experimental results in this paper.

E. ADDITIONAL REMARKS

To follow along the definition of opposition-based optimization, a new individual-based embedding method is proposed in this section. The bad convergence performance of ODE(Ind) is precisely because of new embedding method. However its performance can be further improved by proper setting of control parameters, such as jumping rate.

TABLE 5. Experimental comparison of ode(ind) for different jumping rates.

Function	Type	Jumping rate J_r					
		0.05	0.1	0.15	0.2	0.25	0.3
F1	I	75690	78401	80957	83817	86557	88934
F2	I	83855	86749	89523	92513	95663	98175
F3	I	162006	168325	177433	182946	191707	198378
F5	I	357338	377367	372220	414736	450952	466129
F6	I	79897	82519	85278	87903	91071	93634
F7	I	19183	19482	19982	20331	20328	20548
F8	I	149104	154654	160532	165716	171243	177059
F16	I	2875	2932	2975	3097	3032	3147
F21	I	164222	171457	179786	188570	196401	206032
F22	I	307593	331143	355709	374976	394423	415722
F23	I	20709	21788	22383	23437	24392	25590
F24	I	-	-	-	-	-	-
F30	I	41920	42778	44612	45513	46738	47316
F31	I	488588	569063	607784	620755	666918	696833
F32	I	5101	5315	5378	5482	5593	5648
F33	I	25512	25769	25985	25395	25732	26265
F34	I	31302	32812	35846	35820	37475	38602
F37	I	4316	4420	4526	4508	4617	4723
F38	I	4306	4416	4477	4609	4668	4732
F39	I	3074	3100	3169	3241	3234	3352
F41	I	16438	16819	16989	17381	17646	17890
F53	I	6931	7024	7007	7202	7196	7501
F54	I	7625	7804	7862	8038	8164	8273
F56	I	-	-	-	-	-	-
F57	I	16061	16454	16736	17107	17476	17931
F10	II(whole)	14437	14947	15682	15833	16548	16689
F11	II(whole)	5532	5625	5822	5878	6054	6169
F14	II(whole)	4933	5303	5372	5400	5422	5569
F15	II(whole)	84331	87880	91010	94389	97853	101158
F25	II(whole)	10156	10462	10665	10966	11422	11836
F35	II(whole)	3082	3176	3233	3310	3339	3375
F44	II(whole)	5625	5754	5748	5950	5981	6073
F46	II(whole)	5021	5120	5265	5237	5447	5475
F49	II(whole)	4671	4876	4998	5157	5350	5461
F58	II(whole)	14481	14755	15196	15732	16351	16550
F9	II(part)	3317	3456	3559	3639	3799	3808
F12	II(part)	4217	4305	4356	4474	4500	4639
F13	II(part)	11298	11814	12217	12527	12910	12983
F18	II(part)	192396	197972	202006	220343	221559	238618
F26	II(part)	9994	10186	10465	10502	10837	10941
F27	II(part)	9235	9502	9751	9870	10100	10386
F28	II(part)	9268	9261	9561	9886	10139	10288
F29	II(part)	8691	8946	9296	9616	9761	9947
F40	II(part)	7472	7821	8150	8220	8345	8443
F42	II(part)	3659	3802	3878	3924	4040	4128

F43	II(part)	5629	5872	6070	6333	6506	6649
F48	II(part)	2574	2612	2619	2816	2841	2818
F50	II(part)	245414	249559	262431	269526	266857	295846
F51	II(part)	-	-	-	-	-	-
F55	II(part)	-	-	-	-	-	-
F4	III	434762	629643	903630	-	-	-
F17	III	-	-	-	-	-	-
F19	III	376365	391213	409518	425941	444864	462891
F20	III	6742	6952	7477	7906	8345	8470
F36	III	8299	8260	8322	8131	8103	7936
F45	III	2586	2638	2783	2787	2865	2855
F47	III	2698	2755	2796	2905	2889	2962
F52	III	22521	23069	23657	24165	24898	25278
Total ave. AR		0	0.0329	0.0636	0.0928	0.122	0.150

In addition, the classical population-based embedding method is also a good choice. In a very ordinary, but ingenious way, it speed up the convergence at the expense of population diversity. To an extent, its way of action is similar to that of mutation operation.

Last but not least important, we give an explanation why the experimental results are inconsistent with the theoretical results. Until now, various mathematical theorems have been proved to indicate the usefulness of using the OBL concept [4], [11]–[16]. To sum it all up, it is effective than using the random opposite points, which is proved by many experiments. Unfortunately this effectiveness needs extra fitness evaluation required for the opposite points. If the performance improvement obtained by OBL is insufficient compared to population evolution of DE algorithm, these extra costs inevitably degrade performance of opposition-based algorithm.

V. CONCLUSION

The goal of this paper is exploring the possible reasons behind the good performance and disambiguate some misunderstanding of OBL. In this paper, we reexamined opposition-based learning in respects of function prototype and algorithm design.

If benchmark functions are chosen optimally, intelligent algorithm can achieve even better results. According to the position relationship between optimal solution and center point, an important function prototype, some well-known benchmark functions are divided into three categories: Type I, Type II and Type III. With the help of acceleration rate, we firmly believe that QODE produce biased results in respect of function type and translation factor. These numerical results indicate that ''smart'' matching between benchmark functions and QOBL is an important factor to the good performance of QODE.

In the second part of this paper, a novel individual-based embedding method is proposed to coincide with the definition of opposition-based optimization. Then two opposition-based DE algorithms are compared to discuss two embedding methods in OBL scheme. The results confirm that the convergence difference stem from the embedding method, rather than the utilization rate of opposite points, and opposition-based learning can be considered as a mutation operation to some extent. At last, the effects for different function types and jumping rates are discussed in this paper.

There are several unexplored aspects that need future research attention. For instance, a great potential work is to analyze the extent and causes of negative transfer with utilizing OBL scheme. In the future, we will also develop adaptive or self-adaptive OBL strategies that able to control the jumping rate of opposition during optimization.

ACKNOWLEDGMENT

Qingzheng Xu would like to thank Prof. Y. S. Ong for the valuable comments on the manuscript.

REFERENCES

[1] H. R. Tizhoosh, ''Opposition-based learning: A new scheme for machine intelligence,'' in *Proc. Int. Conf. Comput. Intell. Modelling, Control Automat., Int. Conf. Intell. Agents, Web Technol. Internet Commerce*, Vienna, Austria, Nov. 2005, pp. 695–701.

[2] Q. Xu, L. Wang, N. Wang, X. Hei, and L. Zhao, ''A review of opposition-based learning from 2005 to 2012,'' *Eng. Appl. Artif. Intell.*, vol. 29, pp. 1–12, Mar. 2014.

[3] S. Mahdavi, S. Rahnamayan, and K. Deb, ''Opposition based learning: A literature review,'' *Swarm Evol. Comput.*, vol. 39, no. 1, pp. 1–23, Apr. 2018.

[4] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, ''Quasi-oppositional differential evolution,'' in *Proc. IEEE Congr. Evol. Comput.*, Singapore, Sep. 2007, pp. 2229–2236.

[5] M. Ergezer, D. Simon, and D. Du, ''Oppositional biogeography-based optimization,'' in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, San Antonio, TX, USA, Oct. 2009, pp. 1009–1014.

[6] S. Rahnamayan and G. G. Wang, ''Center-based sampling for population-based algorithms,'' in *Proc. IEEE Congr. Evol. Comput.*, Trondheim, Norway, May 2009, pp. 933–938.

[7] H. Wang, Z. J. Wu, Y. Liu, J. Wang, D. Z. Jiang, and L. L. Chen, ''Space transformation search: A new evolutionary technique,'' in *Proc. ACM/SIGEVO Summit Genet. Evol. Comput.*, Shanghai, China, 2009, pp. 537–544.

[8] Q. Xu, L. Wang, B. He, and N. Wang, ''Modified opposition-based differential evolution for function optimization,'' *J. Comput. Inf. Syst.*, vol. 7, no. 5, pp. 1582–1591, 2011.

[9] 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 *Proc. IEEE Congr. Evol. Comput.*, Beijing, China, Jul. 2014, pp. 2259–2265.

[10] H. Liu, Z. Wu, H. Li, H. Wang, S. Rahnamayan, and C. Deng, ''Rotation-based learning: A novel extension of opposition-based learning,'' in *Proc. Pacific Rim Int. Conf. Artif. Intell.*, Gold Coast, QLD, Australia, 2014, pp. 511–522.

[11] H. Xu, C. D. Erdbrink, and V. V. Krzhizhanovskaya, ''How to speed up optimization? Opposite-center learning and its application to differential evolution,'' *Procedia Comput. Sci.*, vol. 51, no. 1, pp. 805–814, 2015.

[12] Z. Seif and M. B. Ahmadi, ''Opposition versus randomness in binary spaces,'' *Appl. Soft Comput.*, vol. 27, no. 1, pp. 28–37, 2015.

[13] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, ''Opposition-based differential evolution,'' *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 64–79, Feb. 2008.

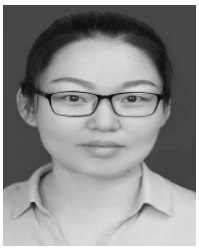
[14] Q. Xu, H. Xu, and W. Wu, ''Theoretical analysis and experimental evidence of opposite-center learning,'' *IEEE Access*, vol. 6, pp. 34948–34966, 2018.

[15] M. Ergezer and D. Simon, ''Mathematical and experimental analyses of oppositional algorithms,'' *IEEE Trans. Cybern.*, vol. 44, no. 11, pp. 2178–2189, Nov. 2014.

[16] Z. Seif and M. B. Ahmadi, ''An opposition-based algorithm for function optimization,'' *Eng. Appl. Artif. Intell.*, vol. 37, no. 1, pp. 293–306, 2015.

QINGZHENG XU received the B.S. degree in information engineering from the PLA University of Science and Technology, Nanjing, China, in 2002, and the Ph.D. degree in control theory and engineering from the Xi'an University of Technology, Xi'an, China, in 2011.

From 2002 to 2016, he was a Lecturer with the Xi'an Communications Institute, Xi'an. Since 2017, he has been a Lecturer with the College of Information and Communication, National University of Defense Technology, Xi'an. He is currently a Visiting Scholar with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His main research interests include opposition-based learning, nature-inspired computation, and combinatorial optimization.



NA WANG received the B.S. degree in mathematics and applied mathematics and the M.S. degree in applied mathematics from Shaanxi Normal University, Xi'an, China, in 2004 and 2007, respectively.

From 2007 to 2016, she was a Lecturer with the Xi'an Communications Institute, Xi'an. Since 2017, she has been an Associate Professor with the College of Information and Communication, National University of Defense Technology, Xi'an. Her main research interests include optimization theory and machine learning.

FENG ZOU received the M.S. degree from Wuyi University, Jiangmen, China, in 2007, and the Ph.D. degree from the Xi'an University of Technology, Xi'an, China, in 2015.

From 2017 to 2018, he was a Visiting Scholar with the College of Electronics and Information Engineering, Tongji University, Shanghai, China. He is currently an Associate Professor with Huaibei Normal University, Huaibei, China. His research interests mainly include evolutionary computing, swarm intelligence optimization, and biological information processing.

JUNGANG YANG received the B.S. degree in fiber optic communication engineering from the PLA University of Science and Technology, Nanjing, China, in 1996, and the M.S. and Ph.D. degrees in information and communication engineering from Xidian University, Xi'an, China, in 2003 and 2008, respectively.

He is currently a Professor with the College of Information and Communication, National University of Defense Technology, Xi'an. His current research interests include fiber optic communication, information service, and intelligent computation.