Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Oppositional Biogeography-Based Optimization

Mehmet Ergezer, Dan Simon, and Dawei Du
Cleveland State University
Department of Electrical and Computer Engineering
Cleveland, Ohio, USA
m.ergezer@csuohio.edu

***Abstract*—We propose a novel variation to biogeography-based optimization (BBO), which is an evolutionary algorithm (EA) developed for global optimization. The new algorithm employs opposition-based learning (OBL) alongside BBO's migration rates to create oppositional BBO (OBƁO). Additionally, a new opposition method named quasi-reflection is introduced. Quasi-reflection is based on opposite numbers theory and we mathematically prove that it has the highest expected probability of being closer to the problem solution among all OBL methods. The oppositional algorithm is further revised by the addition of dynamic domain scaling and weighted reflection. Simulations have been performed to validate the performance of quasi-opposition as well as a mathematical analysis for a single-dimensional problem. Empirical results demonstrate that with the assistance of quasi-reflection, OBƁO significantly outperforms BBO in terms of success rate and the number of fitness function evaluations required to find an optimal solution.**

***Index Terms*—Biogeography-based optimization (BBO), evolutionary algorithms, opposition-based learning, opposite numbers, quasi-opposite numbers, quasi-reflected numbers, probability.**

I. Introduction

Evolutionary Algorithms (EA) are created to solve implicit and explicit functions and, unlike many numerical methods, do not require the objective function to be differentiable. In fact, EAs do not require any information about the objective function. This characteristic makes them a desirable method for solving complex problems.

Biogeography-based optimization (BBO) [1] is a new EA developed for global optimization. BBO is a generalization of biogeography to EA. It is modeled after the immigration and emigration of species between islands in search of more friendly habitats. The islands represent the solutions and they are ranked by their island suitability index (ISI), where a higher ISI signifies a superior fitness value. The islands are comprised of solution features named suitability index variables (SIV), equivalent to GA's genes.

BBO is one of the newest evolutionary algorithms, but it has already proven itself a worthy competitor to its better-known siblings. The Markov analysis in [2], [3] proves that BBO outperforms GA on simple unimodal, multimodal and deceptive benchmark functions when used with low mutation rates. Reference [4] provides experimental studies comparing BBO's performance to many other EA's on a wide set of benchmarks.

This work was supported by NSF Grant 0826124 in the CMMI Division of the Engineering Directorate.

However, there is still some room left for improving BBO since many other techniques have been developed to enhance other EAs. In this paper, we adapt one such algorithm, opposition-based learning (OBL), to BBO in an attempt to attain BBO's highest potential.

OBL is utilized by EAs to accelerate the convergence speed by comparing the fitness of a solution estimate to its opposite and keeping the fitter one in the population. Opposition-based differential evolution (ODE) [5] was the first implementation of OBL to EAs. Later, [6] demonstrated that quasi-opposite points have a better convergence rate than opposite points. In Section III, we provide the analytical expressions illustrating the benefits of selecting quasi-opposite points in a single-dimensional case. Later, we introduce quasi-reflected points and demonstrate that quasi-reflection yields the highest probability of success while requiring less fitness computations than other OBL algorithms. Additionally, we present the effects of increased dimensions on opposite points. We add quasi-reflection to BBO to create oppositional BBO (OBƁO). In OBƁO , the OBL algorithm is further modified by dynamic domain scaling and weighting the reflection amount based on individual's fitness.

The paper is organized as follows: Section II provides a brief introduction to BBO. Section III defines the opposite points, derives the expected probabilities of quasi-opposite points' distance to the solution for a single-dimensional case and presents the effects of higher dimensions for all OBL methods. Section IV describes the proposed OBƁO algorithm. Section V introduces the benchmark problems, the experimental settings and discusses the results obtained. Finally, Section VI states the conclusion and suggests future work.

II. Biogeography-Based Optimization

One of the distinguishing features of BBO is that when updating the population, BBO considers the fitness of the immigrating *and* emigrating islands via the immigration and emigration curves. Fig. 1 illustrates linear BBO immigration and emigration curves. The worst solution has the highest immigration rate; hence, it has a very high chance of borrowing features from other solutions, helping it to improve for the next generation. The best solution has a very low immigration rate, indicating that it is less likely to be altered by the other solutions. The emigration rate works similarly. Note that emigration in BBO does not mean that the emigrating island loses a feature. For example, if a feature in island 1 migrates

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

to island 2, then both islands 1 and 2 have this feature. The worst solution is assumed to have the worst features; thus, it has a very low emigration rate and a low chance of sharing its features. On the other hand, the solution that has the best features also has the highest probability of sharing them. For this research, we implement partial immigration-based BBO as described in [4].

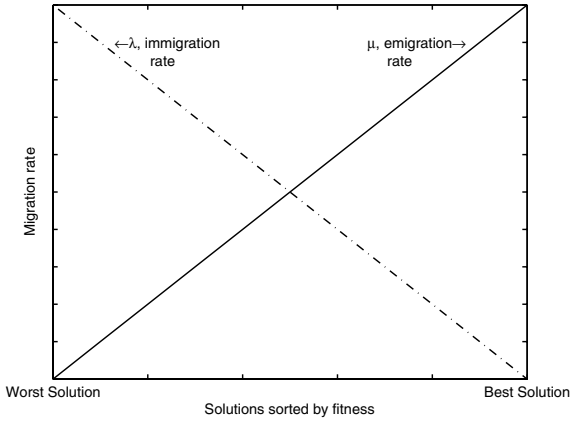

Figure 1. Linear migration rates plotted against the sorted population.

III. Opposition-based learning

Opposition-based learning (OBL) is proposed in [7]. OBL has first been utilized to improve learning and back propagation in neural networks [8], and since then it has been applied to many evolutionary algorithms such as differential evolution [5], particle swarm optimization [9] and ant colony optimization [10].

The philosophy of OBL is that natural learning is time consuming because it is modeled after a very slow process. For example, GA is modeled after the evolution of species and it takes many life cycles for a species to evolve. On the other hand, human society progresses at a much faster rate via "social revolutions" [7]. Hence, if such a model could be simulated, the learning process could be improved. Revolutions are fast and changes are fundamental, whether in politics, economics or any other context. OBL maps this theory to machine learning and proposes to use opposite numbers instead of random ones to quickly evolve the population.

The main principal of OBL is to utilize opposite numbers to approach the solution. The inventors of OBL claim that a number's opposite is probably closer than a random number to a solution. Thus, by comparing a number to its opposite, a smaller search space is needed to converge to the right solution. Later in this section, we prove that a quasi-opposite number is usually closer than a random number to the solution. We also prove that a quasi-opposite number is usually closer than an opposite number to the solution.

A. Quasi-opposition and Opposition Points

In [6], Rahnamayan introduces quasi-opposition-based learning and proves that a quasi-opposite point is more likely to be closer to the solution than the opposite point. In this section, we prove how much quasi-opposition is better than opposition. First, let us define opposite and quasi-opposite numbers in one dimensional space. These definitions can easily be extended to higher dimensions.

Definition 1: Let x be any real number between $[a, b]$. Its opposite, x_o, is defined as

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

Definition 2: Let x be any real number between $[a, b]$. Its quasi-opposite point, x_{qo}, is defined as

$$x_{qo} = \text{rand}(c, \hat{x}_o) \tag{2}$$

where c is the center of the interval $[a, b]$ and can be calculated as $(a + b)/2$ and rand$(c, \hat{x}_o)$ is a random number uniformly distributed between c and $\hat{x}_o$.

Since we reflect x to x_o to accelerate the exploration process, we propose apply the same logic and reflect the quasi-opposite point, x_{qo}, to obtain the quasi-reflected point, x_{qr}.

Definition 3: Let x be any real number between $[a, b]$. Then the quasi-reflected point, x_{qr}, is defined as

$$x_{qr} = \text{rand}(c, x) \tag{3}$$

where rand(c, x) is a random number uniformly distributed between c and x.

Fig. 2 illustrates a point $\hat{x}$, its opposition, $\hat{x}_o$, its quasi-opposition, $\hat{x}_{qo}$ and its quasi-reflection, $\hat{x}_{qr}$.

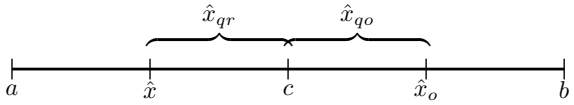

Figure 2. Opposite points defined in domain $[a, b]$. c is the center of the domain and $\hat{x}$ is an estimated solution, generated by an EA. $\hat{x}_o$ is the opposite of $\hat{x}$, and $\hat{x}_{qo}$ and $\hat{x}_{qr}$ are the quasi-opposite and quasi-reflected points, respectively.

B. Probabilistic Performance of Quasi-opposition

If x is the solution to the problem and $\hat{x}$ is our estimated solution provided by the EA, then $\hat{x}_o$ is the opposite of the estimated solution and $\hat{x}_{qo}$ and $\hat{x}_{qr}$ are the quasi-estimates. In this section, we compute the probability of $\hat{x}_{qo}$ being closer than $\hat{x}_o$ to the solution, x, and the expected value of this probability under certain conditions.

Given the scenario in Fig. 2 where a and b are the end points of the solution domain and c is the center of this domain, there are four possibilities for the solution, x: (A) $x \in [a, \hat{x}]$, (B) $x \in [\hat{x}, c]$, (C) $x \in [c, \hat{x}_o]$ or (D) $x \in [\hat{x}_o, b]$. Let's examine each case separately.

Case (A): $x \in [a, \hat{x}]$ as illustrated in Fig. 3.

From Fig. 3, we note that $\hat{x}_{qo}$ is always closer than $\hat{x}_o$ to solution, x. Hence, when $x \in [a, \hat{x}]$, the probability that the quasi-opposition point is closer than the opposite point to the solution is

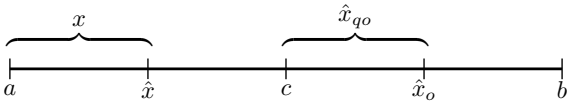

Figure 3. Solution domain with $x \in [a, \hat{x}]$

$$\Pr[|\hat{x}_{qo} - x| < |\hat{x}_o - x|] = 1 \text{ for } x \in [a, \hat{x}] \tag{4}$$

Case (B): $x \in [\hat{x}, c]$ as illustrated in Fig. 4.

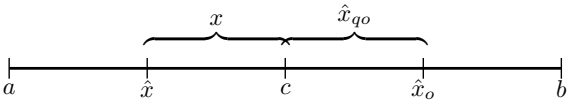

Figure 4. Solution domain with $x \in [\hat{x}, c]$

x is still always closer to $\hat{x}_{qo}$ than $\hat{x}_o$. Hence, when $x \in [\hat{x}, c]$,

$$\Pr[|\hat{x}_{qo} - x| < |\hat{x}_o - x|] = 1 \text{ for } x \in [\hat{x}, c] \tag{5}$$

Case (C): $x \in [c, \hat{x}_o]$ as illustrated in Fig. 5.

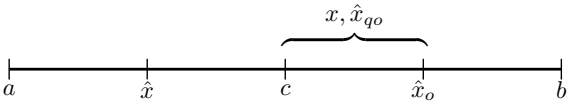

Figure 5. Solution domain, $x \in [c, \hat{x}_o]$

From Fig. 5 we see that

$$\begin{aligned}
&\Pr[|\hat{x}_{qo} - x| < |\hat{x}_o - x|] \text{ for } x \in [c, \hat{x}_o] \\
&= \Pr[|\hat{x}_{qo} - x| < \hat{x}_o - x \mid \hat{x}_{qo} - x < 0] \Pr[\hat{x}_{qo} - x < 0] \\
&+ \Pr[|\hat{x}_{qo} - x| < \hat{x}_o - x \mid \hat{x}_{qo} - x > 0] \Pr[\hat{x}_{qo} - x > 0] \\
&\quad = \Pr[\hat{x}_{qo} > 2x - \hat{x}_o \mid \hat{x}_{qo} < x] \Pr[\hat{x}_{qo} < x] \\
&\qquad + \Pr[\hat{x}_{qo} < \hat{x}_o \mid \hat{x}_{qo} > x] \Pr[\hat{x}_{qo} > x]
\end{aligned} \tag{6}$$

where we have used the total probability theorem [11]. If we assume that x and $\hat{x}_{qo}$ have uniform distribution in $[c, \hat{x}_o]$, then $\Pr[\hat{x}_{qo} < x] = \Pr[\hat{x}_{qo} > x] = \frac{1}{2}$. We can solve the first of the two expressions on the right side of Equation (6) as

$$\begin{aligned}
&\Pr[\hat{x}_{qo} > 2x - \hat{x}_o \mid \hat{x}_{qo} < x] \Pr[\hat{x}_{qo} < x] \\
&\quad = \frac{\Pr[\hat{x}_{qo} > 2x - \hat{x}_o,\, \hat{x}_{qo} < x]}{\Pr[\hat{x}_{qo} < x]} \\
&\quad = \frac{\Pr[2x - \hat{x}_o < \hat{x}_{qo} < x]}{\Pr[\hat{x}_{qo} < x]} \\
&\quad = 2 \int_c^{\hat{x}_o} \int_{2x - \hat{x}_o}^{x} \mathrm{f}(x, \hat{x}_{qo})\, \mathrm{d}\hat{x}_{qo} \mathrm{d}x
\end{aligned} \tag{7}$$

where $\mathrm{f}(x, \hat{x}_{qo})$ is the joint probability distribution function of x and $\hat{x}_{qo}$. This expression can be simplified as

$$2 \int_c^{\hat{x}_o} \int_{2x - \hat{x}_o}^{x} \left(\frac{1}{\hat{x}_o - c} \right)^2 \mathrm{d}\hat{x}_{qo} \mathrm{d}x = \frac{1}{4} \tag{8}$$

Since $\Pr[\hat{x}_{qo} < \hat{x}_o \mid \hat{x}_{qo} > x] = 1$ when $x \in [c, \hat{x}_o]$, we can solve Equation (6) as

$$\Pr[|\hat{x}_{qo} - x| < |\hat{x}_o - x|] = \frac{1}{4} + (1) \left(\frac{1}{2} \right) = \frac{3}{4} \text{ for } x \in [c, \hat{x}_o] \tag{9}$$

Case (D): $x \in [\hat{x}_o, b]$ as illustrated in Fig. 6.

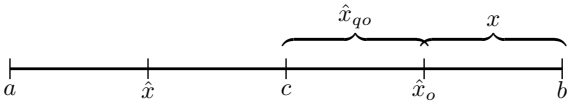

Figure 6. Solution domain, $x \in [\hat{x}_o, b]$

From Fig. 6, we see that $\hat{x}_o$ is always closer than $\hat{x}_{qo}$ to x. Hence,

$$\Pr[|\hat{x}_{qo} - x| < |\hat{x}_o - x|] = 0 \text{ for } x \in [\hat{x}_o, b] \tag{10}$$

Equations (4), (5), (9), and (10) can be combined to calculate the probability of the quasi-opposition point being closer than the opposite point to the solution in the domain $[a, b]$:

$$\begin{aligned}
&\Pr[|\hat{x}_{qo} - x| < |\hat{x}_o - x|] \\
&\quad = \frac{1(\hat{x} - a) + 1(c - \hat{x}) + \frac{3}{4}(\hat{x}_o - c) + 0(b - \hat{x}_o)}{b - a} \\
&\quad = \frac{\frac{1}{8}(a + b) - a + \frac{3}{4}\hat{x}_o}{b - a}
\end{aligned} \tag{11}$$

Assuming that the domain is symmetric, that $b = -a$, and that $\hat{x}_o$ has a uniform distribution, we can calculate the expected probability as

$$\mathrm{E}\left(\Pr[|\hat{x}_{qo} - x| < |\hat{x}_o - x|]\right) = \frac{1}{b} \int_0^b \frac{b + \frac{3}{4}\hat{x}_o}{2b}\, \mathrm{d}\hat{x}_o = \frac{11}{16} \tag{12}$$

C. Comparing Quasi-opposition with Quasi-reflection

Equation (12) shows that the expected probability of $\hat{x}_{qo}$ being closer than $\hat{x}_o$ to the solution is $\frac{11}{16}$ for a symmetric domain. The performance of other types of opposite points is investigated by calculating their corresponding expected probabilities. Table I lists the findings of this analysis.

Row 6 in Table I illustrates that the quasi-opposite and quasi-reflected estimates have the same probability of being closer to the solution. However, when comparing to the original estimated solution, $\hat{x}$, the quasi-reflected estimate (shown in Row 3, with an expected probability of $\frac{11}{16}$) has a higher chance of being closer to the solution than the quasi-opposite one (shown in Row 2, with an expected probability of $\frac{9}{16}$).

Table I
PROBABILISTIC COMPARISON OF VARIOUS OPPOSITION METHODS

Row	Methods	Probability
1	$\mathrm{E}\left(\Pr[\lvert\hat{x}_o - x\rvert < \lvert\hat{x} - x\rvert]\right)$	$1/2$
2	$\mathrm{E}\left(\Pr[\lvert\hat{x}_{qo} - x\rvert < \lvert\hat{x} - x\rvert]\right)$	$9/16$
3	$\mathrm{E}\left(\Pr[\lvert\hat{x}_{qr} - x\rvert < \lvert\hat{x} - x\rvert]\right)$	$11/16$
4	$\mathrm{E}\left(\Pr[\lvert\hat{x}_{qo} - x\rvert < \lvert\hat{x}_o - x\rvert]\right)$	$11/16$
5	$\mathrm{E}\left(\Pr[\lvert\hat{x}_{qr} - x\rvert < \lvert\hat{x}_o - x\rvert]\right)$	$9/16$
6	$\mathrm{E}\left(\Pr[\lvert\hat{x}_{qo} - x\rvert < \lvert\hat{x}_{qr} - x\rvert]\right)$	$1/2$

Note that Row 4 in Table I lists the expected probability for quasi-opposite population to be $\frac{11}{16}$ when compared against the opposite population, but in Row 2, the probability drops to $\frac{9}{16}$ when it is compared against the original population, $\hat{x}$. Thus, we can deduce that when using $\hat{x}_{qo}$, the quasi-opposite population should be compared against the opposite population for the highest probability of success. However, comparing the quasi-opposite population to the opposite population will come with a cost of extra fitness evaluation of the opposite population. Therefore, the most cost effective opposition method with the highest probability of being closer to the solution is to create a quasi-reflected population and choose the fittest individuals among the original and reflected populations, as illustrated in Row 3 of Table I.

D. Effects of Dimension on Opposition-Based Learning

Previous sections demonstrated the advantages of opposite points on one-dimensional problems using expected probabilities. In Fig. 7, we present the expected probabilities of success as the problem dimension increases. Figure legends have been abbreviated for clarification purposes. Entry of $\mathrm{E}\left(\Pr\left[\hat{x}_o, \hat{x}\right]\right)$ is shorthand for $\mathrm{E}\left(\Pr\left[\lvert\hat{x}_o - x\rvert < \lvert\hat{x} - x\rvert\right]\right)$ or the expected probability of $\hat{x}_o$ being closer to the solution than $\hat{x}$. These results are obtained using a MATLAB® simulation. According to these findings, for a 20-dimensional problem, such as the ones presented in Section V, the quasi-reflected estimate is 91 percent more likely to be closer to the solution than the initial estimate. Fig. 7 also demonstrates that the effectiveness of quasi populations increases with the problem's dimension.

IV. OPPOSITIONAL BIOGEOGRAPHY-BASED OPTIMIZATION ALGORITHM

Oppositional BBO (OBBO, also referred as OBƁO) utilizes opposition-based learning as well as other enhancements to accelerate BBO. Table II outlines the structure of OBƁO code. Like many other EAs, OBƁO benefits from elitism; on the other hand, note the absence of mutation in this research. OBƁO starts by creating a random population and replacing duplicates with randomly-generated individuals (Steps 1 and 2). The cost of each individual is calculated and an opposite population is created (Steps 3 and 4, given in more detail in Table III). The elite population is stored separately before performing partial immigration-based BBO (Steps 6 and 7). Once again, any duplicates that exist are replaced and the cost of the population is calculated (Steps 8 and 9). Next, the quasi-reflected population is created as discussed in Section IV-A

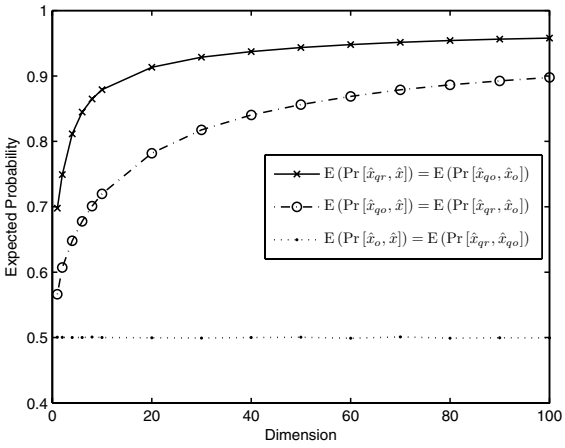

Figure 7. Effects of dimension on expected probabilities of various opposition methods. $\mathrm{E}\left(\Pr\left[\hat{x}_{qr}, \hat{x}\right]\right)$ is the expected value of the probability that $\hat{x}_{qr}$ is closer to the solution than $\hat{x}$. Other legends can be read similarly.

(Step 10) and the least fit are replaced with the elite of the previous generation (Step 11). Later, the population is searched for duplicates and sorted based on fitness values (Steps 12 and 13). Finally, unless the termination criteria is met, the next generation begins (Step 14).

Table II
OBBO ALGORITHM

1) Initialize Population
2) Remove / Replace Duplicates
3) Fitness Evaluation
4) Initialize Opposite Population
5) Start the BBO Generation Loop
6) Store Elite Individuals
7) BBO Migrations
8) Remove / Replace Duplicates
9) Fitness Evaluation
10) Opposite Population Jumping
11) Restore Elite Individuals
12) Remove / Replace Duplicates
13) Sort Population
14) Go to Step 5

A. Implementation of Opposite Population Jumping

Based on the benefits of quasi-reflection presented in Section III-C, we choose to implement quasi-reflected opposition. The OBL algorithm, shown in Table III, is called after J_r generations, where the jumping rate, $J_r \in [0, 1]$, is a control parameter set by the user to jump, or skip, opposite population creation at certain generations to save computational time. OBƁO adds dynamic domain scaling to expedite the optimization process. Dynamic domain scaling means that the opposite population is created within the current generation's domain, instead of the initial domain defined by the user or the domain of a single individual. As the generations progress and the estimated solution converges, the dynamic scaling allows the reflection domain to shrink and create opposite population within a smaller range.

Moreover, we include a reflection weight, κ, which determines the amount of reflection based on the solution fitness. κ forces the least fit individuals to be compared against their furthest possible reflection, whereas the fitter solutions will be reflected to a nearby point. After generating the quasi-reflected population, the function in Table III compares the current population and its quasi-reflection to select the fittest among them. Because the quasi-reflected population's fitness has to be evaluated, OBƁO has to converge faster (with respect to generation count) than original BBO in order to maintain the same CPU load. A benchmark method based on number of cost function calls is introduced in Section V-B to take this into consideration.

Table III
QUASI-REFLECTED OPPOSITE POPULATION JUMPING PSEUDOCODE, WHERE N_p IS THE POPULATION SIZE, D IS THE PROBLEM DIMENSION, THE P IS THE CURRENT POPULATION, OP IS THE OPPOSITE POPULATION, AND THE SUBSCRIPTS ind AND var ADDRESS THE var-TH SOLUTION FEATURE IN THE ind-TH INDIVIDUAL.

1) Evaluate function, if selected by J_r:
   `if (rand > Jr) {quit}`
2) Find the absolute min, max and median for the whole population
3) Create reflection weight, $\kappa \in [0, 1]$, which determines the reflection amount based on individual's fitness
4) Create OP based on the dynamic domain and reflection weight:
   `for ind = 1 to Np`
   `  for var = 1 to D`
   `    //Create a quasi-reflected number between`
   `    //the current variable and the median`
   `    if` $P_{ind,var}$ `< Median`
   $OP_{ind,var} = P_{ind,var} + (\texttt{Median} - P_{ind,var})\kappa_{ind}$
   `    else`
   $OP_{ind,var} = \texttt{Median} + (P_{ind,var} - \texttt{Median})\kappa_{ind}$
5) Calculate the fitness of Opposite Population
6) $P =$ the fittest N_p individuals in P and OP

V. EMPIRICAL ANALYSIS

A. Benchmark Functions

Sixteen benchmark functions are implemented to compare the performance of OBƁO and BBO. Information about these benchmark functions is shown in Table IV. More information on these functions can be found in [1], [5], [12], [13]. The benchmark functions are selected to provide a variety of challenges to OBƁO as each function meets different criteria: mulitmodality, nonseparablity or irregularity. All these functions are general enough to be implemented in any number of dimensions.

Note that the Penalty 1 and Penalty 2 functions, also called Generalized Penalized Functions [12], have typographical errors in most of the literature [14], [15], [16], [17], including some heavily-referenced articles [12], [13]. Readers should refer to Equations 25 and 26 in the original publication [18] for the correct equations.

B. Simulation Settings

Performance analysis presented in this paper is based on the number of cost function evaluations, F_c, performed before reaching the desired solution range. This comparison method is popular in the literature [19] since, generally, the computation of fitness/cost function consumes most of the CPU's resources. The accepted solution range is calculated by a method proposed in [20]:

Table IV
BENCHMARK FUNCTIONS WHERE n IS THE PROBLEM DIMENSION

Function	Domain	argmin	min f(x)
Ackley	$(-32, 32)^n$	0^n	0^n
Alpine	$(-10, 10)^n$	0^n	0^n
Fletcher/Powell	$(-\pi, \pi)^n$	rand$(-\pi, \pi)^n$	0^n
Griewank	$(-600, 600)^n$	0^n	0^n
Penalty1	$(-50, 50)^n$	1^n	0^n
Penalty2	$(-50, 50)^n$	1^n	0^n
Quartic	$(-1.28, 1.28)^n$	0^n	0^n
Rastrigin	$(-5.12, 5.12)^n$	0^n	0^n
Rosenbrock	$(-30, 30)^n$	1^n	0^n
Schwefel 1.2	$(-100, 100)^n$	0^n	0^n
Schwefel 2.21	$(-100, 100)^n$	0^n	0^n
Schwefel 2.22	$(-10, 10)^n$	0^n	0^n
Schwefel 2.26	$(-500, 500)^n$	420.9687^n	$(-418.9829n)^n$
Sphere	$(-100, 100)^n$	0^n	0^n
Step	$(-100, 100)^n$	0^n	0^n
Zakharov	$(-5, 10)^n$	0^n	0^n

$$|f - \hat{f}| < \epsilon_1 |f| + \epsilon_2 \tag{13}$$

where f is the global minimum, $\hat{f}$ is the minimum obtained by the EA, and ϵ_1 and ϵ_2 are small positive numbers, taken as 10^{-4} in this paper. All the benchmarks are computed in 20 dimensions with a population size of 50 over 50 Monte Carlo simulations. In order to avoid infinite run times, we introduced a function evaluation limit of 5,000,000. For OBƁO, jumping rate constant, J_r, is set to 0.3 [5] and finally, for both algorithms, top two solutions in each generation are preserved under elitism.

C. Empirical Results

Table V presents the effects of quasi-reflection and κ on BBO. The table lists the average number of function calls for successful runs, Equation (13), and the success rate, SR, which is defined as the ratio of number of successful runs to the number of trials.

Special attention should be paid to the penalized (Penalty 1 and 2) and noisy (Quartic) problems as these challenges occur frequently in real-world applications. OBƁO provided significant performance boost on these problems. Several conclusions can be drawn from Table V. Problems such as Rosenbrock and Schwefel 2.21 that could not be solved with BBO have a 100% success rate with OBƁO . For all of the benchmarks except Fletcher, the success rate increased when possible and the number of function calls was reduced. The use of OBL increased the average SR of BBO from 70% to 94% and decreased average number of function calls from 370,801 to 5,793 which is a 98% improvement. Based on these analyses, we can note that OBƁO significantly improves BBO's performance while reducing the number of cost function evaluations.

Table V
MEAN OF FUNCTION CALLS MADE FOR SUCCESSFUL RUNS, AND THE SUCCESS RATE, SR

Benchmark	BBO		OBƁO	
Functions	Mean Fc	SR	Mean Fc	SR
Ackley	23,150	1	2,394	1
Alpine	14,293	1	9,430	1
Fletcher	-	0	-	0
Griewank	372,488	0.24	2,102	1
Penalty1	39,092	1	1,513	1
Penalty2	37,082	1	1,678	1
Quartic	168,375	1	27,050	1
Rastrigin	4,997	1	2,111	1
Rosenbrock	-	0	8,223	1
Schwefel 1.2	2,796,393	0.5	4,893	1
Schwefel 2.21	-	0	8,110	1
Schwefel 2.22	13,841	1	2,977	1
Schwefel 2.26	124,248	1	8,092	1
Sphere	4,902	1	1,240	1
Step	157,187	1	997	1
Zakharov	1,064,367	0.48	6,086	1
Mean	370,801	0.70	5,793	0.94

VI. CONCLUSION

In this paper, we presented a new flavor of BBO, entitled oppositional BBO or OBƁO . The proposed algorithm accelerated BBO's performance by incorporating opposition-based learning, dynamic domain scaling and weighted reflections.

Also, a new variation of opposition-based learning named quasi-reflection is introduced. We proved the contribution of quasi-reflection and quasi-opposition by calculating the expected probability of the quasi-opposite population being closer to the solution then the original population for single-dimensional case. We extended this analysis to higher dimensions through computer simulations and determined that the performance of quasi-reflection and quasi-opposition improves with the problem's dimension. Later, we explained that using quasi-reflection along with BBO's estimate has a higher probability of yielding an answer closer to the solution than any other opposition-based method, while requiring the least computational effort. Thus, quasi-reflection was the preferred oppositional algorithm to create OBƁO.

OBƁO significantly outperforms BBO. A collection of 16 well-known 20-dimensional problems were used for this analysis and the average success rate and the average number of cost function evaluations for successful runs of BBO and OBƁO were compared. Overall, OBƁO was able to increase the success rate from 70% to 94% while reducing the mean cost function evaluation by 98%. Thus, the results presented strongly endorse the proposed enhancements.

The presented work can be expanded by finding analytical expressions for the higher dimensional probabilities of quasi-reflected and quasi-opposite populations. A statistical hypothesis test, such as Chi-square test, can be applied to analyze the significance of dynamic domain scaling (the domain of a variable is based on the domain of the whole population) versus the domain as defined in other opposition-based learning algorithms (the domain of a variable is based on the domain of the individual). Finally, even though quasi-reflection improves BBO's performance considerably, neither algorithm could solve the Fletcher problem. Further investigation should be made to explore the reasons behind this.

REFERENCES

[1] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[2] D. Simon, M. Ergezer, and D. Du, "Markov analysis of biogeography-based optimization." Available online at http://academic.csuohio.edu/simond/bbo/markov/.

[3] D. Simon, M. Ergezer, and D. Du, "Population distributions in biogeography-based optimization with elitism." Available online at http://academic.csuohio.edu/simond/bbo/markov/.

[4] D. Simon, "A probabilistic analysis of a simplified biogeography-based optimization algorithm." Available online at http://academic.csuohio.edu/simond/bbo/simplified/.

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

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

[7] H. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *Proceedings of International Conference on Computational Intelligence for Modelling Control and Automation*, vol. 1, pp. 695–701, 2005.

[8] M. Ventresca and H. Tizhoosh, "Improving the convergence of back-propagation by opposite transfer functions," in *IEEE International Joint Conference on Neural Networks*, pp. 9527–9534, 2006.

[9] H. Wang, Y. Liu, S. Zeng, H. Li, and C. Li, "Opposition-based particle swarm algorithm with cauchy mutation," in *IEEE Congress on Evolutionary Computation, Singapore*, pp. 4750–4756, 2007.

[10] A. R. Malisia, "Investigating the application of opposition-based ideas to ant algorithms," Master's thesis, University of Waterloo, 2007.

[11] A. Papoulis, S. Pillai, P. A, and P. SU, *Probability, random variables, and stochastic processes*. McGraw-Hill New York, 1965.

[12] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[13] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1980–1987, 2004.

[14] H. Zhang and J. Lu, "Adaptive evolutionary programming based on reinforcement learning," *Information Sciences*, vol. 178, no. 4, pp. 971 – 984, 2008.

[15] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 107–125, 2008.

[16] S. He, Q. Wu, J. Wen, J. Saunders, and R. Paton, "A particle swarm optimizer with passive congregation," *Biosystems*, vol. 78, no. 1-3, pp. 135 – 147, 2004.

[17] Z. Tu and Y. Lu, "A robust stochastic genetic algorithm (stga) for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 456–470, Oct. 2004.

[18] F. Aluffi-Pentini, V. Parisi, and F. Zirilli, "Global optimization and stochastic differential equations," *Journal of Optimization Theory and Applications*, vol. 47, no. 1, pp. 1–16, 1985.

[19] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 2005.

[20] A.-R. Hedar and M. Fukushima, "Minimizing multimodal functions by simplex coding genetic algorithm," *Optimization Methods and Software*, vol. 18, pp. 265 – 282, 2003.