Oppositional Biogeography-Based Optimization for Combinatorial Problems

Mehmet Ergezer, *Student Member*, Dan Simon, *Senior Member, IEEE*
Cleveland State University
Cleveland, OH 44115
Email: m.ergezer@csuohio.edu

***Abstract*—In this paper, we propose a framework for employing opposition-based learning to assist evolutionary algorithms in solving discrete and combinatorial optimization problems. To our knowledge, this is the first attempt to apply opposition to combinatorics. We introduce two different methods of opposition to solve two different type of combinatorial optimization problems. The first technique, open-path opposition, is suited for combinatorial problems where the final node in the graph does not have be connected to the first node, such as the graph-coloring problem. The latter technique, circular opposition, can be employed for problems where the endpoints of a graph are linked, such as the well-known traveling salesman problem (TSP). Both discrete opposition methods have been hybridized with biogeography-based optimization (BBO). Simulations on TSP benchmarks illustrate that incorporating opposition into BBO improves its performance.**

***Index Terms*—Biogeography-based optimization, opposition, combinatorics, discrete optimization, evolutionary algorithms, graph-coloring problem, traveling salesman problem.**

I. Introduction

Bio-inspired computing has been a fast-growing area in machine learning, producing popular topics such as artificial neural networks [1] and genetic algorithms [2]. Bio-inspired algorithms mimic the time-proven techniques developed in nature to tackle challenging tasks such as learning mechanisms or minimum seeking. Biogeography-based optimization (BBO) [3], [4] is an evolutionary algorithm, derived from the study of biogeography. BBO is inspired by the migration of species amongst islands and imitates this pattern to solve optimization problems.

Opposition-based learning (OBL) was first proposed as a machine intelligence scheme for reinforcement learning [5], [6] and has been employed to improve soft computing methods such as fuzzy systems [7], [8] and artificial neural networks [9]–[12]. Studies [13]–[16] illustrated the capabilities of OBL for solving continuous domain optimization problems by combining it with differential evolution. Since then, OBL has been employed to improve the success rate of various evolutionary algorithms such as biogeography-based optimization [17], particle swarm optimization [18]–[21], ant colony optimization [22], [23] and simulated annealing [24] in a wide range of fields from image processing [8], [25], [26] to system identification [27], [28].

Our previous research showed that augmenting BBO with opposition increased the success rate of BBO while reducing the computational effort for continuous domain problems [17]. In the meantime, BBO has been applied to discrete optimization problems such as the traveling salesman (TSP) [29]–[31] and knapsack problems [4]. Our goal is to show that OBL, which has improved continuous optimization, can also be modified alongside BBO to solve discrete problems including NP-complete problems such as graph coloring.

We refer to a graph as open if the endpoints of a path are not linked, and as closed if the endpoints are connected. We propose two novel discrete domain opposition algorithms for open and closed combinatorial problems. Section II reviews OBL methods for continuous domain problems and proposes two new techniques for combinatorial optimization. Section III provides an overview of BBO as an evolutionary algorithm and outlines oppositional BBO (OBBO). Section IV defines the vertex-coloring problem and the TSP and presents our comparison of oppositional BBO and standard BBO. Concluding remarks are offered in Section V.

II. Opposition in Optimization

This section reviews opposition as previously defined in the continuous domain and introduces open-path and circular opposition to solve open and closed combinatorial problems.

A. Opposition in Continuous Domain

Opposition-based learning has different variants employed for solving continuous domain optimization problems. These variants include opposition, quasi-opposition and quasi reflection. A variable, $\hat{x}$, is reflected through the center of a domain to create its opposite, $\hat{x}_o$ as defined below.

Definition Let $\hat{x}$ be any real number $\in [a, b]$. Its opposite, $\hat{x}_o$, is defined as

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

Quasi-opposition, defined below, reflects a variable to a random point between the center of the domain and $\hat{x}_o$.

Definition Let $\hat{x}$ be any real number $\in [a, b]$. Its quasi-opposite point, $\hat{x}_{qo}$, is defined as

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

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 $\hat{x}_o$.

978-1-4244-7833-0/11/$26.00 ©2011 IEEE

Quasi-reflection, defined below, shifts the variable $\hat{x}$ to a random point between the center of the domain and $\hat{x}$.

Definition Let $\hat{x}$ be any real number and $\hat{x} \in [a, b]$. Then the quasi-reflected point, $\hat{x}_{qr}$, is defined as

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

where $\text{rand}(c, \hat{x})$ is a random number uniformly distributed between c and $\hat{x}$.

Of three opposite points defined above, reference [17] mathematically proves that quasi-reflection yields the highest probability of being closer to the solution of the optimization problem. The proof assumes that the solution is uniformly distributed in the domain and the problem is one-dimensional. Simulation results for higher dimensions yield similar results. The definitions provided for continuous domain opposition in Eqs. 1-3 are for one-dimensional variables; however, they can easily be extended for a multidimensional search space.

Figure 1 illustrates the various opposite points $\in [a, b]$. Assuming that $\hat{x}$ is a solution candidate in an evolutionary algorithm (EA), its opposite and the range for its quasi-opposite and quasi-reflection are shown in this figure.

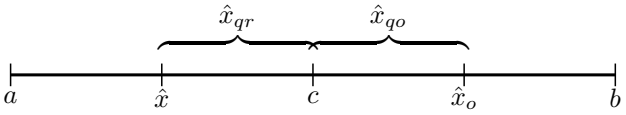

Fig. 1. Opposite points defined in domain $[a, b]$. c is the center of the domain and $\hat{x}$ is an EA individual. $\hat{x}_o$ is the opposite of $\hat{x}$, and $\hat{x}_{qo}$ and $\hat{x}_{qr}$ are the quasi-opposite and quasi-reflected points, respectively.

B. Opposition in Discrete Domain

The previous research that has been published on opposition is for solving continuous domain optimization problems. Recently, there has been research to extend BBO to combinatorial problems such as the traveling salesman problem (TSP) [29]–[31]. Oppositional learning, created for accelerating a continuous search space, can also be modified to be used alongside BBO to solve combinatorial problems, such as graph-coloring and TSP.

We recognize that attempting to apply opposition to a TSP path by simply reversing that path is meaningless because the reversed path will yield the same cost as the original path. For example, in a TSP, tour (1, 2, 3, 4) and its opposite tour, (4, 3, 2, 1), have the same cost because all of the cities preserve their neighbors. Therefore a different definition of opposition is needed. For the TSP, we define an opposite path as a path that maximizes the distance between the adjacent vertices in the original path. Based on this definition, a tour may have more than one opposite.

We propose two new definitions of opposition in discrete space. The first proposed algorithm is for open graph problems, where the final node may be disconnected from the first node, such as the graph-coloring problem. The latter opposition method is for closed walk problems, where the final node is linked to the first node, such as the symmetric TSP.

C. Open-path Opposition

The first method of opposition for discrete domain problems that we propose is open-path opposition. Open path are those that we complete their path when they reach the last vertex on the path. An example of such a problem would be the vertex coloring problem. Refer to Section IV-B for more information on graph-coloring.

In order to implement open-path opposition, proximities between nodes are calculated. If nodes share an edge so that they are directly connected, their proximity is defined as one. If nodes connected through another node, their proximity is two. If nodes are connected through two nodes, their proximity is three, and so on. Consider a path of four nodes, sorted as (1, 2, 3, 4). Table I lists the proximity between each nodes.

TABLE I
PROXIMITY OF NODES (1, 2, 3, 4) FOR CALCULATING THE OPPOSITE PATH.

Node 1	Node 2	Proximity
1	2	1
1	3	2
1	4	3
2	3	1
2	4	2
3	4	1

The opposite of this path would be a path that maximizes the proximity between adjacent nodes while minimizing the proximity between further nodes. Table II lists the original path and its calculated opposite. Numbers above the arrows illustrate the proximity between the nodes in the original path as shown in Table I. The goal of open-path opposition is to maximize the total proximity in a path by spreading the adjacent nodes apart. We can say that the greater the total proximity, the greater is the opposition. The maximum total proximity achievable for our example is seven and it is shown in Table II as the exact opposite path. A less opposite path, named greedy opposite, is also shown in the table. The greedy opposite path uses a greedy algorithm to quickly calculate the approximate opposite of a given path, although, it might not yield the highest degree of opposition.

TABLE II
OPPOSITE PATHS OF NODES IN A TOUR (1, 2, 3, 4) .

Tour	Path and proximities	Total Proximity
EA Individual	$1 \xrightarrow{1} 2 \xrightarrow{1} 3 \xrightarrow{1} 4$	3
Exact Opposite	$3 \xrightarrow{2} 1 \xrightarrow{3} 4 \xrightarrow{2} 2$	7
Greedy Opposite	$1 \xrightarrow{3} 4 \xrightarrow{2} 2 \xrightarrow{1} 3$	6

Notice that calculating the exact opposite is a combinatorial problem of its own, therefore a greedy approximation is developed. The greedy opposite is implemented to maximize the proximity one city at a time. For this example, based on Table I, nodes 1 and 4 have the highest distance between them, so they start the greedy opposite tour. Then, we find the node with the highest proximity to follow the previously discovered

city, and continue until the tour is completed. Because the greedy algorithm seeks the local optimum, it is unsuccessful in finding the exact opposite even for such a small problem. However, note that if we begin with city 3, then the greedy algorithm will find the exact opposite in this example.

Since there is no randomness involved in the definition of the opposite path, a greedy opposite path can be defined at the beginning of a program based on the node count and the opposite population can be created based on this path to save processing time. Reconsider of our example of four nodes. Seeing that the output of the greedy opposition algorithm is deterministic, we can use our greedy path from Table II to calculate the opposite of any other four-node path. To do this, we refer to (1, 2, 3, 4) as a list of node indices, instead of a list of nodes. Therefore, we can map any four-node map to its opposite.

For a given number of variables in a combinatorial problem, we can calculate its greedy opposite by using Algorithm 1.

Algorithm 1 Open-path greedy opposite algorithm

procedure GREEDY OPPOSITE PATH(n) ▷ n is the number of nodes
  for each node index n_i **do**
    if n_i is odd **then**
      $O_{n_i} = \frac{n_i+1}{2}$ ▷ O_{n_i} is the opposite node index
    else n_i is even
      $O_{n_i} = n + 1 - \frac{n_i}{2}$
    end if
  end for
  return O_{n_i}
end procedure

For the four-node problem, the greedy algorithm yields the greedy opposite path: $1 \rightarrow 4 \rightarrow 2 \rightarrow 3$. This greedy algorithm can be used to accelerate the convergence rate of various combinatorial problems, including the graph-coloring problem.

D. Circular Opposition

In Section II-C, we discussed opposition on a open path. However, some problems, such as the symmetric TSP, are closed since the endpoints of the graph are connected. Open-path opposition will not yield a high degree of opposition for these cases as it assumes that the extreme vertices have low proximity and the open-path algorithm attempts to bring them closer. Therefore, here we propose the opposite cycle as an alternative to the opposite path for problems with closed paths.

On a symmetric TSP, given a sequence of cities, starting at any city on the path, moving in either direction, we will return to our starting point and travel the same amount regardless of where we start. Thus, a closed path can be seen as a circular tour. Fig. 2 illustrates a symmetric TSP with eight cities on a circular path. This is an intuitive representation of this problem.

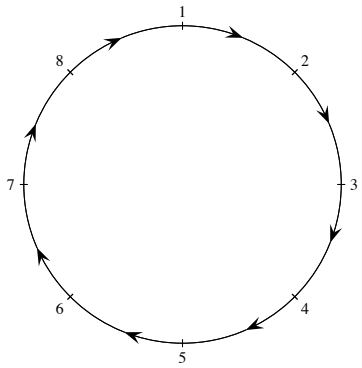

Fig. 2. Eight-city closed path where the path is represented as a circle.

Based on Fig. 2, we can see that to maximize the proximity between the adjacent vertices, we must travel to the opposite side of the circle. This is our definition of opposition for problems with a closed path. Fig. 3 illustrates the opposite of each city in the tour.

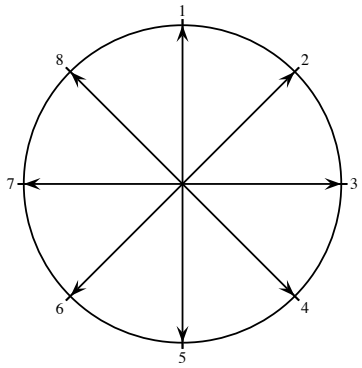

Fig. 3. Eight-city closed path problem with opposite cities indicated across the circular path.

Although Fig. 3 shows the opposite of each city, it does not indicate an opposite path. It reveals that if we start at city 1, its opposite is city 5. But where do we go from there? The opposite of city 5 is 1, but we cannot revisit the same city. The next best thing for us is to travel to city 2 or 8 since both would yield the same amount of opposition. We can choose either of these cities randomly or based on the opposition order, which is explained below. We continue this process until all cities are visited.

We can define permutations on our opposite circuits based on the direction in which we move around the circular path. We call this the order of opposition and four possibilities of it are presented in Table III. These permutations are named according to the direction we choose to advance. For example, CCW opposite indicates that after reaching an opposite city, we always move counter-clockwise around the circle to progress on the path. Thus, after we visit city 5, we start moving counter-clockwise to find the furthest vertex, in this case city 2. The CW opposite is similar, but advances in the clockwise direction to form an opposite cycle. Notice that the

TABLE III
PERMUTATIONS OF OPPOSITE TOUR OF CITIES $(1, 2, \ldots, 7, 8)$. TOURS ARE NAMED AFTER THE DIRECTION FOLLOWED AROUND THE OPPOSITION CIRCLE AFTER EACH CITY VISIT. FOR EXAMPLE, CW OPPOSITE INDICATES THAT FROM THE CURRENT LOCATION, ALGORITHM MUST TRAVEL CLOCKWISE AROUND THE OPPOSITE CIRCLE TO FIND THE LARGEST OPPOSITION.

Path Name	Path Followed
EA Individual	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$
CW Opposite	$1 \to 5 \to 2 \to 6 \to 3 \to 7 \to 4 \to 8$
CCW Opposite	$1 \to 5 \to 8 \to 4 \to 7 \to 3 \to 6 \to 2$
CW-CCW Opposite	$1 \to 5 \to 2 \to 6 \to 8 \to 4 \to 3 \to 7$
CCW-CW Opposite	$1 \to 5 \to 8 \to 4 \to 2 \to 6 \to 7 \to 3$

CW and CCW paths our after city 5 are mirror images of each other and yield the same amount of opposition. We can define the CW opposite path as follows:

Definition Let n be the number of nodes in a graph and P be an even node cycle. The CW opposite path, P_o^{CW}, is defined as

$$\begin{aligned} P &= [1, 2, \ldots, n] \\ P_o^{CW} &= \left[1, 1+\frac{n}{2}, 2, 2+\frac{n}{2}, \ldots, \frac{n}{2}-1, n-1, \frac{n}{2}, n\right] \end{aligned} \tag{4}$$

The other two techniques, CW-CCW and CCW-CW oppositions, reverse direction after each decision. For instance, if CW-CCW opposition moves clockwise to get to city 2, it would then advance counter-clockwise to city 6. Notice that CW-CCW and CCW-CW oppositions create less opposition than CW and CCW by themselves as we progress around the circle.

In our algorithm, we define the middle node to be the reflection point, r_p, calculate opposite cities based on r_p and link every city to its opposite. As future work, different reflection points can be selected to create different levels of opposition, analogous to $\hat{x}_{qo}$and $\hat{x}_{qr}$ in the continuous domain.

Notice that we cannot assign opposite cities as defined in Eq. 4 if n is odd. If we follow the opposite circle (Fig. 3) in an odd-length cycle, the opposite point would end up being between two cities. Then, the CW or CCW option would specify which direction to travel around the circle to find the opposite city.

One way of implementing CW opposition in an odd-length graphs is to add an auxiliary node to the end of the path to complete the city count to an even number. We then calculate the CW opposite of the tour and remove the auxiliary city from the end of tour.

III. PROPOSED ALGORITHM

A. Biogeography-based Optimization

Biogeography-based optimization (BBO) is an evolutionary algorithm proposed as a heuristic optimization technique [3]. BBO is inspired by the science of biogeography which studies the temporal and spatial distribution of species amongst islands.

Biogeography has been popularized by Darwin's zoological research on remote islands [32] and its contributions to the theory of evolution [33]. Islands provided all the necessary tools to biologists to study evolution since there is a large quantity of islands to study and they are isolated and thus minimally affected by outside disturbances.

Inspired by the success of biogeography in the theory of evolution, BBO was developed as an evolutionary algorithm for continuous domain problems, although the original paper searched over a discrete domain for optimal solutions (i.e., continuous problems were discretized) [3]. In BBO, a population of solution candidates are generated, each candidate representing an island. These islands are assigned immigration and emigration rates based on their fitness. The variables in each solution candidate are symbolized by species living on that island. Throughout iterations, species migrate amongst the islands based on the immigration and emigration rates to find a better habitat.

For instance, a solution candidate with higher fitness, a better solution, will have a higher emigration rate, so that it can share its features with less fit islands. At the same time, the same fitter island will have a lower immigration rate in order that it is less likely to be spoiled by migrations from less fit islands. A worse solution candidate will be assigned a high immigration rate in the hopes that the new species will enrich the fitness of the island.

To solve a continuous optimization problem , we search for the best solution that exists within a given domain. Combinatorial problems, such as the ones discussed in this research, are ordering type problems. We are given a list of all vertices that must form the solution and we are to find the best sequence of these vertices that will minimize the cost function. BBO has been employed to solve numerous engineering optimization problems in continuous domains [34]–[39]. References [30], [31] present migration schemes to solve the traveling salesman problem and compare BBO's performance to that of other evolutionary algorithms, including genetic algorithms and ant colony optimization. In this paper, we follow a migration pattern which is inspired by the inver-over operator [40] and modified for BBO in [41].

In the spirit of original BBO, all the islands are assigned emigration and immigration rates proportional to their fitness rankings. We then perform roulette wheel selection to determine an immigrating and an emigrating island, I_i and I_e, based on these rates, and randomly choose a city in the immigrating island to be our migration point, M_p. Next, we seek for the migration point in the emigrating island and locate the adjacent vertex as the flipping point, F_p. A new island is created from the immigrating island by flipping the sequence of vertices between M_p and F_p. Algorithm 2 presents the pseudocode for combinatorial BBO migration.

Algorithm 2 can be illustrated with the following example. Let the randomly selected migration point be $M_p = 3$ and the immigrating and emigrating islands be

$$\begin{aligned} I_i &= [1 \to 3^{M_p} \to 4 \to 6 \to 2 \to 5] \\ I_e &= [6 \to 4 \to 3 \to 2^{F_p} \to 1 \to 5] \end{aligned}$$

Considering that in I_e, M_p is followed by city 2, $F_p = 2$. We

Algorithm 2 Combinatorial BBO migration

procedure MIGRATION(I_i, I_e)
  $M_p = \text{rand}(I_i(city))$ ▷ Random migration point
  $F_p = I_e(M_p + 1)$ ▷ Flip point is adjacent to M_p
  I_{new} =Flip $I_i(M_p + 1 : F_p)$
  return I_{new}
end procedure

then flip the cities between M_p and F_p in I_i and obtain

$$I_{new} = [1 \rightarrow 3^{M_p} \rightarrow 2^{F_p} \rightarrow 6 \rightarrow 4 \rightarrow 5]$$

B. Oppositional Biogeography-based Optimization

The opposition algorithm is added to BBO as a diversity mechanism to improve BBO's convergence rate. The outline of the oppositional BBO algorithm is presented in Algorithm 3. In order to save computational time, the opposite population has a chance of being generated 30% of the time. This probability is determined by the $Opposition Jumping Rate \in [0, 1]$ parameter as specified in Algorithm 3.

Algorithm 3 Oppositional BBO

procedure OBBO($Problem, Opposition\ method$)
  Randomly generate initial population, P
  Generate the opposite of initial population, OP
  Maintain the fittest amongst P and OP
  while Generation $\leq gen\ limit$ **do**
    Perform BBO Migration ▷ Algorithm 2
    Remove duplicates from population
    Calculate the fitness of P
    if $random \leq Opposition\ Jumping\ Rate$ **then**
      Create the opposite population, OP
      Calculate the fitness of OP
      Maintain the fittest amongst P and OP
    end if
    Restore Elite individuals
  end while
  return $Best\ Individual$
end procedure

IV. EXPERIMENTAL RESULTS

In this section, 16 vertex coloring and 16 traveling salesman benchmark problems are simulated on MATLAB® with the settings listed in Table IV. The tabulated results are the mean of the best findings over 20 independent Monte Carlo simulations at the end of 500 generations.

TABLE IV
SIMULATION SETTINGS FOR COMBINATORIAL PROBLEMS.

Variable	Value
Population size	50
Generation limit	500
Number of elites	3
Monte Carlo runs	20
Opposition Jumping Rate	0.3

TABLE V
SYMMETRIC TSP BENCHMARK PROBLEMS AND THEIR OPTIMAL RESULTS AS POSTED BY TSPLIB [48].

Benchmark	Optimal Solution	Dimension
att532	27,686	532
berlin52	7,542	52
bier127	118,282	127
ch130	6,110	130
d18512	645,238	18,512
kroA150	26,524	150
kroA200	29,368	200
kroC100	20,749	100
lin105	14,379	105
lin318	42,029	318
p654	34,643	654
rat575	6,773	575
rl11849	923,288	11,849
st70	675	70
usa13509	19,982,859	13,509
vm1084	239,297	1,084

A. Traveling salesman problem

The TSP [42] is a well-known closed path combinatorial problem. The TSP is classified as an NP-hard problem and currently there is no polynomial-time algorithm that can guarantee an optimal solution. In the TSP, we are given a list of cities and their coordinates. We sort this list to minimize the length of the closed path traveled while only visiting each city once. This problem is based on the challenge faced by the traveling salesman who tries to find the shortest route which would allow him to visit all the cities once before returning to the departure city. The TSP represents many real-world applications such as the vehicle routing problem (i.e. for postal services or buses) [43]–[45], and printed circuit board (PCB) drilling problems [46], [47]. For instance, to manufacture a PCB, tens of thousands of holes must be drilled to place components. The solution of the TSP, where the cities represent the holes, would portray the path the drill must follow from one hole to the next.

For this paper, we focus solely on the symmetric traveling salesman problem where the distance between two nodes is identical when traveling in either direction. The set of TSP benchmark problems are borrowed from TSPLIB [48]. Table V lists these benchmark problems, their dimensions and minimum costs. For our simulations, we chose to implement clockwise (CW) circular opposition, Table III, as our opposite algorithm.

The mean (out of 20 Monte Carlo simulations) of the best results obtained from BBO and oppositional BBO is represented in Table VI along with the geometric mean. BBO with CW circular opposition, BBO/CO, is able to find a shorter route for 14 of the benchmark problems while BBO has a better route for two problems. Note that we use a relatively small population size and relatively few generations, so the BBO solutions are not close to the optimal solutions. However, our main point in this paper is to perform relative comparison between BBO and BBO/CO.

TABLE VI
MEAN OF THE BEST SOLUTIONS OBTAINED BY BBO AND BBO WITH CIRCULAR OPPOSITION (BBO/CO) TO SYMMETRIC TSP BENCHMARK PROBLEMS.

Benchmark	BBO	BBO/CO
att532	1,154,304	**1,140,103**
berlin52	**9,795**	9,811
bier127	302,056	**298,700**
ch130	20,552	**20,304**
d18512	58,521,418	**58,369,040**
kroA150	109,793	**108,651**
kroA200	169,256	**165,191**
kroC100	**57,509**	57,799
lin105	42,005	**41,661**
lin318	375,896	**374,011**
p654	1,440,864	**1,422,779**
rat575	83,835	**82,699**
rl11849	85,134,513	**84,926,068**
st70	1,162	**1,147**
usa13509	2,105,421,221	**2,098,340,568**
vm1084	7,208,117	**7,142,633**
Geometric Mean	493,973	**489,864**

B. Vertex Coloring

Vertex coloring [49] is the most studied graph-coloring problem since the other coloring problems can be transformed into it. Graph-coloring has many real-word applications related to scheduling, including register allocation [50], wireless network testing [51] and final exam timetables at universities [52].

In vertex coloring, we are giving a a graph $G(V, E)$ denoting list of countries on a map (vertices) and their neighbors (edges). The neighboring cities are represented as vertices that are linked with an edge. Connected vertices cannot share the same color. The goal is to find the minimum number of of colors needed to color the vertices. This number is denoted as the chromatic number, $\chi(G)$. Vertex coloring is an NP-complete problem.

Figure 4 illustrates a three-color graph-coloring problem and its solution. In this problem, there are eight countries (vertices) and 13 connections (edges). The minimum number of colors needed is $\chi(G) = 3$.

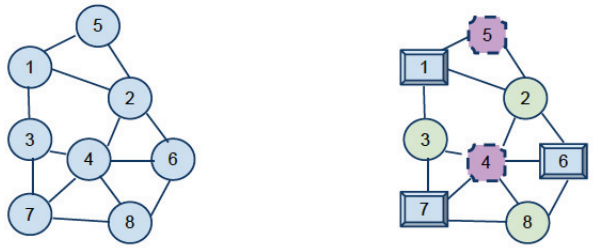

Fig. 4. Example of a three-color map with eight vertices and 13 edges. The figure on the right is the minimally colored map.

Various evolutionary approaches have been created to solve the graph-coloring problem [53]–[55]. Our method is a hybrid between an evolutionary algorithm (BBO) and the greedy algorithm described in Algorithm 5. This technique is similar to the hybrid genetic algorithm scheme presented in [56]. The role of BBO is to sort the list of countries and to provide this re-ordered list to the greedy algorithm which quickly assigns a color to each country. This simple methodology does not guarantee that an optimal solution can be found, however, it stores the vertices as a list so that open-path opposition can be easily applied.

Each BBO individual in the population stores a list of vertices as its solution features (islands). Vertices are rearranged through the generations and conveyed to the greedy algorithm to minimize the chromatic number. Algorithm 4 outlines the hybrid BBO/Greedy algorithm.

Algorithm 4 Vertex coloring with BBO

```
1: procedure BBO COLORING(V, E)
2:     Initialize population by shuffling the order of vertices
3:     while Generation count is not reached do
4:         Perform BBO migration on the order of vertices ▷ Algorithm 2
5:         Cost function calls Greedy Vertex ▷ Algorithm 5
6:     end while
7: end procedure
```

The goal of the greedy algorithm is to quickly assign a valid color to each country based on the order of vertices generated by BBO. Algorithm 5 presents the pseudocode for the greedy vertex coloring algorithm.

Algorithm 5 Greedy vertex coloring

```
1: procedure GREEDY VERTEX(V, E)
2:     for each vertex do
3:         Find all of its neighbors
4:         Find the colors of all the neighbors
5:         Assign the smallest available color index that is not assigned to a neighbor
6:     end for
7:     return number of colors
8: end procedure
```

Table VII lists the benchmark problems borrowed from [57] which are assembled from various resources [53], [58], [59]. The table lists the number of vertices and edges for each problem along with the chromatic number, $\chi(G)$, if available.

Simulation results for graph-coloring benchmarks are depicted in Table VIII. These are the mean of the best results obtained from each algorithm after 20 independent Monte Carlo simulations. We note that BBO augmented with open-path opposition (BBO/OPO) reaches the chromatic number in six of the 16 benchmarks within 500 generations. However, it can outperform BBO in only one problem. As future work, different degrees of open-path opposition can be created and compared against BBO.

TABLE VII
LIST OF BENCHMARK PROBLEMS ALONG WITH THEIR OPTIMAL SOLUTION FOR VERTEX COLORING. "NA" INDICATES NOT AVAILABLE (I.E., NOT KNOWN).

Benchmark	$\chi(G)$	# Vertices	# Edges
anna	11	138	493
david	11	87	406
DSJC125.1	NA	125	1472
DSJR500.1	NA	500	7110
huck	11	74	301
le450.5a	5	450	5714
miles750	31	128	2113
myciel3	4	11	20
myciel4	5	23	71
myciel5	6	47	236
myciel6	7	95	755
queen10.10	NA	100	2940
queen11.11	11	121	3960
queen5.5	5	25	160
queen6.6	7	36	290
queen7.7	7	49	476

TABLE VIII
MEAN OF THE BEST RESULTS OBTAINED BY BBO AND BBO/OPO (OPEN-PATH OPPOSITION) ALGORITHMS AFTER 100 GENERATIONS FOR GRAPH-COLORING PROBLEMS.

Benchmark	BBO	BBO/OPO
anna	11	11
david	11	11
DSJC125.1	**11**	12
DSJR500.1	**19**	20
huck	11	11
le450.5a	30	30
miles750	35	**34**
myciel3	4	4
myciel4	5	5
myciel5	**6**	7
myciel6	11	11
queen10.10	**21**	22
queen11.11	25	25
queen5.5	**7**	8
queen6.6	**10**	11
queen7.7	**12**	13
Geometric Mean	**12.0**	12.4

V. CONCLUSION

In this paper, we presented the TSP as a circular graph and graph coloring as an open-ended graph. Corresponding opposition algorithms, circular and open-path opposition, have been introduced to assist our evolutionary algorithm of choice, BBO, to solve combinatorial optimization problems. The objective of both opposition methods was to create an opposite path by maximizing the proximity between adjacent nodes.

The circular opposition technique was developed for graphs where the last node was linked to the first one. The circular opposition was tested on 16 traveling salesman problems and was found to outperform standard BBO in 14 of them.

The open-path opposition was introduced for open-ended combinatorics and was tested on 16 graph-coloring problems. BBO/OPO was able to reach the optimal solution in six of these benchmarks, but could only surpass BBO on one of the problems.

Further research could focus on combining the proposed opposition methods with other EAs and comparing the effect of opposition on other combinatorial optimization algorithms. The statistical significance of the oppositional EAs should also be analyzed. Furthermore, future research efforts could concentrate on exploring different degrees of opposition, motivated by its continuous-domain counterpart, for open- and closed-path combinatorics.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments and suggestions that helped to improve this paper. This work was supported by NSF Grant 0826124 in the CMMI Division of the Engineering Directorate.

REFERENCES

[1] J. Dayhoff and J. DeLeo, "Artificial neural networks," *Cancer*, vol. 91, no. S8, pp. 1615–1635, 2001.
[2] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs (3rd ed.)*. London, UK: Springer-Verlag, 1996.
[3] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.
[4] D. Simon, M. Ergezer, and D. Du, "Markov models of biogeography-based optimization," *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, vol. 41, pp. 299–306, 2011.
[5] H. Tizhoosh, "Reinforcement learning based on actions and opposite actions," in *International Conference on Artificial Intelligence and Machine Learning*, 2005.
[6] ——, "Opposition-based reinforcement learning," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 10, no. 4, pp. 578–585, 2006.
[7] H. Tizhoosh and F. Sahba, "Quasi-global oppositional fuzzy thresholding," pp. 1346–1351, 2009.
[8] H. Tizhoosh, "Opposite fuzzy sets with applications in image processing," in *Proceedings of International Fuzzy Systems Association World Congress*, Lisbon, Postugal, 2009, pp. 36–41.
[9] M. Ventresca and H. Tizhoosh, "Improving the convergence of backpropagation by opposite transfer functions," in *IEEE International Joint Conference on Neural Networks*, 2006, pp. 9527–9534.
[10] ——, "Opposite transfer functions and backpropagation through time," in *Foundations of Computational Intelligence, 2007. FOCI 2007. IEEE Symposium on*. IEEE, 2007, pp. 570–577.
[11] M. Shokri, H. Tizhoosh, and M. Kamel, "Opposition-based q (lambda) with non-markovian update," in *Proc. IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL 2007), Hawaii*, 2007, pp. 288–295.
[12] M. Rashid and A. Baig, "Improved Opposition-Based PSO for Feedforward Neural Network Training," in *Information Science and Applications (ICISA), 2010 International Conference on*, 2010, pp. 1–6.
[13] S. Rahnamayan, "Opposition-based differential evolution," Systems Design Engineering, University of Waterloo, 2007.
[14] S. Rahnamayan, H. Tizhoosh, and M. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.
[15] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Quasi-oppositional differential evolution," in *Proc. IEEE Congress on Evolutionary Computation CEC 2007*, 2007, pp. 2229–2236.
[16] S. Rahnamayan and G. G. Wang, "Solving large scale optimization problems by opposition-based differential evolution (ode)," *WSEAS Transactions on Computers*, vol. 7, pp. 1792–1804, October 2008.
[17] M. Ergezer, D. Simon, and D. Du, "Oppositional biogeography-based optimization," in *IEEE International Conference on Systems, Man and Cybernetics*, 2009, pp. 1009–1014.
[18] 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*, 2007, pp. 4750–4756.
[19] F. Shahzad, A. Baig, S. Masood, M. Kamran, and N. Naveed, "Opposition-Based Particle Swarm Optimization with Velocity Clamping," *Advances in Computational Intelligence*, pp. 339–348, 2009.

[20] J. Tang and X. Zhao, "An enhanced opposition-based particle swarm optimization," in *Intelligent Systems, 2009. GCIS'09. WRI Global Congress on*, vol. 1. IEEE, 2009, pp. 149–153.

[21] Y. Chi and G. Cai, "Particle swarm optimization with opposition-based disturbance," in *Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on*, vol. 2. IEEE, 2010, pp. 223–226.

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

[23] A. Malisia, "Improving the exploration ability of ant-based algorithms," in *Oppositional Concepts in Computational Intelligence*. Springer, 2008, pp. 121–142.

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

[25] F. Khalvati, H. Tizhoosh, and M. Aagaard, "Opposition-based window memoization for morphological algorithms," in *IEEE Symposium on Foundations of Computational Intelligence, Honolulu, Hawaii, USA*, 2007, pp. 425–430.

[26] S. Rahnamayan and H. Tizhoosh, "Image thresholding using micro opposition-based differential evolution (micro-ode)," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE, 2008, pp. 1409–1416.

[27] D. Subudhi, B; Jena, "Nonlinear system identification using opposition based learning differential evolution and neural network techniques," *IEEE Journal of Intelligent Cybernetic Systems*, vol. 1, pp. 1–13, 2009.

[28] B. Subudhi and D. Jena, "A differential evolution based neural network approach to nonlinear system identification," *Applied Soft Computing*, vol. 11, pp. 861 – 871, 2011.

[29] D. Du, "Biogeography-based optimization: Synergies with evolutionary strategies, immigration refusal and Kalman filters," Master's thesis, Cleveland State University, 2009.

[30] H. Mo and L. Xu, "Biogeography migration algorithm for traveling salesman problem," *Advances in Swarm Intelligence*, pp. 405–414, 2010.

[31] Y. Song, M. Liu, and Z. Wang, "Biogeography-based optimization for the traveling salesman problems," in *2010 Third International Joint Conference on Computational Science and Optimization*. IEEE, 2010, pp. 295–299.

[32] C. Darwin, "On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life," *New York: D. Appleton*, 1859.

[33] R. H. MacArthur and E. O. Wilson, *The Theory of Island Biogeography*. Princeton University Press, 1967.

[34] R. Rarick, D. Simon, F. Villaseca, and B. Vyakaranam, "Biogeography-based optimization and the solution of the power flow problem," in *IEEE International Conference on Systems, Man and Cybernetics, 2009*. IEEE, 2009, pp. 1003–1008.

[35] G. Lozovyy, P.; Thomas and D. Simon, "Biogeography-based optimization for robot controller tuning," in *Computational Modeling and Simulation of Intellect: Current State and Future Perspectives*. IGI Global, 2011.

[36] H. Kundra and M. Sood, "Cross-Country Path Finding using Hybrid approach of PSO and BBO," *International Journal*, vol. 7, 2010.

[37] A. Bhattacharya and P. Chattopadhyay, "Biogeography-based optimization for different economic load dispatch problems," *IEEE Transactions on Power Systems*, vol. 25, no. 2, pp. 1064–1077, 2010.

[38] V. Panchal, S. Goel, and M. Bhatnagar, "Biogeography based land cover feature extraction," in *World Congress on Nature & Biologically Inspired Computing*. IEEE, 2010, pp. 1588–1591.

[39] M. Ergezer, B. E. Abali, and D. Simon, "Biogeography-based optimization identifies material coefficients as an inverse problem," January 2011, poster session presented at NSF CMMI Research and Innovation Conference, Atlanta, GA.

[40] G. Tao and Z. Michalewicz, "Inver-over operator for the TSP," in *Parallel Problem Solving from NaturePPSN V*. Springer, 1998, p. 803.

[41] D. Du and D. Simon, "Biogeography-based optimization for the traveling salesman problem," in *Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Submitted.

[42] J. Kruskal Jr, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical society*, vol. 7, no. 1, pp. 48–50, 1956.

[43] N. Christofides, A. Mingozzi, and P. Toth, "The vehicle routing problem," *Revue Française d'Informatique et de Recherche Opérationnelle*, vol. 10, no. 2, pp. 55–70, 1976.

[44] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 3, pp. 345–358, 1992.

[45] P. Toth and D. Vigo, *The vehicle routing problem*. Society for Industrial Mathematics, 2002.

[46] V. Magirou and A. Nicolitsas, "The efficient drilling of printed circuit boards," *Interfaces*, vol. 16, no. 4, pp. 13–23, 1986.

[47] G. Onwubolu and M. Clerc, "Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization," *International Journal of Production Research*, vol. 42, no. 3, pp. 473–491, 2004.

[48] G. Reinelt, "TSPLIB–A traveling salesman problem library," *INFORMS Journal on Computing*, vol. 3, no. 4, p. 376, 1991.

[49] F. Leighton, "A graph coloring algorithm for large scheduling problems," *Journal of Research of the National Bureau of Standards*, vol. 84, no. 6, pp. 489–503, 1979.

[50] G. Chaitin, M. Auslander, A. Chandra, J. Cocke, M. Hopkins, and P. Markstein, "Register allocation via coloring," *Computer Languages*, vol. 6, no. 1, pp. 47–57, 1981.

[51] S. Even, O. Goldreich, S. Moran, and P. Tong, "On the NP-completeness of certain network testing problems," *Networks*, vol. 14, no. 1, pp. 1–24, 1984.

[52] M. Carter, "A survey of practical applications of examination timetabling algorithms," *Operations Research*, vol. 34, no. 2, pp. 193–202, 1986.

[53] D. Johnson, C. Aragon, L. McGeoch, and C. Schevon, "Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning," *Operations research*, pp. 378–406, 1991.

[54] P. Galinier and J. Hao, "Hybrid evolutionary algorithms for graph coloring," *Journal of Combinatorial Optimization*, vol. 3, no. 4, pp. 379–397, 1999.

[55] D. Costa and A. Hertz, "Ants can colour graphs," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 295–305, 1997.

[56] C. Fleurent and J. Ferland, "Genetic and hybrid algorithms for graph coloring," *Annals of Operations Research*, vol. 63, no. 3, pp. 437–461, 1996.

[57] M. Trick. (2010, August) Graph coloring instances. [Online]. Available: http://mat.gsia.cmu.edu/COLOR/instances.html

[58] F. Leighton, "A graph coloring algorithm for large scheduling problems," *Journal of Research of the National Bureau of Standards*, vol. 84, no. 6, pp. 489–503, 1979.

[59] D. Knuth, *The Stanford GraphBase: a platform for combinatorial computing*. ACM Press, 1993.