Journal of AI and Data Mining
Vol 8, No 3, 2020, 313-329.
DOI: 10.22044/JADM.2020.7835.1922

Solving Traveling Salesman Problem based on Biogeography-based Optimization and Edge Assembly Cross-over

A. Salehi and B. Masoumi*

Faculty of Computer and information Technology, Islamic Azad University, Qazvin Branch, Qazvin, Iran.

Received 27 December 2018; Revised 23 December 2019; Accepted 08 February 2020

**Corresponding author: Masoumi@qiau.ac.ir (B. Masoumi).*

Abstract

The Biogeography-Based Optimization (BBO) algorithm has recently been of great interest to the researchers for its simplicity of implementation, efficiency, and low number of parameters. The BBO algorithm in optimization problems is one of the new algorithms that have been developed based on the biogeography concept. This algorithm uses the idea of animal migration to find suitable habitats for solving the optimization problems. The BBO algorithm has three principal operators called migration, mutation, and elite selection. The migration operator plays a very important role in sharing information among the candidate habitats. The original BBO algorithm, due to its poor exploration and exploitation, sometimes does not perform desirable results. On the other hand, the Edge Assembly Cross-over (EAX) has been one of the high powers cross-overs for acquiring off-spring, and it increases the diversity of the population. A combination of the BBO algorithm and EAX can provide a high efficiency in solving the optimization problems including the traveling salesman problem (TSP). In this paper, we propose a combination of those approaches to solve the traveling salesman problem. The new hybrid approach is examined with standard datasets for TSP in TSPLIB. In the experiments, the performance of the proposed approach is better than the original BBO and four others widely used metaheuristics algorithms.

Keywords: *Biogeography-Based Optimization, Evolutionary Algorithms, Edge Assembly Cross-over, Genetic Algorithm, Traveling Salesman Problem.*

1. Introduction

In the recent years, many meta-heuristics algorithms have been proposed to solve complex optimization problems. Meanwhile, nature-inspired algorithms have shown a good performance in solving the optimization problems. The biogeography-based optimization (BBO) algorithm is a population-based meta-heuristics algorithm that was developed by Simon in 2008 [1]. The main idea behind this algorithm is inspired by the natural migration of the species between habitats. A migration operator leads to the sharing of information between habitats, where each habitat is a candidate solution of the problem.

The BBO algorithm is similar to the evolutionary algorithms like particle swarm optimization (PSO) in the sense that each candidate solution in the search space moves toward better solutions in each iteration. In the BBO algorithm, unlike the genetic algorithm (GA), the poor solution will not be eliminated at the iteration, and in the next iteration, new solutions will be created. The major problem of the BBO algorithm is that it may be trapped in the local optima of the objective function. Many studies have been conducted to improve the performance of the original BBO algorithm, and most of these studies, which will be discussed below, have focused on improving the migration operator of the BBO algorithm.

One of the known issues in the field of optimization problem is the traveling salesman problem (TSP), which is to find the shortest path to the optima.

Finding the shortest route among *n* different cities is the main issue of TSP, and the distance between the cities is given as a matrix as an input. This problem is NP-hard, and thus has attracted the attention of many researchers [2]. So far, many algorithms have been proposed to solve this problem. In general, these algorithms can be divided into two categories. One is the exact algorithms such as dynamic programming, branch-and-bound, and linear programming that can get the optimal solution but the problem is that their execution time expands exponentially according to the dimensions of the problem. The latter are the approximate and heuristics algorithms including constructive heuristics (like nearest neighbor, match twice, and stitch), iterative improvement, randomized improvement (like GA, Simulated Annealing (SA), Tabu search, ant colony optimization, river formation dynamics, PSO, etc., that can quickly provide good solutions [3]. In the recent years, many evolutionary algorithms (EAs) have been developed to solve TSP. The main objective of these algorithms has been to provide an appropriate cross-over operator for TSP since the performance of EAs depends heavily on the cross-over operator. In other words, the design of a good cross-over operator can increase the efficiency of the algorithm in solving problems. The Edge Assembly Cross-over (EAX) operator, first introduced by Nagata *et al*. has been one of the high-power cross-over operators to acquire off-spring [4].

In this paper, a novel hybrid BBO with EAX named as BBOEAX is proposed for solving TSP. The main idea of this research work is to improve the solutions obtained by the BBO algorithm using the EAX operator and causing diversity in the population. In this case, the solutions generated by the BBO algorithm have been given as parent tours to the EAX operator, which generated new off-springs as solutions. This procedure caused a variety of algorithm solutions because a random selection of parent tours caused the candidate's solutions to be highly diverse. The BBO search algorithm has been found to be faster and more reliable, and has generally resulted in a better optimization than the other algorithms for the considered problems. The BBO algorithm has also received much attention in the recent years for its good capabilities. The performance of the proposed approach compared with the original BBO algorithm and four others widely used metaheuristics algorithms including GA [5, 6], PSO [7], SA, and Differential Evolution (DE).

The rest of this paper has been organized as follows. Section 2 discusses the related work. In Section 3, the original BBO algorithm has been explained. Section 4 describes EAX in detail. In Section 5, the proposed BBOEAX algorithm has been described in detail. In Section 6, the performance of BBOEAX has been assessed on standard benchmark functions and the famous TSP. Finally, in Section 7, the paper has been concluded.

2. Related works

Many attempts to improve the performance of the BBO algorithm have been made by the researchers after the algorithm was introduced. In order to overcome the weaknesses of the original BBO algorithm, many ideas have been proposed that emphasizes the improvement of the performance of the migration operator. Zhang et al. have presented a novel hybrid algorithm based on the BBO algorithm and the Grey Wolf Optimizer (GWO), named as the HBBOG algorithm. In the proposed algorithm, both the BBO and GWO algorithms have been improved [8]. Here, in order to improve the BBO algorithm, the original migration operator was replaced with a multi-migration, and instead of the traditional mutation operator, the differential mutation operator was used to enhance the performance of the algorithm. The multi-migration operator uses two different migration operators according to the random-dimensional migration based on random selection. Yang et al. have presented an improved BBO algorithm that uses the non-linear migration operator, and they have applied the proposed algorithm in path planning for the mobile robot [9]. In the presented algorithm, a non-linear migration operator was used instead of the traditional migration operator. The main idea of the presented algorithm was the dynamic change of the migration rate based on the quantity. Bansal has presented a modified blended migration and polynomial mutation, named as BBO-MBLX-PM [10]. The modified migration operator used a blended cross-over for migration that was described as:

$$Hi(SIV) \leftarrow \alpha Hi(SIV) + (1-\alpha) Hj(SIV) \quad (1)$$

Here, α affects the performance of the migration operator and can be determined by the habitat fitness or randomly. Also, the mutation operator was replaced with the polynomial mutation. In order to make the search process more diverse and to find solutions more accurately with high convergence rates, the modified BBO introduced by Farswan *et al*. was named as MBBO [11]. Their immigrating habitat accepted SIVs from

immigrating habitat, best habitat, and random habitat instead of accepting SIVs only from emigrating habitat. In order to enhance the performance of the original BBO algorithm, Al-Roomi *et al*. have introduced a novel hybrid BBO with SA called MpBBO [12]. In their method, the principal condition of selecting the immigrating habitat is based on the metropolis criterion rules. Chen *et al*. have developed the covariance matrix-based migration (CMM) to decrease the dependence of the basic BBO algorithm to the pre-defined system coordinates, which is one of the main disadvantages of the BBO algorithm [13].

Fan *et al*. have developed a discrete BBO algorithm for detecting the overlapping community detection to improve the efficiency of the algorithm [14]. They used an affinity degree for mutation operator and designed problem-specific rules for migration operator. Khishe *et al*. have introduced a novel exponential-logarithmic migration operator for the original BBO algorithm, named as the ELBBO algorithm [15]. The main idea of their algorithm is that most undesirable habitats have an ascending logarithmic emigration rate and descending exponential immigration rates. On the other hand, the rich habitats have an exponential emigration rate and logarithmic immigration rate. Paraskevopoulos *et al*. have proposed a modified BBO, named as the real-coded biogeography-based optimization (RCBBO), which combines fuzzy decision-making for the cognitive radio engine design used in the internet of things (IoT) [16]. In order to improve the population diversity, the authors employed Gaussian, Cauchy, and Levy mutations as the mutation operator. Feng *et al*. have proposed a modified BBO algorithm named as PRBBO for solving the global optimization problems [17]. In the modified algorithm, the triple combination is used, which includes migration operator combined with random ring topology, a modified mutation operator, and a self-adaptive Pow-ell's method. In their method, the local ring topology is used instead of global topology to increase the population diversity. Lohokare *et al*. have introduced a novel version of the BBO algorithm based on the memetic behavior, called ABBOMDE, to improve mutation, and clearing of duplicate operators is used to speed-up the efficiency of the BBO algorithm [18].

EAX has been used for solving a TSP due to its high performance as well as the other similar issues in many evolutionary algorithms. Nagata *et al*. have applied EAX for solving TSP [19]. First, the advantages of the EAX operator were considered against the other cross-over operators, and then the advantages of the EAX operator in TSP with other candidate operators were examined. Nagata has developed a novel approach based on EAX for solving the capacitated vehicle routing problem [20]. Haque *et al*. have used the GA algorithm, memetic algorithm, and EAX to improve the solution of TSP [21]. Blocho *et al*., to reduce the number of routes and minimize them in the VRP issue with time windows, have developed a parallel algorithm based on EAX [22]. Their algorithm used the EAX when exchanging the best solutions between processes. For increasing the diversity of population in TSP, a fast EAX algorithm was introduced [23]. The main idea in the introduced algorithm is to localize the EAX operator by changing the edges, which leads to the local execution of the EAX algorithm.

In the recent decades, many studies have been conducted to provide an efficient algorithm for solving TSP. Based on this, the local search strategies are designed to solve the problem of TSP in order to find the optimal solutions. For example, Kocer *et al*. have used an improved artificial bee colony (ABC) algorithm and a local search algorithm for solving TSP [24]. Also, a lot of research works have been conducted to solve TSP based on heuristic algorithms and evolutionary algorithms. Osaba *et al*. have used an improved discrete BAT algorithm for solving TSP [25]. Cheng *et al*. have introduced a hybrid artificial fish algorithm to solve TSP [26]. In the presented algorithm, the genetic cross-over operator was used to improve the efficiency of the artificial fish algorithm. Ardalan *et al*. have developed a novel Imperialist Competitive Algorithm (ICA) for TSP [27]. In the presented algorithm, all cities were divided into several groups and each group was visited only once. Liao *et al*. have used an improved version of PSO for solving TSP with an evolutionary algorithm [28]. The presented algorithm consists of two phases to solve the TSP efficiently. The first phase consists of fuzzy clustering, and in the second phase, the PSO algorithm is executed based on the genetic-based algorithm.

3. Biogeography-Based Optimization (BBO)

In this section, we describe the original BBO algorithm and summarize its main operators, migration operator, and mutation operator. BBO algorithm is a new intelligence-based metaheuristic algorithm inspired by nature based on the concept of animal migration to find the habitat. In this algorithm, each habitat represents a candidate

solution of the problem. The migration of species from one habitat to another is based on the suitability index called the habitat suitability index (HSI). HSI actually represents the individual's fitness. The parameters such as rainfall, temperature, region, and humidity affect the excellent characteristics of biological habitats. In the BBO algorithm, from the Simon's viewpoint, these characteristics are called the suitability index variables (SIVs). Simply an n-dimensional habitat, which is a candidate solution of the problem, is formed by n SIVs whose HSI denotes its fitness. In the BBO algorithm, each habitat has an immigration rate, which means that it has a desire to accept poor habitats, and emigration rate, which means that there is a strong tendency to migrate to rich habitat. In BBO algorithm, in each solution, the solution features within habitats based on the immigration and emigration are improved. High-HSI habitats share their good features with low-HSI habitats, and low-HSI habitats accept the new features of high-HSI habitats. Based on the BBO algorithm approach, the sharing of features in good solutions to other solutions has a high probability, and poor solutions have high probabilities to accept SIVs from other solutions. The emigration rate of the habitat with a high HSI decreases to a habitat with a low HSI so the habitat with the highest HSI has the maximum emigration rate. The immigration rate of the habitat with a high HSI increases to a habitat with a low HSI, and therefore, the habitat with the highest HSI has the maximum immigration rate. The immigration rate λ and the emigration μ are calculated according to the following two formulas [1]:

$$\lambda_i = I\left(1-\frac{k_i}{n}\right) \quad (2)$$

$$\mu_i = E\left(\frac{k_i}{n}\right) \quad (3)$$

where the immigration rate of the i^{th} habitat is determined by λ_i, and μ_i is the emigration rate for the i^{th} habitat. The maximum immigration rate and the maximum emigration rate are denoted with I and E, respectively, and N determines the size of the population; k_i stands for the fitness rank of the i^{th} habitat after sorting fitness of the i^{th} habitat so that the worst solution has k_i of 1 and the best solution has k_i of N.

Migration and mutation are two principle operators in the BBO algorithm. The migration operator is responsible for generating a new solution in each iteration, and is similar to the cross-over operator in the evolutionary algorithm. The mutation operator is randomly assigned to habitats, and is responsible for preserving the diversity of habitats and preventing the trapping of the algorithm in the local optimal.

Changing the number of species from t to $(t+\Delta t)$ in the habitat with s species is equal to the P_s probability, as follows:

$$P_s(t+\Delta t) = P_s(t)(1-\lambda_s\Delta t-\mu_s\Delta t)+P_{s-1}\lambda_{s-1}\Delta t+P_{s+1}\mu_{s+1}\Delta t \quad (4)$$

The immigration rate of a habitat with s species is determined by $\lambda_s.\mu_s$ is the emigration rate when there are s species in the habitat [14].

For a habitat with s species at time t + Δt, one of the following conditions should hold:

1. When in the time t there are s species in the habitat, then in the times t and $t+\Delta t$, there are no immigration and emigration among the species.

2. When in the time t there are $(s-1)$ species in the habitat, then in the times t and $t+\Delta t$, one species is immigrated.

3. When in the time t there are $(s+1)$ species in the habitat, then in the times t and $t+\Delta t$ one species is emigrated.

For simplicity, we ignore the probability of immigration or emigration more than one species, and consider the Δ value 0 $(\Delta t \to 0)$.

$$\dot{P}_S = \begin{cases} -(\lambda_S+\mu_S)P_S+\mu_{S+1}P_{S+1} & S=0 \\ -(\lambda_S+\mu_S)P_S+\lambda_{S-1}P_{S-1}+\mu_{S+1}P_{S+1} & 1\le s\le S_{max}-1 \\ -(\lambda_S+\mu_S)P_S+\lambda_{S-1}P_{S-1} & S=S_{max} \end{cases} \quad (5)$$

Therefore, solutions with very high HSI as well as low HSI solutions are rarely possible. Solutions with a medium HSI are more possible. The mutation rate m_i is as follows, where m_{max} denotes a user-defined parameter, the existence probability

is determined by p_i, and $p_{max} = max\{p_i, i = 1,2,\ldots, p_s\}$.

$$m_i = m_{max}\left(1 - \frac{P_i}{P_{max}}\right) \quad (6)$$

In each solution, the mutation operator is randomly generated, and is based on the probabilistically of replacing SIV.

Algorithm 1 shows the pseudo-code of migration operator, and the i^{th} habitat is denoted by H_i and the j^{th} habitat is denoted by H_j. Also, n is the number of habitats and d is the dimension of a solution.

Algorithm 1. Pseudo-code of migration operator in the original BBO.

```
initialize n;                          // Population Size
for i: = 1 to n do
   Select H_i According to λ_i; from Eq. (2)
    if H_i is selected, then
      if rand (0,1) < λ_i,  then
        for j: = 1 to d do   // d is Dimension
      Select Habitat H_j according to μ_i; from Eq.(3)
      if  H_j is selected, if rand (0,1) < μ_i, then
     replace SIV H_i with Selected SIV from H_j;
 end if
        end for
      end if
    end if
  end for
```

The mutation operator during the BBO process changes the habitat SIVs randomly to determine the diversity of populations based on the probability of each habitat. It is also the duty of the mutation operator to avoid getting caught in the local optima. The pseudo-code of the migration operator is shown in Algorithm 2. Also, Algorithm 3 describes the pseudo-code of the basic BBO.

Algorithm 2. Pseudo-code of mutation operator in the original BBO.

```
Population size = n;
for i = 1 to Population do
  Select Habitat H_i According to P_i; from Eq.(6)
    if H_i is Selected, if rand (0,1) < m_i, then
        H_i(SIV) ← randomly generated SIV;
    end if
end for
```

Finally, after applying the migration and mutation operator to the relevant habitat, the comparison between the habitat individual and the original individual is done. In the next assessment, if the obtained HSI value of the individual is higher than the original value, the original HSI value is well-replaced by the new value obtained; otherwise, there is no change in the original HSI value. By repeating continuous assessments and applying operators, the algorithm obtains better individuals and directs poor habitats to better solutions. This process directs the algorithm to find optimal solutions.

Algorithm 3. Pseudo-code of the original BBO algorithm.

```
Randomly initialize a Population of n Habitats  H_i ,
i = 1,…..,n;
 Initialize Max Iteration;
while (Termination Criteria) do
  Calculate Fitness (HSI) for each Habitat and
      sort Habitats according their HSI;
   for i = 1 to n do  // n is Population size;
     Calculate λ and μ for each Habitat according to HSI;
     from (2, 3)
   end for
   /*Migration

    Select H_i with Probability according to λ_i;  from
    Eq.(2)
   if  H_i is selected, then
     select H_j with Probability according to μ_i ; from
      Eq(3)
       if H_j is selected then; from Eq.(5)
           Randomly Select SIV from H_j;
            Replace SIV in H_i with one from H_j ;
        end if
     end if
/* Mutation
   Select H_i with Probability according to the
       Mutation rate; from Eq. (6)
   if H_i(SIV) is selected, then from Eq. (5)
         Perform Mutation;
    end if

  Evaluate the Fitness values of the Habitats;
  Perform Elitism and Update the Best Solution;
end while
return Best Solution;
```

4. Edge Assembly Cross-over (EAX)

In this section, EAX is explained in detail. Most of the evolutionary algorithms use the cross-over operator because it is an operator that generates better solutions to the problem [4, 18]. In other words, the cross-over operator, by combining the parental responses, generates new off-spring in the next iteration, where the new generation inherits features from their parents. If this is done continuously, then the generations will be produced, and there is the possibility of getting

better solutions. The cross-over operator originally belongs to GA, which is used by other evolutionary algorithms too. In the recent years, different models have been developed for the cross-over operator but the edge assembly cross-over has one of the high performances based on the two advantages: (i) a wide variety of children can be generated from a pair of parents because intermediate solutions are constructed under the relaxed condition of TSP, and (ii) children can be constructed without introducing long edges, cross-over with very high efficiency compared with them EAX generates new solutions (off-spring) by combining two solutions (parents) in a population. Off-springs generated by EAX were totally composed of edges of parents. The aim of EAX is to inherit as many edges as possible from parent to child. When two parents are selected for cross-over, EAX combines them into an individual solution. The two parents are determined by A and B, respectively. The stages of the EAX operator are as follow:

1. Selecting two parents for cross-over, called tour-*A* and tour-*B*, respectively. G_{AB} is defined as a graph constructed by merging tour-*A* and tour-*B*.

2. Extracting *AB-cycles* from G_{AB} graph. *AB-cycles* are closed loops in G_{AB}, which are derived from the alternate tracing of the edges of Tour-A and Tour-B.

3. Creating *E-set* with an *AB-cycles* based on a specific rule.

4. Applying *E-set* to tour-*A* to construct an intermediate solution. For example, the edges of tour-*A* can be removed in the *E-set* from tour-*A* and replaced by the edges of tour-*B* in the *E-set*.

5. Intermediate solution should be modified to create a valid tour by merging sub-tours together.

In the first step, two parents are selected for cross-over, and the G_{AB} graph is made. Then *AB-cycles* are extracted from the G_{AB} graph. When the graph is undirected, the edge between two nodes is the same, and *AB-cycles* of them are ineffective. In other words, these types of cycles consist of only two edges that are ineffective, and should be eliminated. In the next step, according to the rules, which will be described later, an *E-set* is constructed by selecting different combinations of *AB-cycles*. Next, by applying the *E-set* to the tour-*A*, intermediate solution, which includes several sub-tours, is made. In the final stage, the sub-tours are combined in an innovative way and a valid tour is constructed. In order to connect two sub-tours, one edge is removed from the first sub-tour, and one edge from the second sub-tour, and the two new edges are added to connect them. Selecting the edges to remove, as well as selecting the sub-tours to connect to each other, should be intelligently determined. Figure 1 shows the steps of EAX.

EAX Strategies

There are different approaches to construct *E-set*, and hence, each *E-set* can be constructed from any combination of *AB-cycle*s. Here are some strategies to construct an *E-set*.

EAX-1AB: In this strategy, a single *AB-cycle* constructs a one *E-set*. Hence, each *AB-cycle* alone is an *E-set* and leads to the increase in the number of *E-sets*. This method makes intermediate solutions that tend to be similar to tour-*A*.

EAX-Rand: Random selection of *AB-cycles* constructs these types of *E-sets*. This method makes intermediate solutions that tend to use the edges of both tour-*A* and tour-*B*.

EAX-Block: First of all, a one *AB-cycle* that is large enough to be selected randomly is called center *AB-cycle*. Then the small *AB-cycles* that the incident to the center *AB-cycle* are added to the *E-set*.

Random selection of *AB-cycles* to construct *E-sets* is simple and can generate a wide variety of off-springs. Selecting single *AB-cycles* is also simple and leads to use both parents. The most important disadvantage of these two methods is that they may be caught up in the local optimum. Since the *AB-cycle* of EAX-Block method is large enough, it is more efficient than the two methods above. A more detailed description of these methods can be found in [20-23].

EAX-Block algorithm steps:

1. Selecting a one *AB-cycle* with large size that will be considered as the center *AB-cycle*.

2. Center *AB-cycle* applied to tour-*A*. Constructing an intermediate solution, $u_i =$ (i= 1,2,…,m), i is the i^{th} sub-tour and m is the number of sub-tour.
3. Selecting an *AB-cycle* that satisfies the following criteria.

- ✓ First condition: Connect with the vertices in U_i.

- ✓ Second condition: Their size is smaller than the center *AB-cycle*.

4. Creating *E-set* from center *AB-cycle* and selecting *AB-cycles* in step 3.

The more advantageous EAX-Block than simple EAX is that assembling a block of tour-*A* and a block of tour-*B* leads to construct medium solutions. If the size of the *AB-cycle* selected is small, the number of vertices connected to tour-*A* (tour-*B*) tends to be smaller. In this case, EAX-block gains the most ideal feature and can satisfy the above conditions. Given what was described about EAX-Block, the particular conditions of this method were applied to the proposed algorithm. Comparing the three strategies mentioned above shows that EAX-Block works faster than the optimal response, which causes the speed of the algorithm to rise. Table 1 shows the comparison of the three strategies above.

Table 1. Comparison of the strategies of constructing *E-set*.

Dataset / Strategies	Eli51	Berlin52	Kroa100	Qatar194
EAX-1AB	498	9234	23176	10126
EAX-Rand	467	7981	22895	9954
EAX-Block	431	7551	22234	9735

5. Proposed BBOEAX Algorithm

In this section, a hybrid algorithm based on BBO and EAX, called BBOEAX, for solving TSP is proposed. In the proposed algorithm, the migration operator had an important role in the performance of the BBO algorithm because it was responsible for sharing information between habitats. The migration operator generated new solutions in each iteration to the problem that better solutions were replaced by the current solution. Since the EAX operator has been one of the best cross-over operators to produce new off-spring, it could help the BBO's migration operator to generate new solutions.

The main idea of this combination was that since the EAX operator selected the parent's edges randomly, the habitats had a variety of SIVs, which was one of the great weaknesses of the BBO algorithm. In other words, the original BBO algorithm, due to the complete replacement of rich habitat indices with poor habitats, after repeated repetition of the algorithm's execution, all habitats had almost identical and repetitive SIVs that would no longer be able to make better solutions. The EAX operator, by randomly selecting the edges and increasing diversity in the population, partially solves this problem in the BBO algorithm. Choosing two habitats as the parent for the EAX operator and changing their variables based on the operator's rules created new off-springs that were highly diverse. The same variation in habitat indices made the algorithm achieves better solutions.

In the proposed hybrid BBOEAX algorithm, when a poor habitat was selected based on the λ parameter, the migration operator replaced its SIV with the SIV of rich habitats. Then this poor habitat was selected along with another random habitat as the EAX operator's parents. After these two steps, the mutation operator was used to prevent the trapping of the algorithm in the local optimal. Figure 2 shows a flowchart of the BBOEAX. Also, the pseudo-code of the proposed BBOEAX algorithm has been shown in Algorithm 4.

Algorithm 4. Pseudo-code of the proposed BBOEAX algorithm.

Randomly initialize a Population of n Habitats H_i, i = 1,…..,n;
while the Termination Criteria are not Satisfied, **do**
Calculate Fitness (HSI) for each Habitat and sort Habitats according their HSI;

for i = 1 to n **do**, where n is Population size;
  Calculate λ and μ for each Habitat Based on HSI; from Eq. (2, 3)
end for
/***Migration**
Select H_i with Probability based on λ_i; from Eq. (2)
if H_i is selected, **then**
  select H_j with Probability based on μ_i; from Eq.(3)
    if H_j is selected **then** from Eq. (5)
      Randomly Select SIV from H_j;
      Replace SIV in H_i with one from H_j;
  end if
end if
/* EAX
  Select two habitats as a parent based on best strategy
  Execute EAX procedure
  Add the valid tours to the population
/* **Mutation**
Select H_i with Probability based on the Mutation rate; from (6)
if $H_i(SIV)$ is selected, **then**
    Perform Mutation; from Eq. (5)
end if
  Evaluate the Fitness values of the Habitats;
Perform Elitism and Update the Best Solution;
end while
return Best Solution;

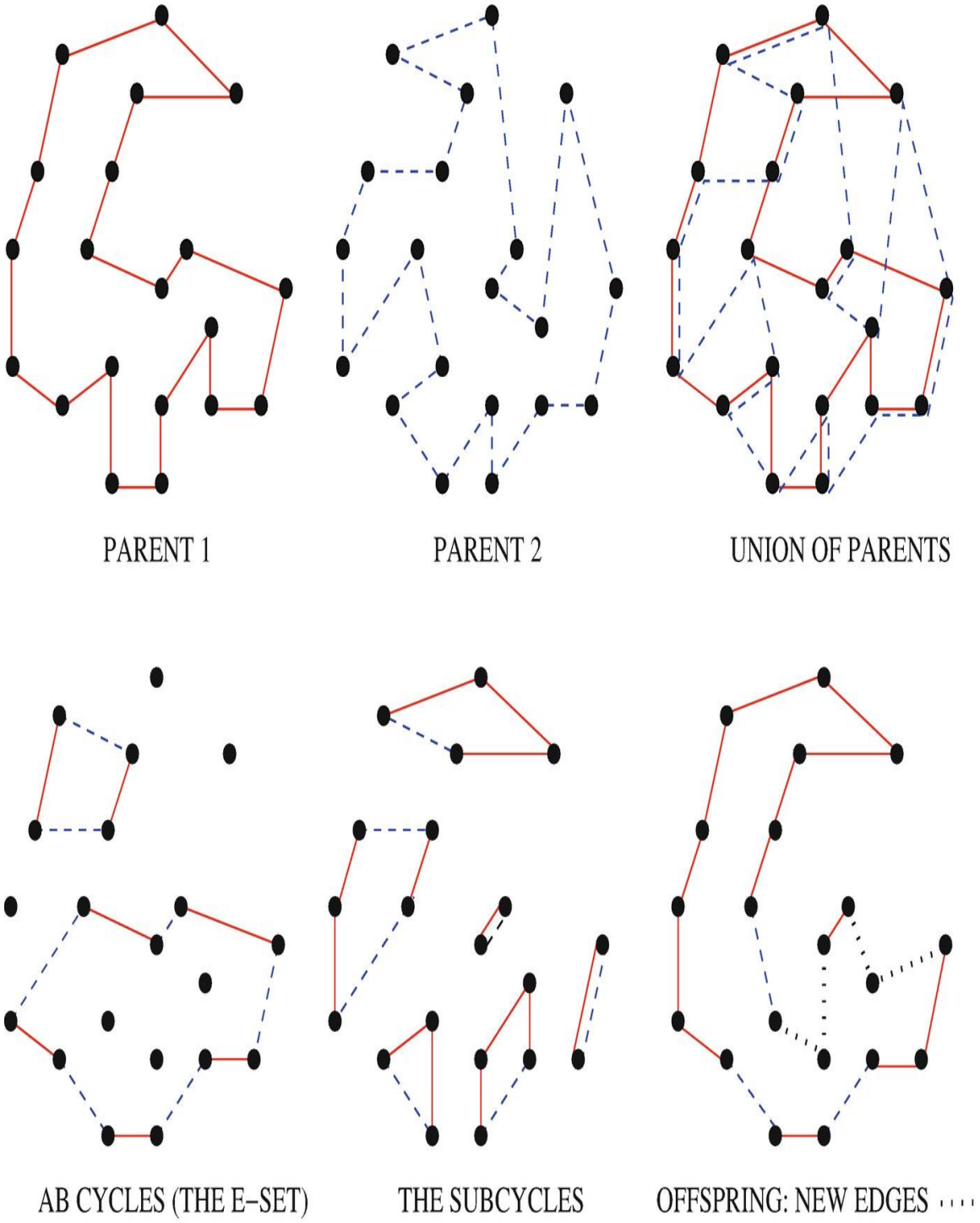

Figure 1. Stages of producing off-spring by combining parent edges; Figure use from [23].

6. Experimental results

In this section, we compare the proposed hybrid BBOEAX algorithm with the basic BBO algorithm and with different evolutionary algorithms such as PSO, GA, SA, and DE. In order to evaluate the performance of the proposed BBOEAX algorithm on the standard benchmark functions and TSP, several experiments have been conducted whose results have been reported below. In the experiments, various benchmark functions and datasets of TSPLib have been used [29].

In all experiments, the size of the population was 200 and the maximum iterations of the in algorithm was considered to be 500. In addition, for each experiment, all algorithms were executed 30 times independently for each problem, and the results of the tests were presented based on the best, worst, mean, and standard deviations. Also, all

implementations were done by the MATLAB software (version R2016).

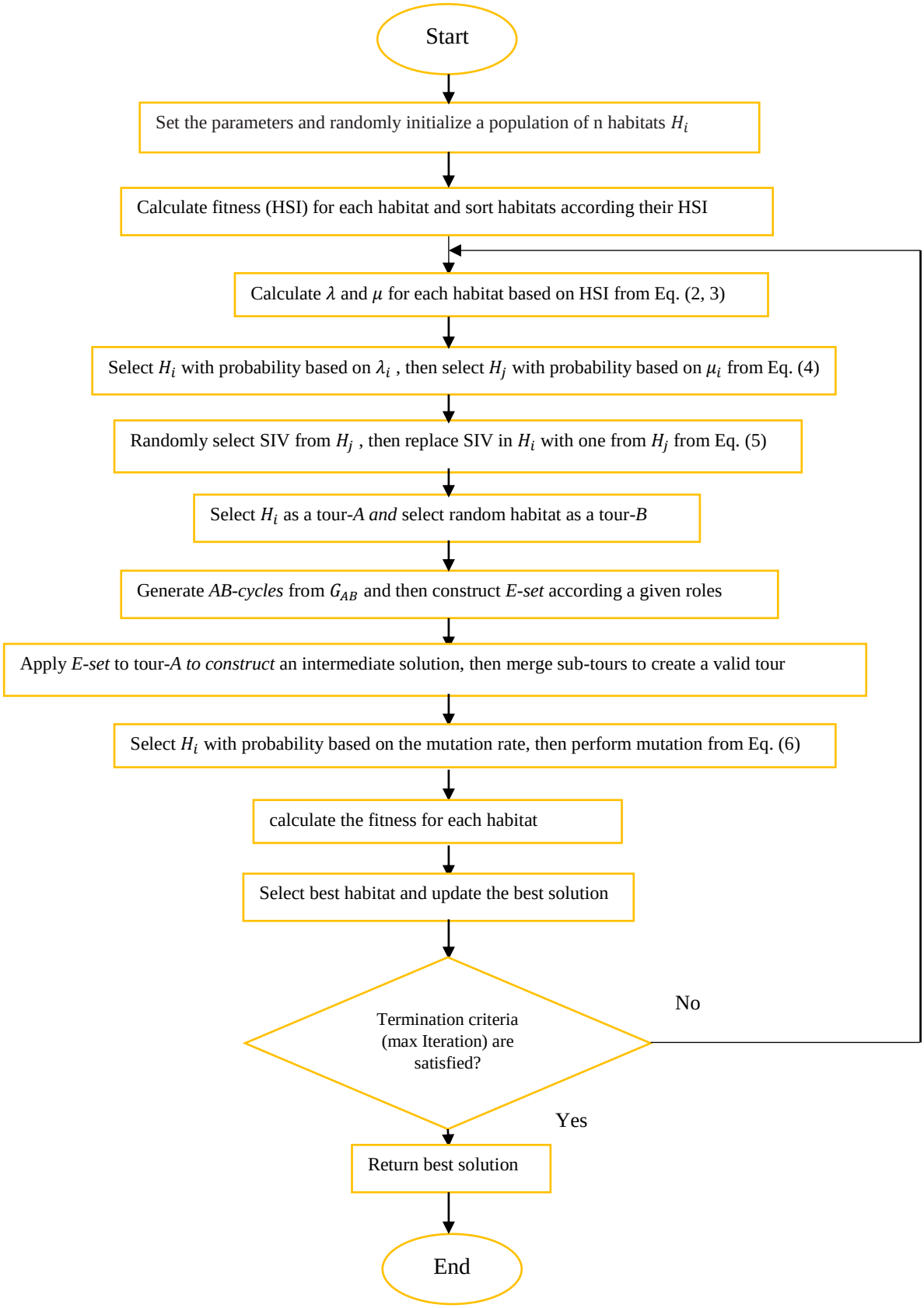

Figure 2. Main process of BBOEAX for solving alpha problem using offspring from parent edges combination.

In the previous study, the parameter settings, like population size impact, and mutation probability impact of the BBO algorithm in details were considered [30]. Also, other algorithm parameter settings were considered. The parameters of the GA and SA algorithms were based on [31]. The parameters of the different evolution algorithms were determined by [32]. In addition, the PSO algorithm parameters were considered based on [33]. The parameter settings of all algorithms recommended by authors were considered and summarized in table 2. The experiments were performed on 12 standard test functions. Also, the experiments were performed over 50 times running algorithm on any benchmark functions. The list of standard test functions used in the experiments is represented in table 3. Additional explanations about these test functions can be found in [13, 34]. TSPLIB, since its release, has included a collection of different types of traveling salesman problem instance with different sizes; it is used to compare the results of the newly proposed algorithms. Table 4 shows the datasets used in this paper. The other TSP instances could be found in [29, 35].

Experiment 1: This experiment is conducted to compare the proposed algorithm with several evolutionary algorithms like BBO [30], DE [32], GA [5], PSO [7], and SA [5] on standard functions with many local minima. These functions reflect the algorithm's ability to escape from the poor local optima and locate a good near-global optimum [36].
In order to evaluate the efficiency of the algorithms, the parameter Success Rate (SR) is used. The success rate of the algorithm is to obtain the minimum required result above the specified accuracy value before the termination condition expires [36, 37].

Table 5 presents the values for SR, mean value, and standard deviation for all algorithms. From the results, the following points can be made:

1. The results show that the proposed algorithm has a better performance than the BBO, GA, and PSO algorithms.

2. In some functions (like $f_1, f_9, f_{10,}$ and f_{12}) the proposed algorithm in terms of SR with DE and SA algorithms is equivalent and, in some
functions, the proposed algorithm is ranked the third.

3. In function f_5, only BBOEAX has a SR, and no algorithm has SR.

Experiment 2: This experiment is conducted to compare the proposed algorithm with several evolutionary algorithms on TSP instances with less than 100 cities and more than 100 cities based on TSPLIB. Tables 6 and 7 present the results of comparing the proposed BBOEAX algorithm with the original BBO algorithm and other evolutionary algorithms based on the best, mean, worst, and standard deviation. The results showed that the proposed BBOEAX achieved better results than the other algorithms and performed better. On the other hand, by examining the results shown in table 6 and 7, it was clear that the BBOEAX algorithm could get a better optimal result than the others. Also figures 3 and 4 show the results of the simulations for two datasets. Also, in all cases, the proposed algorithm had the lowest standard deviation than the other algorithms.

Table 2. Parameter settings of all algorithms.

Algorithm	*Itr*	*Pop*	*E*	*I*	*MP*	*Pc*	*w*	C_1	C_2	W_r	*T*	*T0*	*α*	*q0*	*F*	*CR*	*NP*	*SR*
BBOEAX	500	100	1	1	0.5	---	---	---	---	---	---	---	---	---	---	---	---	---
GA	500	100	---	---	0.5	0.2	---	---	---	---	---	---	---	---	---	---	---	---
PSO	500	100	---	---	---	---	1.5	1.5	1.5	0.9	---	---	---	---	---	---	---	---
SA	500	100	---	---	---	0.99	---	0.8	---	---	10000	1	0.9	---	---	---	---	---
DE	500	100	---	---	---	---		---	---	---	---	---	---	0.9	0.9	0.9	5	20%

Table 3. Standard functions used in experiments.

	Function	D	Range space	Global minimum
f_1	Sphere	d	[-5.12, +5.12]	0
f_2	Ackley	d	[-32.768, +32.768]	0
f_3	Griewank	d	[-600, +600]	0
f_4	Rastrigin	d	[-5.12, +5.12]	0
f_5	Zakharov	d	[-5, +10]	0
f_6	Rosenbrock	d	[-2.048, +2.048]	0
f_7	Michalewicz	d	$[0, \pi]$	at d = 2: -1.8013 at d = 5: -4.687658 at d = 10: -9.66015
f_8	Langermann	2	[0, 10]	at d = 2: -.1621259 at d = 5: -1.4
f_9	Levy	d	[-10, +10]	0
f_{10}	Rotated Hyper Ellipsoid	d	[-65.536, +65.536]	0
f_{11}	Schaffer N.2	2	[-100, +100]	0
f_{12}	Matyas	2	[-10, +10]	0

Experiment 3. This experiment is conducted to compare the proposed algorithm with several evolutionary algorithms in terms of the execution time. We have measured the execution time using a computer with processor Intel Core 5 with a clock frequency of 2.53GHz, which supports Microsoft Windows 10 Pro on 8GB, RAM. Table 8 shows the results of this experiment.

From the results of *Experiment* 2 and 3, the following can be concluded:

1. In all datasets, the proposed BBOEAX algorithm has the best performance.

2. Due to the good performance of the proposed algorithm, the standard deviation of the BBOEAX algorithm is much lower than the other algorithms.

3. The proposed algorithm performs well in terms of time complexity, and does not have much time complexity than the basic BBO and GA.

Table 4. TSP instances with its details.

No.	*Name*	*Dimension*	*Optima length*
1	Oliver	30	423
2	Djibouti	38	6656
3	Eil	51	428
4	Berlin	52	7544
5	Pr	76	108159
6	Kroa	100	21285
7	Gr	137	69853
8	Ch	150	6532
9	U	159	42080
10	Qatar	194	9352
11	D	198	10557
12	Kroa	200	29368
13	Lin-Kernighan	318	42029

4. Although the proposed algorithm is hybrid, it still performs better in terms of time complexity than the PSO, SA, and DE algorithms.

Statistical analysis

To ensure the efficiency of the BBOEAX algorithm, some statistical analyses were applied on the proposed BBOEAX and other algorithms. Based on the results of table 5, in order to ensure the results obtained, three different statistical tests were examined. First, in order to statistically validate the superiority of the proposed BBOEAX, the Friedman's test was carried out, and the resulting Friedman statistic has been shown in table 9.

In order to ensure the statistical results, the results of the proposed BBOEAX algorithm were compared using ANOVA (ANalysis of VAriance) with five other algorithms. Table 10 presents the ANOVA test for the proposed algorithm relative to the five existing algorithms including BBO, GA, PSO, SA, and DE, respectively, and in all case, the p-value was less than 0.05.

The third test is the relative error (RE) to evaluate the performance of the proposed BBOEAX algorithm.

The absolute error ratio to the obtained resulting value is called a relative error (RE). In the other words, relative error is based on the size of measure, which is expressed in percentages not in a single unit, and is defined as follows:

$$RE = \frac{B - O}{O} \times 100 \qquad (7)$$

Table 5. Comparison of proposed algorithm with different evolutionary algorithms.

Functions		BBO [30]	DE [32]	GA [5]	PSO [7]	SA [5]	BBOEAX
f_1	SR	0	50	0	45	50	50
	Mean	2.98E-08	4.57E-25	1.28E-06	5.34E-08	7.81E-50	2.98E-08
	Std	2.32E-05	2.02E-24	9.06E-07	1.60E-07	3.35E-49	2.25E-08
f_2	SR	0	50	0	0	50	0
	Mean	5.40E-02	6.87E-14	7.37E-03	1.78E-02	2.88E-14	7.28E-04
	Std	1.43E-02	1.23E-13	1.96E-03	2.50E-02	7.28E-15	2.48E-04
f_3	SR	0	0	0	0	0	0
	Mean	1.11E-01	3.08E-02	3.48E-02	2.03E-01	6.59E-03	3.58E-03
	Std	2.98E-02	1.64E-02	3.00E-02	1.18E-01	7.98E-03	4.40E-03
f_4	SR	0	0	0	0	47	10
	Mean	3.772218	1.48E-04	3.22E-04	11.24737	2.02E-07	2.01E-04
	Std	1.159120	1.04E-03	1.72E-04	5.975528	4.94E-07	1.40E-03
f_5	SR	0	0	0	0	0	15
	Mean	1.53E-03	5.55E-05	1.46E-01	2.20E-01	9.230691	2.68E-07
	Std	7.08E-04	7.70E-05	9.02E-02	2.62E-01	3.953846	1.63E-07
f_6	SR	0	0	0	0	0	0
	Mean	3.30907	4.79E-01	6.711351	5.387153	1.056651	0.284873
	Std	1.496001	6.33E-01	1.206713	2.675939	4.39E-01	0.170509
f_7	SR	0	0	0	0	0	0
	Mean	-9.25997	-9.6602	-9.02469	-6.08352	-9.66006	-8.74362
	Std	2.74E-01	1.07E-14	2.75E-01	1.03100	7.18E-04	0.486548
f_8	SR	0	0	0	0	0	0
	Mean	-4.12784	-4.14734	-4.13968	-4.13526	-4.15567	-4.1276
	Std	2.81E-02	1.30E-02	2.46E-02	3.16E-02	2.53E-04	0
f_9	SR	0	50	0	2	50	50
	Mean	9.92E-05	1.84E-24	2.71E-06	1.18E-03	1.50E-32	3.06E-08
	Std	3.57E-05	6.51E-24	1.19E-06	2.26E-03	8.29E-48	1.53E-08
f_{10}	SR	0	50	0	0	50	15
	Mean	4.01E-02	6.00E-22	9.10E-04	6.91E-04	7.12E-41	4.60E-06
	Std	1.90E-02	2.03E-21	5.18E-04	1.60E-03	1.07E-40	4.14E-06
f_{11}	SR	50	50	50	50	24	50
	Mean	4.04E-10	0.00E+00	0.00E+00	0.00E+00	2.27E-07	0.00E+00
	Std	6.25E-10	0.00E+00	0.00E+00	0.00E+00	3.55E-07	0.00E+00
f_{12}	SR	50	50	50	50	14	50
	Mean	1.85E-09	6.98E-23	7.87E-18	7.74E-41	1.78E-06	0.00E+00
	Std	3.02E-09	2.71E-22	1.79E-17	3.31E-40	3.04E-06	0.00E+00

Table 6. Computational results for TSP instances with less than 100 cities.

Problem	Algorithm	Best	Mean	Worst	Std
Oliver30	BBO [30]	426.1971	485.3531	574.4661	35.546117
	GA [5]	491.3252	618.6262	699.7872	45.219135
	PSO [7]	424.6353	424.6353	424.6353	00000000
	SA [5]	493.4304	588.2051	660.7181	36.695123
	DE [32]	489.6571	603.4635	652.5531	33.748171
	BBOEAX	**424.4640**	**424.4641**	**424.4641**	**00000000**
Djibouti38	BBO	8058.335	8560.427	8976.178	234.73575
	GA	9554.783	12555.41	14805.01	1232.0711
	PSO	10731.38	13063.77	15714.61	1290.3118
	SA	9040.522	11264.95	13439.68	877.70417
	DE	10846.29	12359.65	13643.71	545.04813
	BBOEAX	**7835.317**	**8083.586**	**8297.894**	**110.12118**
Eli51	BBO	508.4931	559.3012	674.0454	42.448216
	GA	608.4192	659.4078	730.6268	34.618139
	PSO	713.3733	779.6131	837.4869	39.669347
	SA	671.6291	715.0792	761.4861	26.200874
	DE	579.5692	605.0076	648.2844	18.714788
	BBOEAX	**431.5877**	**446.4324**	**461.3597**	**9.8121793**
Berlin52	BBO	7982.5481	8655.9231	9438.2871	382.30432
	GA	9624.1782	10611.933	11637.985	499.31688
	PSO	13178.384	14323.472	15552.447	644.14576
	SA	10895.756	12300.661	13543.494	734.75249
	DE	8539.9484	9581.8117	10348.811	523.56614
	BBOEAX	**7551.6247**	**7816.7274**	**8373.5017**	**248.05667**
Pr76	BBO	137461.27	147251.95	162164.6	6466.6002
	GA	128674.51	134545.24	141685.4	3401.7594
	PSO	164352.95	191527.09	216085.6	12754.771
	SA	147363.88	163999.80	186892.2	11105.078
	DE	124397.16	129074.27	133429.5	2169.7517
	BBOEAX	**115743.94**	**121185.88**	**129366.5**	**3245.9509**

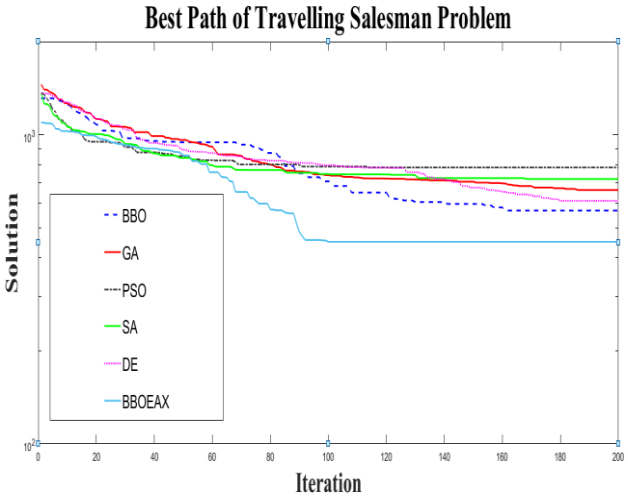

Figure 3. Comparison results of BBOEAX with the other algorithms on Eli51.

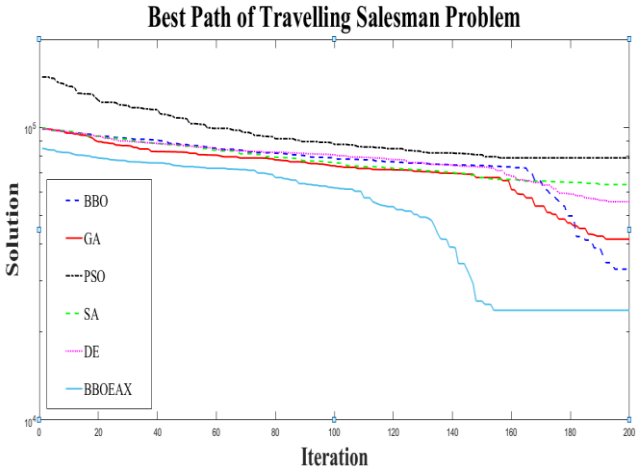

Figure 4. Comparison results of BBOEAX with the other algorithms on Kroa100.

Table 7. Computational results for TSP instances above 100 cities.

Problem	Algorithm	Best	Mean	Worst	Std
Kroa100	BBO [30]	26397.428	31569.145	37348.299	2747.1506
	GA [5]	34486.736	38500.081	43582.496	2696.6862
	PSO [7]	72736.435	78566.767	83624.192	3064.2611
	SA [5]	58908.485	64817.101	70260.192	3138.7088
	DE [32]	46827.651	52434.201	59280.184	3054.6447
	BBOEAX	**22234.268**	**22549.539**	**23218.721**	**203.5753**
Gr137	BBO	76554.967	82036.835	88248.651	2906.7115
	GA	79253.674	84167.192	89793.176	3400.6297
	PSO	98347.624	104046.754	112748.35	3181.8374
	SA	93418.091	97932.479	106248.47	3218.0138
	DE	76492.638	88030.612	93752.564	3089.0839
	BBOEAX	**71057.693**	**73950.368**	**79365.196**	**1991.7772**
Ch150	BBO	796.47171	7911.7371	9421.674	1410.0441
	GA	11347.647	12555.495	13428.163	432.15277
	PSO	13861.273	14612.601	15293.865	383.69794
	SA	12728.672	13456.060	14656.793	453.54869
	DE	9426.8941	10473.475	11736.628	580.95523
	BBOEAX	**6751.6582**	**7093.4667**	**7433.683**	**173.00415**
U159	BBO	49875.287	56282.732	62751.593	3186.2762
	GA	68324.077	74064.228	78268.584	2366.4047
	PSO	87542.724	90809.042	93372.207	1671.8518
	SA	78726.698	82412.124	85208.048	1679.4464
	DE	58052.709	64236.296	69317.191	3286.0455
	BBOEAX	**43502.289**	**45477.601**	**47441.104**	**1179.6131**
Qatar194	BBO	10633.805	11393.943	12168.959	451.34112
	GA	15169.959	16688.867	18095.085	868.58783
	PSO	19394.101	23033.71	26396.321	1648.3598
	SA	13346.514	14073.736	14776.197	392.79677
	DE	11947.502	12690.896	13497.604	408.04494
	BBOEAX	**9735.741**	**10291.172**	**10895.157**	**319.96251**
D198	BBO	16397.491	19900.069	23557.945	2063.856
	GA	27144.156	33506.772	38631.578	3075.8319
	PSO	43840.630	51076.161	56361.112	2562.8273
	SA	38098.781	41819.347	48140.933	2127.9538
	DE	17624.941	24094.804	29615.694	3147.4947
	BBOEAX	**12587.326**	**15606.778**	**17718.439**	**1319.9594**
Kroa200	BBO	36369.528	42063.848	46071.994	2131.3622
	GA	39486.949	46899.655	52119.066	3100.8285
	PSO	57753.542	64231.523	69233.273	3187.7587
	SA	66625.107	79782.986	226335.37	27882.541
	DE	49627.103	56737.964	62006.027	3096.0753
	BBOEAX	**33206.541**	**34527.082**	**36857.621**	**896.22611**
Lin-Kernighan318	BBO	48053.932	54926.895	61832.942	3597.9537
	GA	59123.764	64088.804	71452.102	3151.3728
	PSO	75673.101	82638.126	88263.931	2527.3489
	SA	88703.267	96800.513	104894.44	3648.1643
	DE	67263.018	74045.119	83239.693	3444.9274
	BBOEAX	**45109.376**	**46755.492**	**49875.865**	**1230.3681**

Table 8. Comparison of proposed algorithm with different evolutionary algorithms in term of execution time.

Algorithms / Dataset	BBO [30]	GA [5]	PSO [7]	SA [5]	DE [32]	BBOEAX
Oliver	10.98	11.58	13.42	12.67	15.29	11.28
Djibouti	13.61	14.2	17.84	15.93	17.81	14.67
Eli51	20.32	22.64	25.62	27.94	29.41	23.83
Berlin	20.46	23.88	27.64	28.61	30.87	26.23
Pr76	36.64	37.89	38.73	37.45	39.54	38.2
Kroa100	57.05	60.52	63.82	61.42	66.87	63.48
Gr137	96.11	99.73	103.47	98.8	108.62	101.73
Ch150	112.19	118.34	126.94	117.28	128.12	116.72
U150	124.96	128.18	135.76	126.44	132.71	128.27
Qatar194	177.63	182.75	192.78	180.49	189.16	184.67
D198	185.4	188.72	193.24	187.32	197.21	189.49
Kr200	193.52	201.38	208.91	198.27	215.73	203.42
LK318	469.41	498.24	516.81	548.62	586.46	482.08

Table 9. Ranking of algorithms based on Friedman's test.

Algorithm	Average ranking	Final rank
BBO	2.23	2
GA	3.92	4
PSO	5.54	6
SA	4.85	5
DE	3.46	3
BBOEAX	1.00	1

Table 11. Results of the relative error.

Problem	BBO	GA	PSO	SA	DE	BBOEAX
Oliver30	14.74	46.24	0.386	39.05	42.66	0.346
Djibouti38	28.61	88.63	96.27	69.24	85.69	21.44
Eli51	30.41	53.75	81.78	66.73	41.07	4.095
Berlin52	14.73	40.71	89.85	63.04	27.00	3.670
Pr76	36.14	24.39	77.07	51.62	19.33	12.04
Kroa100	48.31	80.87	269.1	204.5	146.3	5.938
Gr137	17.44	20.49	48.95	40.19	26.02	5.865
Ch150	21.11	92.20	123.6	105.9	60.33	8.590
U159	33.75	76.00	115.8	95.84	52.65	8.074
Qatar194	21.83	78.45	146.2	50.48	35.70	10.04
D198	88.50	217.3	383.8	296.1	128.2	47.83
Kroa200	43.23	59.69	118.7	171.6	93.19	17.56
L-K318	30.68	52.48	96.62	130.3	76.17	11.24

Table 10. ANOVA test for BBOEAX *vs.* BBO, GA, PSO, SA, and DE at the 0.05 significant level.

No.	BBOEAX *vs.* BBO		BBOEAX *vs.* GA		BBOEAX *vs.* PSO		BBOEAX *vs.* SA		BBOEAX *vs.* DE	
	$x-y(\%)$	$p-value$	$x-y(\%)$	$p-value$	$x-y(\%)$	$p-value$	$x-y(\%)$	$p-value$	$x-y(\%)$	$p-value$
1	-68.9483	1.16E-19	-189.256	8.17E-67	-0.17110	0.97968	-157.520	8.04E-56	-182.9	1.01E-64
2	-498.973	0.022582	-4598.39	4.16E-50	-5317.79	2.38E-58	-3135.12	1.25E-31	-4273.0	3.49E-46
3	-112.868	8.14E-31	-212.974	1.54E-63	-333.176	9.80E-93	-268.646	2.98E-78	-158.57	9.74E-47
4	-839.198	5.677E-9	-2795.20	4.45E-48	-6506.74	1.2E-101	-4483.92	2.46E-76	-1765.0	3.87E-27
5	-26066.0	8.15E-28	-13359.3	2.36E-10	-70341.2	1.70E-81	-42813.9	4.86E-51	-7888.3	0.000103
6	-9019.61	2.16E-27	-15950.5	1.51E-54	-56017.2	9.5E-140	-42267.5	3.3E-119	-29884.6	1.19E-94
7	-8086.45	4.06E-20	-10216.8	5.53E-28	-30096.3	1.12E-87	-23982.1	1.10E-72	-14080.2	4.54E-42
8	-818.270	0.000010	-5462.02	1.25E-71	-7519.13	7.20E-93	-6362.59	1.44E-81	-3380.00	6.99E-44
9	-10805.1	9.52E-41	-28586.6	1.5E-100	-45331.4	1.1E-133	-36934.5	9.0E-119	-18758.6	3.45E-72
10	-1102.76	6.451E-7	-6397.69	1.23E-70	-12742.5	7.6E-118	-3782.56	7.88E-41	-2399.72	2.12E-22
11	-4293.29	2.17E-10	-17899.9	1.27E-66	-35469.3	7.0E-113	-26212.5	1.21E-91	-8488.02	2.11E-28
12	-7536.76	0.013024	-12372.5	0.000059	-29704.4	1.41E-18	-45255.9	2.29E-33	-22210.8	5.76E-12
13	-8171.40	6.65E-20	-17333.3	4.28E-52	-35882.6	1.48E-98	-50045.0	2.9E-122	-27289.6	6.19E-80

Also figure 5 and figure 6 show the stability of the ompared algorithms in achieving coverage, that all algorithms were executed 30 times independently (the algorithms were executed 30 times per TSP

instances for each number of individuals defined in table 5, i.e. in total, 180 runs per TSP instances). According to figure 5, the proposed BBOEAX had a good performance to stablish full coverage. The optimum tour length was determined by *O*, and the obtained result via the proposed algorithm was determined by *B*. The relative error results have been shown in table 11. Figure 7 also shows the obtained REs of algorithms. (the mean value was considered.)

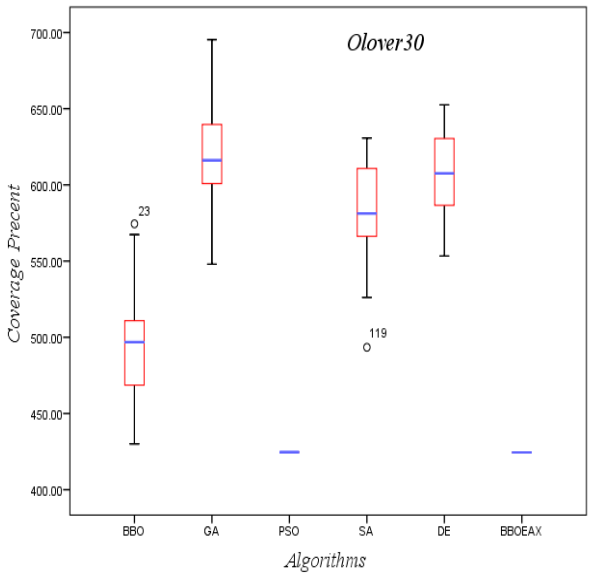

Figure 5. Stability analysis on the branch coverage on Eli51.

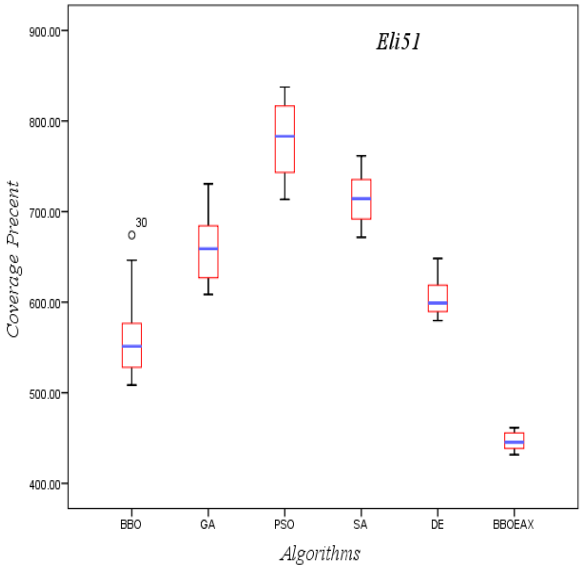

Figure 6. Stability analysis on the branch coverage on Oliver30.

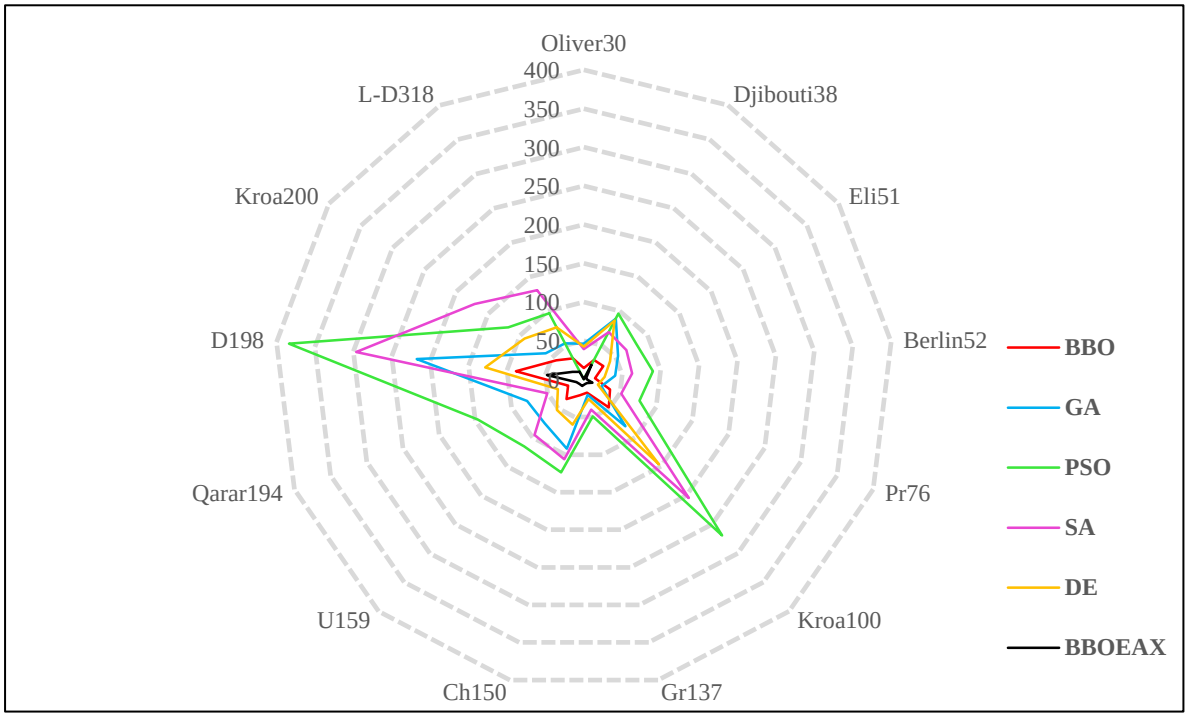

Figure 7. RE comparisons of the all algorithms on TSP dataset.

7. Conclusion

In this work proposed a new hybrid BBO with EAX for solving optimization problems including the traveling salesman problem. In this research work, using the capabilities of the EAX operator, the proposed BBOEAX algorithm eliminated the weaknesses of the original BBO algorithm, and it obtained better solutions, and performed better than the original version of BBO and other evolutionary algorithms. The EAX operator had different strategies that was chosen in the proposed algorithm after examining the best possible strategy. The performance of the EAX operator was that with random choices, the diversity was increased in the population, which would provide better solutions to the algorithm. The BBO algorithm alone had a high performance, and its combination with the EAX operator increased its efficiency. The efficiency of the proposed BBOEAX algorithm was compared with the original version of BBO and other evolutionary algorithms, and the results obtained showed that the proposed BBOEAX algorithm had a high efficiency in obtaining quadratic solutions.

References

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

[2] Matai, R., Singh, S., & Lal, M. (2010). Traveling salesman problem: an overview of applications, formulations, and solution approaches, in: Traveling Salesman Problem, Theory and Applications, In Tech, pp. 1–24.

[3] Yang, X.S. (2014). Firefly algorithms, in: Nature-Inspired Optimization Algorithms, Elsevier, 2014, pp. 111–127.

[4] Nagata, Y. & Kobayashi, S. (1997). Edge Assembly Crossover: A High-power Genetic Algorithm for the Traveling Salesman Problem. In PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, pp. 450-457.

[5] Goldberg, D. E., (1989). Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1989.

[6] Safaee, B. & Kamaleddin Mousavi Mashhadi, S. (2017). Optimization of fuzzy membership functions via PSO and GA with application to quad rotor, Journal of AI and Data Mining (JAIDM), vol. 5, no. 1, 2017, pp. 1-10.

[7] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization, Proc. IEEE Int. Conf. Neural Network, vol. 4, pp. 1942-1948, 1995.

[8] Zhang, X., Kang, Q., Cheng, J. & Wang, X. (2018). A Novel Hybrid Algorithm Based on Biogeography-Based Optimization and Grey Wolf Optimizer. Applied Soft Computing, Elsevier, vol. 67, pp. 197-214, 2018.

[9] Yang, J. & Li, L. (2017). Improved Biogeography-Based Optimization Algorithm for Mobile Robot Path Planning. Proceedings of 2017 Chinese Intelligent Systems Conference, Lecture Notes in Electrical Engineering 460, pp. 219-229.

[10] Bansal, J.C. (2016). Modified Blended Migration and Polynomial Mutation in Biogeography-Based Optimization. Harmony Search Algorithm, Advances in Intelligent Systems and Computing 382, pp. 217-225.

[11] Farswan, P., Bansal, J.C. & Deep, K. (2016). A Modified Biogeography Based Optimization. Harmony Search Algorithm, Advances in: Intelligent Systems and Computing 382, pp. 227-238.

[12] Al-Roomi, A.R. & El-Hawary, M.E. (2016). Metropolis biogeography- based optimization. Information Sciences, vol. 360, pp. 73-95.

[13] Chen, X., Tianfield, H., Du, W. & Liu, G. (2016). Biogeography-based optimization with covariance matrix-based Migration. Applied Soft Computing, vol. 45, pp. 71-85.

[14] Fan, H., Zhong, Y. & Zeng, G. (2017). Overlapping community detection based on discrete biogeography optimization. Appl Intell, vol. 48, pp. 1314–1326.

[15] Khishe, M., Mosavi, M.R. & Kaveh, M. (2017). Improved migration models of biogeography-based optimization for sonar dataset classification by using neural network. Applied Acoustics, vol. 118, pp. 15-29.

[16] Paraskevopoulos, A., Dallas, P.I., Siakavara, K. & Goudos, S.K. (2017). Cognitive Radio Engine Design for IoT Using Real-Coded Biogeography-Based Optimization and Fuzzy Decision Making. Wireless Pers Commun, vol. 97, pp. 1813-1833.

[17] Feng, Q., Liu, S., Zhang, J., Yang, G. & Yong, L. (2017). Improved biogeography-based optimization with random ring topology and Powell's method. Applied Mathematical Modelling, vol. 41, pp. 630-649.

[18] Lohokare, M. R., Pattnaik, S. S., Panigrahi, B. K. & Das, S. (2013). Accelerated biogeography-based optimization with neighborhood search for optimization. Applied Soft Computing, vol. 13, pp. 2318-2342.

[19] Nagata, Y. & Kobayashi, S. (2006). An analysis of Edge Assembly Crossover for the Traveling Salesman Problem. 0-7803-5731-0/99/$10.00 01999 IEEE.

[20] Nagata, Y. (2007). Edge Assembly Crossover for the Capacitated Vehicle Routing Problem. EvoCOP 2007, LNCS 4446, pp. 142–153.

[21] Haque, M. J. & Magld, K. W. (2012). Improving the Solution of Traveling Salesman Problem Using Genetic, Memetic Algorithm and Edge assembly

Crossover. (IJACSA) International Journal of Advanced Computer Science and Applications, vol. 3, no. 7, pp. 108-111.

[22] Błocho, M. & Czech, Z. J. (2012). A parallel EAX-based algorithm for minimizing the number of routes in the vehicle routing problem with time windows. Proceedings of the 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems, pp. 255-265.

[23] Nagata, Y. (2006). Fast EAX Algorithm Considering Population Diversity for Traveling Salesman Problems. EvoCOP 2006, LNCS 3906, pp. 171–182.

[24] Kocer, H. E. & Akca, M. R. (2014). An Improved Artificial Bee Colony Algorithm with Local Search for Traveling Salesman Problem. Cybernetics and Systems: An International Journal, vol. 45, pp. 635–649, Taylor & Francis Group.

[25] Osaba, E., Yang, X. S., Diaz, F. & Garcia, P. L. (2016). An improved discrete bat algorithm for symmetric and asymmetric Traveling Salesman Problems. Engineering Applications of Artificial Intelligence, vol. 48, pp. 59-71.

[26] Cheng, C. Y., Li. H. F. & Bao, C. H. (2015). Hybrid Artificial Fish Algorithm to Solve TSP Problem. Proceedings of the 6th International Asia Conference on Industrial Engineering and Management Innovation.

[27] Ardalan, Z., Karimi, S., Poursabzi, O. & Naderi, B. (2014). A novel imperialist competitive algorithm for generalized traveling salesman problems. Applied Soft Computing, vol. 26, pp. 546-555.

[28] Liao, Y. F., Yau, D. H. & Chen, C.L. (2012). Evolutionary algorithm to traveling salesman problems. Computers and Mathematics with Applications, vol. 64, pp. 788-797.

[29] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html/ Accessed 5 April 2018.

[30] Salehi, A. & Masoumi, B. (2019), Participative Biogeography-Based Optimization. Journal of Optimization in Industrial Engineering, vol. 12, no. 1, pp. 79- 91.

[31] Chen, Y. & Jia, Y. (2015). Research on Traveling Salesman Problem Based on the Ant Colony Optimization Algorithm and Genetic Algorithm. The Open Automation and Control Systems Journal, vol. 7, pp. 1329-1334

[32] Wei, H., Hao, Z., Huang, H., Li, G. & Chen, Q. (2016). A Real Adjacency Matrix-Coded Differential Evolution Algorithm for Traveling Salesman Problems. BIC-TA 2016, Part II, CCIS 682, pp. 135–140.

[33] Liu, Z. G., Ji, X. H. & Liu, Y. X. (2018). Hybrid non-parametric particle swarm optimization and its stability analysis. Expert Systems with Applications, vol. 92, pp. 256-275.

[34] Guo, W., Wang, L. Si, C. Zhang, Y. & Tian, H. (2016). Novel migration operators of biogeography-based optimization and Markov analysis. Soft Comput, Springer-Verlag Berlin Heidelberg, 21, vol. 22, pp. 6605-6632.

[35] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ Accessed 5 April 2018.

[36] Brest, S. Greiner, B. Boškovic, et al. (2006). Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Transactions on Evolutionary Computation, vol. 10, pp. 646–657.

[37] Ranjini, S. & Murugan, S. (2018). Memory based Hybrid Dragonfly Algorithm for Numerical Optimization Problems. Expert Systems with Applications,vol. 83, pp. 63-78.

[38] Gong, W., Cai, Z. & Ling, C.X. (2010). DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization. Soft Comput, Springer-Verlag, vol. 15, no. 4, pp. 645-665.

نشریه هوش مصنوعی و داده کاوی

حل مسئله فروشنده دوره گرد مبتنی بر الگوریتم بهینه‌سازی جغرافیای زیستی و اسمبلی مجدد لبه

عباس صالحی و بهروز معصومی*

دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه آزاد اسلامی واحد قزوین، قزوین،

ارسال ۲۰۱۸/۱۲/۲۷؛ بازنگری ۲۰۱۹/۱۲/۲۳؛ پذیرش ۲۰۲۰/۰۲/۰۸

چکیده:

الگوریتم بهینه‌سازی مبتنی بر جغرافیای زیستی (BBO) اخیراً به دلیل سادگی در اجرای، کارآیی و تعداد کم پارامترها مورد توجه محققان قرار گرفته است. الگوریتم BBO در مسائل بهینه‌سازی یکی از الگوریتم‌های جدیدی است که بر اساس مفهوم جغرافیایی ارائه شده است. این الگوریتم از ایده مهاجرت حیوانات برای یافتن زیستگاه‌های مناسب برای حل مسائل بهینه‌سازی استفاده می‌کند. الگوریتم BBO دارای سه عملگر اصلی به نام‌های مهاجرت، جهش و انتخاب نخبگان است. اپراتور مهاجرت نقش بسیار مهمی در به اشتراک گذاری اطلاعات در بین زیستگاه‌های کاندید ایفا می‌کند. الگوریتم اصلی BBO، به دلیل اکتشاف و شناسایی ضعیف، گاهی اوقات نتایج مطلوبی را بدست نمی‌آورد. از طرف دیگر، اسمبلی مجدد لبه (EAX) یکی از قوی‌ترین عملگرهای متقاطع برای دستیابی به فرزندان بوده است و این باعث افزایش تنوع جمعیت می‌شود. ترکیبی از الگوریتم BBO و EAX می‌تواند راندمان بالایی در حل مسائل بهینه‌سازی از جمله مسئله فروشنده دوره گرد (TSP) داشته باشد. در این مقاله، ما ترکیبی از این رویکردها را برای حل مسئله فروشنده دوره گرد پیشنهاد می‌کنیم. رویکرد ترکیبی جدید با مجموعه داده‌های استاندارد برای TSP در TSPLIB بررسی شده است. در آزمایشات، عملکرد الگوریتم پیشنهادی بهتر از الگوریتم اصلی BBO و چهار الگوریتم فرابتکاری پرکاربرد دیگر است.

کلمات کلیدی: بهینه‌سازی مبتنی بر جغرافیای زیستی، الگوریتم‌های تکاملی، اسمبلی مجدد لبه، الگوریتم ژنتیک، مسئله فروشنده دوره گرد.