RAIRO-Oper. Res. 53 (2019) 1033–1059
https://doi.org/10.1051/ro/2019039

RAIRO Operations Research
www.rairo-ro.org

A SELF-ADAPTIVE BIOGEOGRAPHY-BASED ALGORITHM TO SOLVE THE SET COVERING PROBLEM

Broderick Crawford[1], Ricardo Soto[1,*], Rodrigo Olivares[2], Luis Riquelme[1], Gino Astorga[1], Franklin Johnson[3], Enrique Cortés[3], Carlos Castro[4] and Fernando Paredes[5]

Abstract. Using the approximate algorithms, we are faced with the problem of determining the appropriate values of their input parameters, which is always a complex task and is considered an optimization problem. In this context, incorporating online control parameters is a very interesting issue. The aim is to vary the parameters during the run so that the studied algorithm can provide the best convergence rate and, thus, achieve the best performance. In this paper, we compare the performance of a self-adaptive approach for the biogeography-based optimization algorithm using the mutation rate parameter with respect to its original version and other heuristics. This work proposes altering some parameters of the metaheuristic according to its exhibited efficiency. To test this approach, we solve the set covering problem, which is a classical optimization benchmark with many industrial applications such as line balancing production, crew scheduling, service installation, databases, among several others. We illustrate encouraging experimental results, where the proposed approach is capable of reaching various global optimums for a well-known instance set taken from the Beasleys OR-Library, and sometimes, it improves the results obtained by the original version of the algorithm.

Mathematics Subject Classification. 68W25, 90C27, 93B40.

Received April 29, 2017. Accepted March 29, 2019.

1. Introduction

In the areas of optimization and engineering, there are a variety of problems that are complex to solve regarding computational costs, and thousands, even millions, of iterations are required to find their optimal solutions. These problems are commonly called NP-hard [23], and one of the alternatives for solving them is the exact algorithms, such as branch & bound [27], branch & cut [36], and backtracking [22]. However, these methods are not appropriate for large-scale problems because they require large computational capacities, time

Keywords. Metaheuristics, biogeography-based optimization algorithm, set covering problem.

[1] Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile.
[2] Universidad de Valparaíso, Valparaíso, Chile.
[3] Universidad de Playa Ancha, Valparaíso, Chile.
[4] Universidad Técnica Federico Santa María, Valparaíso, Chile.
[5] Universidad Diego Portales, Santiago, Chile.
*Corresponding author: ricardo.soto@pucv.cl

Article published by EDP Sciences

and cost to reach a precise solution [3]. Therefore, we use metaheuristics to design or improve general heuristic procedures that require high performance. The general purpose of a metaheuristic is to find a good solution for the problem in a reasonable computational time (not necessarily the optimal solution, as in the case of exact algorithms).

One of the fairly new metaheuristics is the biogeography-based optimization algorithm (BBOA). This method belongs to the family of population algorithms for minimization problems with binary and real variables, and it is useful for maximizing and minimizing problems [30]. BBOA is inspired by the concept of the Habitat Suitability Index (HSI), which is generated from the characteristics of a habitat. The better the habitat characteristics are, the higher the HSI, and the worse the habitat characteristics are, the lower the HSI. Additionally, when the habitat has a high HSI, more species live there, unlike a habitat with a lower HSI [30,51]. Each habitat also has immigration and emigration rates and mutation probabilities, which derive from the number of species in the habitat.

In this work, we propose a self-adaptive biogeography-based optimization algorithm (SA-BBOA) that allows for the setting of the mutation rate parameter during the run according to the best solutions found using the fitness value (HSI). This approach is applied to solve the set covering problem (SCP), whose goal is to cover a range of needs at the lowest cost. The SCP can be applied for location services, selection of files in a database, simplifying boolean expressions, slot allocation, among others [5]. Currently, there are many papers that deal with resolution methods for the set covering problem. These are exact methods [3,4,20] and heuristic methods to solve a range of problems such as in [26]. The set covering problem was also successfully solved with metaheuristics such as the tabu search [7], simulated annealing [47], genetic algorithm [24,32], ant colony optimization [2,32], particle swarm optimization [12], hybrid algorithms [1,46], the hybrid ant algorithm [13], binary cat swarm optimization [9], the bat algorithm [14], the cuckoo search [42], artificial bee colony [11], the binary firefly algorithm [15], the shuffled frog leaping algorithm [17], the soccer league competition algorithm [25], the binary black hole algorithm [38], the binary fruit fly algorithm [18], and the fish swarm algorithm [45].

BBOA has already been used to solve many combinatorial optimization problems, among them, the classic traveling salesman problem. The TSP consists of finding the shortest route between a set of points, visiting all of them only once and returning to the starting point [35]. It was solved by using BBOA in [34], demonstrating that this method behaves very effectively for combinatorial optimization problems, and it is even better than other traditional methods that are inspired by nature. Additionally, BBOA has been used to solve optimization problems such as in [29], where it was indicated that BBO generally performs better than the genetic algorithm (GA) and particle swarm optimization (PSO) in handling constrained single-objective optimization problems. Undoubtedly, the BBOA is a method that may have a great potential to solve the SCP.

The remainder of this paper is structured as follows. Section 2 presents a detailed description of the SCP and an example. Section 3 describes the BBOA we use. In Section 4, all modifications to the BBOA are discussed for solving the SCP. Subsequently, the experimental results and comparisons with other algorithms are given in Section 5. Finally, the conclusion and future works are in Section 6.

2. Problem statement

The set covering problem (SCP) is a classic combinatorial optimization problem, belonging to NP-hard class [23] that has been used in a wide range optimization problems including airline and bus crew scheduling [39], the location of emergency facilities [48], railway crew management [6], steel production [49], and vehicle scheduling [21].

The SCP consists in finding a set of elements that covers a range of needs at the lowest cost. In its matrix form, a feasible solution corresponds to a subset of columns and the needs are associated with the rows and treated as constraints. The problem aims at selecting the columns that optimally cover all the rows.

Formally, we define the problem as follows: let $A = (a_i^j)$ be a binary matrix with M-rows ($\forall\ i \in I = \{1, \ldots, M\}$) $\times$ N-columns ($\forall\ j \in J = \{1, \ldots, N\}$), and let $C = (c_j)$ be a vector representing the cost of each column j, assuming that $c_j > 0$, $\forall\ j \in N$. Then, we observe that a column $j \in \{1, \ldots, N\}$ covers a row i if

TABLE 1. Doctors and list of procedures.

		Doctors					
		A	B	C	D	E	F
Procedures	1	✓			✓		
	2				✓	✓	
	3		✓	✓			
	4			✓			✓
	5		✓	✓			✓
	6		✓				

$a_i = 1$. Therefore, we have:

$$a_i^j = \begin{cases} 1, & \text{if row } i \text{ can be covered by column } j \\ 0, & \text{otherwise} \end{cases}.$$

The SCP finds a minimum cost subset S of columns such that each row is covered by at least one column of j. An integer programming formulation of the SCP is as follows:

$$\begin{gathered} \texttt{minimize} \sum_{j=1}^{n} c_j x_j \\ \text{subject to:} \\ \sum_{j=1}^{n} a_i^j x_j \geq 1 \quad \forall\, i \in I = \{1, \ldots, M\} \\ x_j \in \{0, 1\} \quad \forall\, j \in J = \{1, \ldots, N\}. \end{gathered} \tag{2.1}$$

It is possible to consider at least two methods to work with this problem: unicost and nonunicost. The unicost variant states that the cost for including a decision variable is equal to 1 for all of them. Conversely, the nonunicost variant considers that the decision variables can have a different inclusion value. To clarify the set covering problem, we first present an example with a unicost vector, $c_j = 1$.

A medical center needs to keep doctors on call so that qualified individuals are available to perform every medical procedure that might be required (there is an official list of such procedures). For each of several doctors available for on-call duty, the additional salary they need to be paid and which procedures they can perform are known. The goal is to choose doctors so that each procedure is covered at a minimum cost. A list of procedures and doctors is illustrated in Table 1.

The binary-programming model includes the following decision variables:

$$x_j = \begin{cases} 1, & \text{if Doctor } j \text{ is selected} \\ 0, & \text{otherwise} \end{cases}. \tag{2.2}$$

The objective function minimizes the number of doctors, *i.e.*:

$$\texttt{minimize} \sum_{j=1}^{6} x_j. \tag{2.3}$$

Constraints: each procedure should be covered by at least one doctor. This can be seen in the following summary:

(1) Procedure 1 is covered by Doctor 1 or 4.
(2) Procedure 2 is covered by Doctor 4 or 5.
(3) Procedure 3 is covered by Doctor 2 or 3.
(4) Procedure 4 is covered by Doctor 3 or 6.
(5) Procedure 5 is covered by Doctor 2, 3 or 6.
(6) Procedure 6 is covered by only Doctor 2.

Thus, the integer programming model is as follows:

$$\begin{gathered}
\texttt{minimize}\ x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\
\text{subject to:} \\
x_1 + x_4 \geq 1 \\
x_4 + x_5 \geq 1 \\
x_2 + x_3 \geq 1 \\
x_3 + x_6 \geq 1 \\
x_2 + x_3 + x_6 \geq 1 \\
x_2 \geq 1 \\
x_j \in \{0,1\}, \forall\, j \in \{1,\ldots,6\}
\end{gathered}$$

Modeling as a unit cost optimization problem, we can find different optimal solutions. One of the solutions is given by $\langle x_1, x_2, x_3, x_4, x_5, x_6\rangle = \langle 0,1,0,1,0,1\rangle$, and its objective value is 3. This solution is represented by the binary vector in Figure 1, where each value x_j is a component of the solution. The optimal solution represents the minimum number of doctors to cover all procedures.

We can transform this example into a nonunit cost problem; it is necessary to differentiate the cost of each doctor. The component c_j in the cost vector is associated with each x_j, $\forall\, j \in \{1,\ldots,6\}$. If we consider that the cost for each doctor is stated in Table 2, the objective function becomes:

$$\texttt{minimize} \sum_{j=1}^{6} c_j x_j. \tag{2.4}$$

The tuple $\langle x_1, x_2, x_3, x_4, x_5, x_6\rangle = \langle 0,1,1,1,0,0\rangle$ gives the minimum value equal to 160. The vector that describes the new optimal solution is depicted in Figure 2.

0	**1**	**0**	**1**	**0**	**1**
x_1	x_2	x_3	x_4	x_5	x_6

FIGURE 1. Representation of binary unit cost solution.

TABLE 2. Cost of selecting doctors.

Doctors	Cost (c_j)
A	55
B	65
C	35
D	60
E	50
F	60

0	1	1	1	0	0
x_1	x_2	x_3	x_4	x_5	x_6

FIGURE 2. Representation of binary nonunit cost solution.

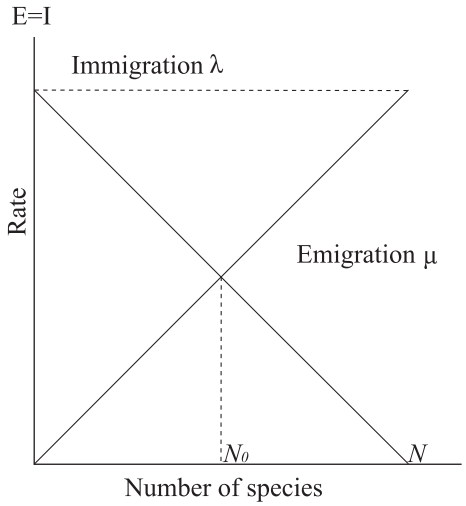

FIGURE 3. Species model of a single habitat.

However, to apply self-adaptive algorithm correctly, we define a binary vector H_i as the ith solution of the set covering problem and H_i^j as the jth decision variable, whose value is 1 if this component is part of the solution, or 0 otherwise.

3. Biogeography-based optimization algorithm

Biogeography studies the migration between habitats, speciation, and the extinction of species. In this research line, Ma & Simon proposed the BBOA based on the mathematical models of biogeography proposed in the 1960s [30].

3.1. The biogeography phenomenon

The biogeography phenomenon is based on the concept of biogeography, which deals with the distribution of species that depend on different factors such as rainfall, diversity of topographic features, temperature, and land area. Biogeography describes how species migrate from one island or habitat to another, including how new species arise and how species go extinct. A habitat is an island that is geographically isolated from other habitats [30]. A habitat that has a High Suitability Index (HSI) is geographically well suited for the life of the species. When a species shows a high HSI, its migration probability increases, which can lead to a species changing its current habitat for a nearby habitat with a lower HSI value. This process is named emigration. In immigration, the species move toward the high HSI habitat having few species. Then, based on the number of species, it is possible to predict the rate of immigration and emigration. Habitats with a low HSI have a high species immigration rate because of its sparse populations and a high rate of emigration, such as conditions causing rapid departure or the extinction of species. This behavior is shown in Figure 3, where I and E are the highest rates of immigration and emigration, and these are the same for simplicity. N is the maximum number of species. Finally, λ is the immigration rate, and μ is the emigration rate.

3.2. Optimization algorithm

In complexity theory, several problems are studied due to the exponential growth of the set of potential solutions. The set covering problem is one of these problems. The exact methods are a good alternative if we need to guarantee the optimal solution or we need to determine that there is no solution. However, when we try to solve the most difficult instances of these problems, a tremendous increase in the runtime appears for exact methods. For this reason, many problems are solved by metaheuristics. It is known that these algorithms do not guarantee finding the best solution, but they provide a sufficient solution in a shorter resolution time.

To solve the most difficult instances of the set covering problem, the basis of biogeography theory mentioned above is used. For this, we consider the existence of a set of potential solutions, analogous to a set of habitats with a diversity of species, where the best solutions will be those with the highest HSI and the poorest solutions with have the smallest HSI. The HSI measure of similarity of each solution to the fitness in other optimization algorithms based on population can be derived from the objective function. Then, the components of a solution are given by their characteristics, called the suitability index variable (SIV). In the case of the SCP, those correspond to the binary values of the N decision variables [37]. Habitats tend to be variable for the species. These are composed of several features (SIVs) that indicate how viable the habitat is to inhabit. According to these characteristics, species migrate and immigrate. In addition, some attributes of a habitat may appear randomly, such as natural disasters. To imitate the immigration and emigration behavior, BBOA proposes two operators, the migration operator (migration based on habitat characteristics) and the mutation operator (migration based on unexpected events in the habitat). Both phenomena are detailed in the next subsections.

3.2.1. Migration operator

Species can migrate between habitats. In BBOA, the characteristics of the solutions may affect others and themselves, using immigration and emigration rates to share information between them probabilistically. In this metaheuristic and based on Figure 3, the immigration curve is used to probabilistically determine whether or not to immigrate each feature, or SIV, in any solution. If a characteristic of the solution is selected to immigrate, a solution to migrate one of its characteristics is probabilistically selected randomly. Based on the above description, the main steps in the BBOA are detailed in Algorithm 1.

Algorithm 1 begins with a loop statement for each solution, and in each iteration, an H_i is selected under a probability λ_i. If a habitat H_i is selected, then it is necessary to take each component part of the solution. The size of the solution is given by the parameter N and represents the decision variables. If a habitat H_i is selected with probability λ_i, then two components $\{k, k'\} \in [1, \ldots, N]$ from H_i are selected. The first component is selected with the probability μ_i, and the second component is randomly selected. Then, $H_i^{k'}$ is copied to H_i^k. k and k' represent the habitat species. This process is known as exploration.

Algorithm 1 Migration operator

```
{PopSize is the size of the population}
for i = 1 to PopSize do
    Select H_i with probability λ_i
    if H_i is selected then
        {N is the size of the solution}
        Select H_i^k with a probability μ_i
        if H_i^k is selected then
            Randomly select a component k' ∈ [1, ..., N]
            Set H_i^k' = H_i^k
        end if
    end if
end for
```

The mutation operator helps the algorithms to avoid local optimum and explore the search space. During the optimization process, the mutation rate is not often a fixed value [50–52].

3.2.2. Mutation operator

A natural habitat may be affected by cataclysmic events that drastically changes its HSI [30]. This could cause a count of species that is different from its equilibrium value (species arriving from neighboring habitats, diseases, natural disasters and others). Thus, the HSI of the habitat could suddenly change due to random events.

In BBOA, the likelihood probability of each species ($P(species)$) is used to determine the mutation rates. These are determined by the balance between the immigration and emigration rates (Fig. 3) as a balance between these rates indicates that the probability that the number of *species* is greater; thus, species immigrate at a rate that is similar to the number of species that migrate in the same habitat. Given that, the best and worst habitats are less likely to have the number of *species*. This finding is explained in detail in [30]. Then, the mutation rate is represented by $m(species)$, and it is calculated as follows:

$$m(species) = m(\max)\left(\frac{1 - P(species)}{P(\max)}\right) \tag{3.1}$$

where $m(\max)$ is the maximum probability of the mutation parameter, and $P(\max)$ is the probability of a maximum existing *species*. Algorithm 2 explains this operator.

Algorithm 2 Mutation operator

```
{N is the size of the solution}
for all j, (∀ j = 1, ..., N) do
    Calculate mutation rate m(j) based on (3.1)
    Select SIV from H_i with probability m(j)
    if H_i^j is selected then
        Replace H_i^j with a randomly generated SIV.
    end if
end for
```

For each habitat, the probability of *species* is calculated, and then, each characteristic that is selected for mutation by this probability is randomly replaced with another SIV.

Note that in binary problems, the mutation operator for the exchange of an SIV acts so that $H_i^j = 1 - H_i^j$ [51].

3.2.3. Algorithm description

The features and steps are described in general terms of the BBOA:

Step 1. Initialize the parameters. Map the SIV and habitats according to the problem solutions. Initialize a maximum of species N (for simplicity, matching with the size of the population), immigration, emigration, and mutation maximum rates. An elitist parameter is used to save the best solutions.

Step 2. Randomly initialize a set of habitats, where each habitat corresponds to a possible solution to the problem.

Step 3. For each habitat, calculate the HSI and accordingly, the number of species (the greater the HSI, the greater the number of species). Then, calculate the rates of immigration and emigration.

Step 4. Probabilistically use the rates of immigration and emigration to modify the habitats (Migration operator).

Step 5. For each habitat, update the probability of a number of species. Then, mutate based on the mutation rate (mutation operator).

Step 6. Return to step 3, and finish until a stopping criterion is satisfied.

Note that after each habitat is modified (steps 2, 4, and 5), its feasibility as a problem solution should be verified. If it does not represent a feasible solution, then a method needs to be implemented to map it to the set of feasible solutions [30].

4. Biogeography-based optimization algorithm for the set covering problem

After the description of the problem and the technique for its use, we present in this section the implementation and adaptation of BBOA to obtain acceptable results for the SCP.

4.1. General considerations

As general considerations of the algorithm implementation and indifference to the BBOA base, we can highlight the following:

- The population is sorted in each generation, where the first solution is the highest HSI, and last solution is the worst.
- The length of each solution (SIVs) equals the length of the cost vector in all instances of SCP.
- A repair function for infeasible solutions is used.
- A parameter of elitism, which stores the 2 solutions with the lowest cost over the generations, is used.
- The stop criterion is a maximum number of generations.
- Adaptive mutation rate.

Similar to other evolutionary algorithms, the biogeography-based optimization begins with an initial population of potential solutions, called the "habitat". At each generation of the algorithm, a set of habitats is improved and mapped as a group of solutions to the set covering problem. During this process, the best solution is chosen as the best habitat according to its HSI value. The objective function gives the HSI value, and if it is high, then we can say that this habitat describes a good solution. In the next subsection, we explain how the HSI value is calculated. Sometimes, the potential solutions are unfeasible. In this work, we propose a technique that handles solutions that violate the restrictions. Finally, to improve the performance of the BBOA, we present a self-adaptive variation of the mutation rate parameter that allows us to enhance the quality of the solutions.

4.2. Fitness

An important point of the implemented algorithm is the calculation of the HSI, also called the fitness in other population-based optimization algorithms. In a BBOA, the solutions with a greater HSI are the best, while the worst are those with a low HSI. SCP is a minimization problem, and BOA must be adapted. The best solutions are those with the lowest value.

$$\text{HSI} = \frac{1}{\text{total cost solution}} = \frac{1}{\sum_{j=1}^{n} c_j x_j}. \tag{4.1}$$

4.3. Heuristic feasibility operator

Generally, metaheuristics may provide solutions that violate some constraints of the problem. For instance, a new SCP solution owning uncovered rows clearly violates a subset of constraints. To provide feasible solutions, the algorithm needs additional operators. To this end, we employ a heuristic operator that achieves the generation of feasible solutions and additionally eliminates column redundancy.

For making all solutions feasible, we calculate a percentage based on the cost of column j over the sum of all the constraint matrix rows covered by a column j, as shown in equation (4.2).

$$\frac{c_j}{\sum_{i=1}^{n} a_i^j}. \tag{4.2}$$

The unfeasible solutions are repaired by adding the columns of the solution that had the lower ratio. After this, a local optimization step is applied where column redundancy is eliminated. A column is redundant when it can be deleted, and the feasibility of the solution is not affected.

Algorithm 3 starts with the initialization of variables taken from the instance in Lines 1–5, The recognition of the rows that are not covered are in Lines 6 and 7. Between the statements 8 and 18, a "greedy" heuristic is run. Between the instructions 8 and 12, the columns with a lower ratio are added to the solution. Between lines 13 and 18, the redundant columns with higher costs are deleted, while the solution is feasible.

Algorithm 3 Heuristic feasibility operator.

1: $I \leftarrow$ The set of all rows.
2: $J \leftarrow$ The set of all columns.
3: $\alpha_i \leftarrow$ The set of columns that cover row i, $i \in I$.
4: $\beta_j \leftarrow$ The set of rows covered by column j, $j \in J$.
5: $N \leftarrow$ The set of N-columns in a solution.
6: $w_i \leftarrow$ The number of columns that cover row i, $i \in I$. For this, $w_i \leftarrow |N \cap \alpha_i|$, $\forall\, i \in I$
7: $U \leftarrow$ The set of uncovered rows. For this, $U = \{i \in I \mid w_i = 0\}$
8: **while** row $i \in U$ (in increasing order of i) **do**
9: Find the first column j in increasing order of $j \in \alpha_i$ that minimizes $\dfrac{c_j}{|U \cap \beta_j|}$
10: Add j to N, and set $w_i \leftarrow w_i + 1$, $\forall\, i \in \beta_j$
11: Set $U \leftarrow U - \beta_j$
12: **end while**
13: **while** column $j \in N$ (in decreasing order of j) **do**
14: **for all** row $i \in \beta_j$ **do**
15: **if and only if** $w_i \geq 2$ **then**
16: $N \leftarrow N - j$
17: $w_i \leftarrow w_i - 1$
18: **end if**
19: **end for**
20: **end while**
21: **return** the feasible solution H.

4.4. Adaptive mutation rate

In BBOA, the maximum mutation rate is very influential on the quality of the solutions. This mutation scheme tends to enhance the diversity among the population, which helps to decrease the chance of becoming trapped in local optima. For this, the value of this parameter is a low number (approximately 0.0005 to 0.004). In the convergence of the BBOA, solutions generally stagnate in a local optimum, losing valuable iterations. When this happens, we implement a method that increases the maximum mutation rate, adding diversity and avoiding long stagnation.

The maximum rate of mutation should be increased allowing for new solutions to be obtained when there is stagnation. For this, a percentage of 10% of deadlock over the missing iterations is calculated. If this is true, the maximum mutation rate is increased by 0.0009 over the previous rate. Then, if the percentage of stagnation continues to increase up to 20%, the rate increases again so that the local optimum changes. By applying this method, the maximum mutation rate, which is a parameter BBOA, becomes variable. This method is a variation of the BBOA algorithm that we call SA-BBOA and that was discovered through experimentation; we note the improvements in the results.

4.5. Binary approaches

Set covering is a problem whose domain is limited to binary values, namely, $H_i^j \in \{0, 1\}$, $\forall\, j \in \{1, \ldots, N\}$. Therefore, in this paper, we used a binary representation for each habitat-candidate solution.

TABLE 3. $\mathcal{S}$-Shape and $\mathcal{V}$-Shape transfer functions.

$\mathcal{S}$-Shape		$\mathcal{V}$-Shape	
$\mathcal{S}_1$:		$\mathcal{V}_1$:	
	$g(x_i^j) = \frac{1}{1+e^{-2x_i^j}}$		$g(x_i^j) = \left\lvert \operatorname{erf}\left(\frac{\sqrt{\pi}}{2}x_i^j\right)\right\rvert$
$\mathcal{S}_2$:		$\mathcal{V}_2$:	
	$g(x_i^j) = \frac{1}{1+e^{2x_i^j}}$		$g(x_i^j) = \lvert \tan h\left(x_i^j\right)\rvert$
$\mathcal{S}_3$:		$\mathcal{V}_3$:	
	$g(x_i^j) = \frac{1}{1+e^{\frac{-x_i^j}{2}}}$		$g(x_i^j) = \left\lvert \frac{x_i^j}{\sqrt{1+[x_i^j]^2}}\right\rvert$
$\mathcal{S}_4$:		$\mathcal{V}_4$:	
	$g(x_i^j) = \frac{1}{1+e^{\frac{-x_i^j}{3}}}$		$g(x_i^j) = \left\lvert \frac{2}{\pi}\arctan\left(\frac{\pi}{2}x_i^j\right)\right\rvert$

The standard version of the biogeography-based optimization algorithm is designed to solve problems with real domains. This task is resolved by transforming domains by applying binarization strategies, which are responsible for forcing elements to move in a binary domain. The binarization strategy is composed of a transfer function and a discretization method.

We evaluate different functions separated into two families [8,33]: $\mathcal{S}$-Shape and $\mathcal{V}$-Shape (see Tab. 3). Independent of the generated value H_i^j, the function $g(H_i^j)$ is always in a real domain between 0 and 1 [8], as shown in Figure 4.

Once a transfer function is applied, the input real number is mapped to a real number belonging to a $[0,1]$ interval. Then, a discretization method is required to produce a binary value from the real value. To achieve this, we test four different methods:

(1) Standard: If the condition is satisfied, the standard method returns 1; otherwise, it returns 0

$$H_i^j = \begin{cases} 1, & \text{if } r \text{ and } \leq g(H_i^j) \\ 0, & \text{otherwise} \end{cases}. \tag{4.3}$$

(2) Complement: If the condition is satisfied, the standard method returns the complement value

$$H_i^j = \begin{cases} \overline{H_i^j}, & \text{if } r \text{ and } \leq g(H_i^j) \\ 0, & \text{otherwise} \end{cases}. \tag{4.4}$$

(3) Static probability: A probability is generated and evaluated with a transfer function

$$H_i^j = \begin{cases} 0, & \text{if } g(H_i^j) \leq \alpha \\ H_i^j, & \text{if } \alpha < g(H_i^j) \leq \frac{1}{2}(1+\alpha) \\ 1, & \text{if } g(H_i^j) \geq \frac{1}{2}(1+\alpha) \end{cases}. \tag{4.5}$$

(4) Elitist: The discretization method elitist roulette, also known as Monte Carlo, randomly selects among the best individuals of the population, with a probability proportional to its fitness

$$H_i^j = \begin{cases} H_i^j, & \text{if } r \text{ and } \leq g(H_i^j) \\ 0, & \text{otherwise} \end{cases}. \tag{4.6}$$

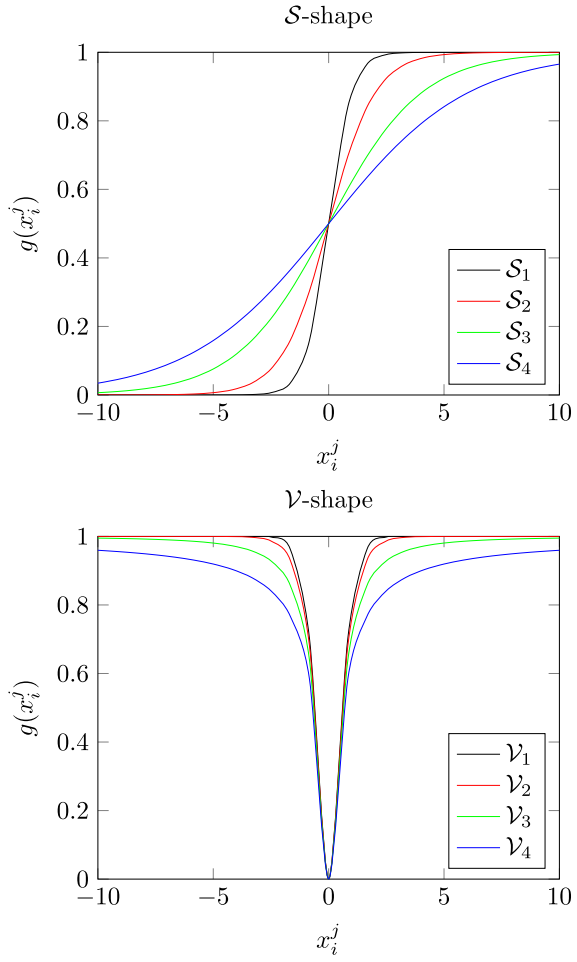

FIGURE 4. Behavior of the $\mathcal{S}$-Shape and the $\mathcal{V}$-Shape functions.

In the sample phase, we determine that the binarization strategy that achieves the best results is $\mathcal{S}_2 + Standard$.

Finally, we incorporate the pseudocode of SA-BBOA (see Algorithm 4) solving the set covering problem. Inputs to the procedure are the value of the population size $popSize$, the number of maximum iterations T, $m(\max)$, I, and E.

In the first step on Line 2, the SCP instance is loaded. Then, in Lines 4–9, a loop statement is presented. These instructions allow for the generation of the initial solutions (random habitats) and determine the best solution according to its performance. This performance is given by the cost of the solution in the objective function.

Then, while a termination criterion (a maximum number of iterations or a sufficiently good solution was not reached) is met, each fitness of a potential solution is evaluated (Lines 13–50). As previously mentioned, the set covering problem is a minimization problem. This evaluation is handled by a comparison presented at Line 13. If the new min value is less than the min global, the min global is changed by the new min value, and the best solution is stored in $\hat{H}$.

Then, the mutation rate $m(species)$ is calculated according to equation (3.1). We use this probability for selecting an SIV (jth component) of a habitat H_i. The SIV component is replaced with a randomly generated

value. Next, in Lines 30–36, a solution H_i is selected with a probability λ_i to select a new SIV $k \in [1, \ldots, N]$. If $\mu_i > Random[0, 1]$, then a component $k \in [1, \ldots, N]$ is randomly selected, and it is copied as $H_i^j = H_i^k$.

Algorithm 4 Self-adaptive biogeography-based optimization algorithm.

Require: Problem input data, $popSize, T, m(\max), I, E$.
Ensure: The best solution that resolves the set covering problem.
1: {N is the solution length, and c_j is the cost vector, $1 \leq j \leq N$.}
2: $\{N, c_j\} \leftarrow loadProblemData()$
3: {Produce the first generation of $popSize$ habitat.}
4: **for all** habitat $H_i, (\forall\, i = 1, \ldots, popSize)$ **do**
5: **for all** SIV j, $(\forall\, j = 1, \ldots, N)$ **do**
6: $H_i^j \leftarrow Random\{0, 1\}$
7: **end for**
8: $dofeasible(H_i)$
9: $HSI_i \leftarrow \dfrac{1}{\sum_{j=1}^{n} c_j H_i^j}$
10: **end for**
11: $globalfit \leftarrow +\infty$
12: {Produce T-generations of $popSize$ habitat.}
13: **while** $t < T$ **do**
14: $\{minfit, minindex\} \leftarrow min(HSI)$
15: **if** $minfit < globalfit$ **then**
16: $globalfit \leftarrow minfit$
17: **for all** SIV j, $(\forall\, j = 1, \ldots, N)$ **do**
18: $\hat{H}^j(t) \leftarrow H^j_{minindex}(t)$
19: **end for**
20: **end if**
21: **for all** habitat $H_i, (\forall\, i = 1, \ldots, popSize)$ **do**
22: **for all** SIV j, $(\forall\, j = 1, \ldots, N)$ **do**
23: Calculate mutation rate $m(j)$ based on equation (3.1).
24: Select SIV from H_i with probability $m(j)$.
25: **if** $m(j) > Random[0, 1]$ **then**
26: $\{H_i^j$ is selected.$\}$
27: Replace H_i^j with a randomly generated SIV.
28: **end if**
29: **end for**
30: $dofeasible(H_i)$
31: Select H_i with probability λ_i.
32: {N is the solution length.}
33: **for all** SIV j, $(\forall\, j = 1, \ldots, N)$ **do**
34: Select H_i^j with a probability μ_i.
35: **if** $\mu_i > Random[0, 1]$ **then**
36: $\{H_i^j$ is selected.$\}$
37: Select randomly a component $k \in [1, \ldots, N]$.
38: Set $H_i^j = H_i^k$.
39: **end if**
40: **end for**
41: $dofeasible(H_i)$
42: {Adaptive mutation rate.}
43: **if** $HSI_i < HSI_{minindex}$ **then**
44: $HSI_{minindex} \leftarrow HSI_i$
45: $miss_{it} \leftarrow 0$
46: **else**
47: $miss_{it} \leftarrow miss_{it} + 1$
48: {Missing iterations.}
49: **if** $miss_{it} > T \times 10\%$ **then**
50: **for all** SIV j, $(\forall\, j = 1, \ldots, N)$ **do**
51: $m(j) \leftarrow m(j) + m(j) \times 0.0009$
52: **end for**
53: $miss_{it} \leftarrow 0$
54: **end if**
55: **end if**
56: $t \leftarrow t + 1$
57: **end for**
58: **end while**
59: **return** Postprocess results and visualization.;

At the end of the SA-BBOA, we present the adaptive mutation rate method. If the solutions do not improve in a period of time (missing iterations), the maximum mutation rate is increased by 0.0009 over the rate. This process runs until the local optimum changes.

5. Experiments and results

For the experiments, BBOA and the new approach SA-BBOA were implemented in the Java programming language. The experiments were carried out on a Windows 8.1 operating system, with an Intel Core i3 2.50 GHz processor with 6 GB of RAM. For both algorithms, the parameter values used were: $popSize = 15$, $m(\max) = 0.004$, $I = 1$, $E = 1$ and the maximum iterations = 6000. Each instance was executed 30 times. Moreover, we used preprocessed instances for SCP, obtained from the OR-Library [5]. Table 4 describes the group of instance sets, the number of rows or constraints (M), the number of columns or variables (N), the range of costs, the density (percentage of nonzeroes in the matrix) and whether the optimal solution is known or unknown.

The results are evaluated using the relative percentage deviation (RPD). The RPD value quantifies the deviation of the objective value $Z_{\min}$ from Z_{opt}, which is the minimal best-known value for each instance in our experiment, and it is calculated as follows:

$$\text{RPD} = \left(\frac{Z_{\min} - Z_{\text{opt}}}{Z_{\text{opt}}} \right). \tag{5.1}$$

5.1. Biogeography algorithms comparison

In this section, we compare the proposed self-adaptive biogeography optimization algorithm *vs.* the basic algorithm. Tables 5 and 6 illustrate the results obtained for the instances from groups 4 and 5, and 6 to C, respectively. Table 7 details the results obtained for instances from groups NRE, NRF, and NRG. Finally, results of the group NRH are exposed in Table 8.

Red-bold font emphasizes the cases in which the self-adaptive biogeography optimization algorithm outperformed the original version.

Regarding the instance sets between 4 and C, the self-adaptive approach shows a high performance for reaching new optimum values: for group 4, the new values are in instances 4.1, 4.5, 4.7, 4.9, 4.10; in group 5 the

Table 4. SCP instances taken from the Beasley's OR-Library.

Instance group	M	N	Cost range	Density (%)	Best known
4	200	1000	[1,100]	2	Known
5	200	2000	[1,100]	2	Known
6	200	1000	[1,100]	5	Known
A	300	3000	[1,100]	2	Known
B	300	3000	[1,100]	5	Known
C	400	4000	[1,100]	2	Known
D	400	4000	[1,100]	5	Known
NRE	500	5000	[1,100]	10	Unknown (except **NRE.1**)
NRF	500	5000	[1,100]	20	Unknown (except **NRF.1**)
NRG	1000	10000	[1,100]	2	Unknown (except **NRG.1**)
NRH	1000	10000	[1,100]	5	Unknown

TABLE 5. Computational results of groups 4 and 5.

Instance	Z_{opt}	Biogeography-based optimization algorithm					
		Basic			BBOA-AS		
		Z_{min}	RPD	Times (ms)	Z_{min}	RPD	Times (ms)
4.1	429	430	0.002	811.1	**429**	**0.00**	981.8
4.2	512	512	0.000	917.0	512	0.000	967.7
4.3	516	516	0.000	921.8	516	0.000	956.3
4.4	494	495	0.002	887.3	495	0.002	989.2
4.5	512	514	0.004	920.6	**512**	**0.000**	998.2
4.6	560	560	0.000	934.8	560	0.000	997.2
4.7	430	431	0.002	994.4	**430**	**0.000**	982.4
4.8	492	492	0.000	991.5	492	0.000	899.1
4.9	641	644	0.005	899.6	**643**	**0.003**	989.2
4.10	514	515	0.002	982.4	**514**	**0.000**	932.2
5.1	253	253	0.000	1714.7	**253**	**0.000**	1887.4
5.2	302	305	0.010	1202.3	**302**	**0.000**	1988.6
5.3	226	228	0.010	1588.3	**226**	**0.000**	2001.4
5.4	242	242	0.000	1378.0	242	0.000	2089.2
5.5	211	211	0.000	1461.3	211	0.000	2021.5
5.6	213	214	0.002	1351.8	**213**	**0.000**	2111.6
5.7	293	293	0.000	1542.3	293	0.000	2128.2
5.8	288	289	0.010	1644.9	**288**	**0.000**	2123.2
5.9	279	281	0.020	1454.8	**279**	**0.000**	2432.2
5.10	265	265	0.000	1583.7	265	0.000	2153.4

new values are in instances 5.1, 5.2, 5.3, 5.6, 5.8, and 5.9; and in group 6, the optimum values are achieved for both algorithms.

For the instance group A, we can see that the SA-BBOA reaches the one-only optimum values that the basic algorithm cannot find. Only considering the instance set B, we can observe that the SA-BBOA exhibits clear robustness to find the same optimal value as the BBOA. If we compare the instance groups C, we see again that SA-BBOA obtains better results than BBOA. Finally, if we analyze the instance groups D, we can determine that not exists a difference between the two approaches.

Finally, if we analyze the most difficult instances NRE, NRF, NRG, and NRH (see Tabs. 7 and 8), we can see that again the BBOA shows an evident inefficiency for solving the set covering problem since it reaches only 3 of 20 optimal values. Nevertheless, the self-adaptive approach shows that we can further improve the BBO algorithm performance by finding four better values and six new values.

If we focus on the time required for reaching the solutions, we may observe that times are very similar for the two algorithms. However, we must consider that SA-BBOA needs the computation of the adaptive process and is able to outperform the basic BBOA in terms of the optimum values reached. We can also observe a small difference in terms of solving times in favor of the SA-BBOA with respect to BBOA.

Figures 5 and 6 illustrate the convergence charts for the most difficult instances of the test groups 4 to NRH. Here, we observe that for group 4, the convergence of the SA-BBOA is clearly faster than the others. For group 5, the SA-BBOA begins with a bad quality solution but improves its performance in the middle of the process outperforming the basic BBOA. The performance of the instances from groups 6, A, B, C, and D are similar, all of them achieving an early convergence.

Convergence is similar in the NRE and NRF group. In both cases, SA-BBOA achieves better performance. Finally, for the benchmarks from groups NRG and NRH, the behavior of SA-BBOA is clearly earlier than its competitor.

TABLE 6. Computational results of groups 6, A, B, and C.

Instance	Z_{opt}	Biogeography-based optimization algorithm					
		Basic			BBOA-AS		
		Z_{min}	RPD	Times (ms)	Z_{min}	RPD	Times (ms)
6.1	138	138	0.000	2545.5	138	0.000	2953.4
6.2	146	146	0.000	2802.1	146	0.000	2961.7
6.3	145	145	0.000	2798.5	145	0.000	2789.0
6.4	131	131	0.000	2817.3	131	0.000	2993.7
6.5	161	161	0.000	2855.6	161	0.000	2987.4
A.1	253	254	0.002	2989.5	**253**	**0.000**	3097.1
A.2	252	252	0.000	2988.3	252	0.000	3083.2
A.3	232	232	0.000	2998.2	232	0.000	3007.3
A.4	234	234	0.000	2991.8	234	0.000	3153.8
A.5	236	236	0.000	2989.3	236	0.000	3125.4
B.1	69	69	0.000	3203.5	69	0.000	3242.1
B.2	76	76	0.000	3113.3	76	0.000	3413.6
B.3	82	80	0.003	3523.3	**80**	**0.000**	3443.3
B.4	79	79	0.000	3142.1	79	0.000	3594.3
B.5	72	73	0.001	333.3	**72**	**0.000**	3443.4
C.1	227	229	0.009	3751.1	**227**	**0.000**	3957.5
C.2	219	219	0.000	3960.4	219	0.000	3991.4
C.3	243	245	0.008	3974.6	245	0.008	3943.4
C.4	219	219	0.000	3830.2	219	0.000	3933.9
C.5	215	215	0.000	3879.6	215	0.000	3914.6

TABLE 7. Computational results of groups D, NRE, NRF, and NRG.

Instance	Z_{opt}	Biogeography-based optimization algorithm					
		Basic			BBOA-AS		
		Z_{min}	RPD	Times (ms)	Z_{min}	RPD	Times (ms)
D.1	60	60	0.000	5828.3	60	0.000	6431.8
D.2	66	67	0.015	5990.2	67	0.015	6454.5
D.3	72	73	0.014	5971.2	73	0.014	6746.7
D.4	62	62	0.000	5903.4	62	0.000	6716.3
D.5	61	61	0.000	5891.4	61	0.000	6617.2
NRE.1	29	30	0.034	6200.0	**29**	**0.000**	6135.0
NRE.2	30	31	0.033	6235.4	31	0.033	6115.1
NRE.3	27	28	0.037	5988.3	28	0.037	5991.9
NRE.4	28	29	0.036	6875.2	**28**	**0.000**	7081.0
NRE.5	28	28	0.000	7298.1	28	0.000	7191.1
NRF.1	14	14	0.000	6788.4	14	0.000	7118.5
NRF.2	15	15	0.000	8313.4	15	0.000	8212.1
NRF.3	14	17	0.214	8934.4	**16**	**0.143**	9293.8
NRF.4	14	16	0.143	7746.3	14	0.000	8023.3
NRF.5	13	14	0.077	7021.1	**13**	**0.000**	7220.2
NRG.1	176	179	0.017	7143.5	179	0.017	7644.8
NRG.2	154	158	0.026	7948.0	158	0.026	8584.5
NRG.3	166	170	0.024	8771.8	**166**	**0.000**	9874.0
NRG.4	168	170	0.012	8123.7	**168**	**0.000**	8919.2
NRG.5	168	170	0.012	9013.4	**168**	**0.000**	9921.2

TABLE 8. Computational results of the group NRH.

Instance	Z_{opt}	Biogeography-based optimization algorithm					
		Basic			BBOA-AS		
		Z_{min}	RPD	Times (ms)	Z_{min}	RPD	Times (ms)
NRH.1	63	66	0.048	15114.6	**64**	**0.015**	17911.2
NRH.2	63	67	0.063	16343.4	67	0.063	19733.4
NRH.3	59	64	0.085	19243.7	**64**	**0.085**	21840.3
NRH.4	58	64	0.103	18241.6	**63**	**0.086**	21421.7
NRH.5	55	63	0.145	24442.6	63	0.145	22242.5

5.2. Statistical test

To show a significant difference between the basic and self-adaptive approach for the biogeography-based optimization algorithm, we perform a contrast statistical test for each instance through the *Kolmogorov-Smirnov-Lilliefors* to determine the independence of the samples [28], and *Wilcoxon's signed rank* [31] to compares the results statistically.

For both tests, we consider a hypothesis evaluation, which is analyzed assuming a *p_value* of 0.05, *i.e.*, values smaller than 0.05 determine that the corresponding hypothesis cannot be assumed. Both tests were conducted using *GNU Octave*[6]. The first test allows us to analyze the independence of samples by determining whether the Z_{min} results from the 30 executions of each instance are from a normal distribution or they are independent. To proceed, we propose the following hypotheses: H_0 states that Z_{min} follows a normal distribution. H_1 states the opposite. The conducted test yielded a *p_value* lower than 0.05; therefore, H_0 cannot be assumed. Next, as the samples are independent and cannot be assumed to follow a normal distribution, it is not feasible to use the central limit theorem to approximate the distribution of the sample mean as Gaussian. Therefore, we assume the use of a nonparametric test for evaluating the heterogeneity of samples. For that, we use the *Wilcoxon's signed rank* test. This is a paired test that compares the medians of two distributions. To proceed, we propose the following new hypotheses: H_0: $\tilde{Z}_{min}$ achieved by basic BBOA is better than $\tilde{Z}_{min}$ achieved by SA-BBOA. H_1 states the opposite.

Tables 9–11 compare the basic biogeography-based optimization algorithm *vs.* its self-adaptive approach for all tested instances via the *Wilcoxon's signed rank* test. As the significance level is also established to 0.05, smaller values than 0.05 define that H_0 cannot be assumed. Bold font is used for a winner value of the metaheuristic stated in the column of the table, *e.g.*, for instance 4.1, the self-adaptive version is better than the basic version as its value is lower than 0.05, and then, H_0 cannot be assumed.

According to the results, those for *p_values* lower than 0.05 for the basic biogeography-based optimization algorithm are 9; the results for the self-adaptive approach are 32. The remainder does not provide significant information. These results illustrate again that the performance of the self-adaptive approach is better than basic BBOA.

5.3. BBOA-AS *vs.* other optimization techniques

To evidence the performance of our self-adaptive approach, we perform a comparison with different approximation techniques: binary cat swarm optimization (BCSO) [10], binary firefly optimization (BFO) [16], binary shuffled frog leaping algorithm (BSFLA) [17], binary artificial bee colony algorithm (BABC) [19], and binary electromagnetism-like algorithm (BELA) [41]. It will additionally incorporate a comparative using Mixed Integer Linear Programming (MIP) as exact solving method implemented on *MiniZinc G12 MIP*. With the solver,

[6]Available at https://www.gnu.org/software/octave/download.html

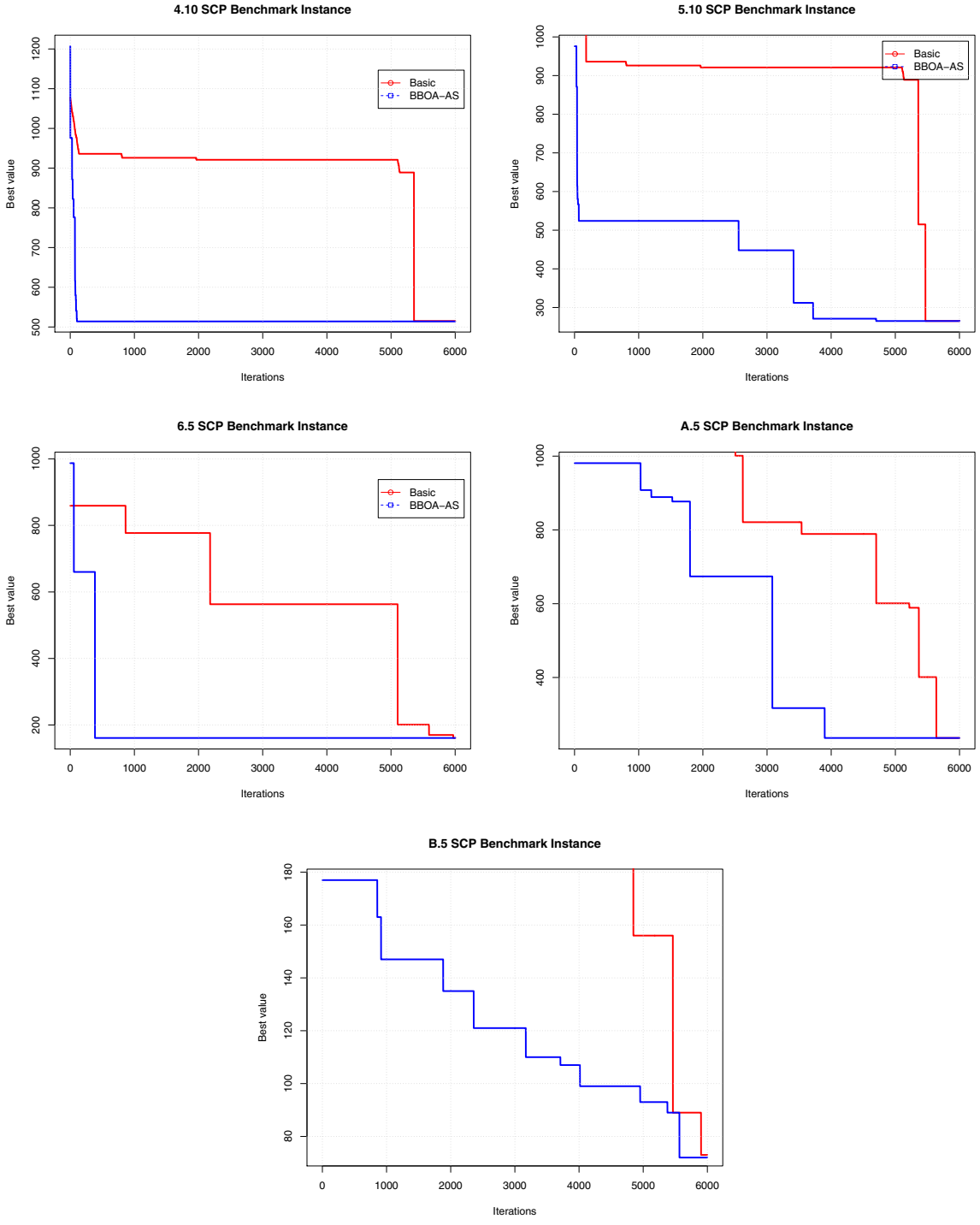

FIGURE 5. Convergence charts for the most difficult instances of the test groups 4, 5, 6, A, and B.

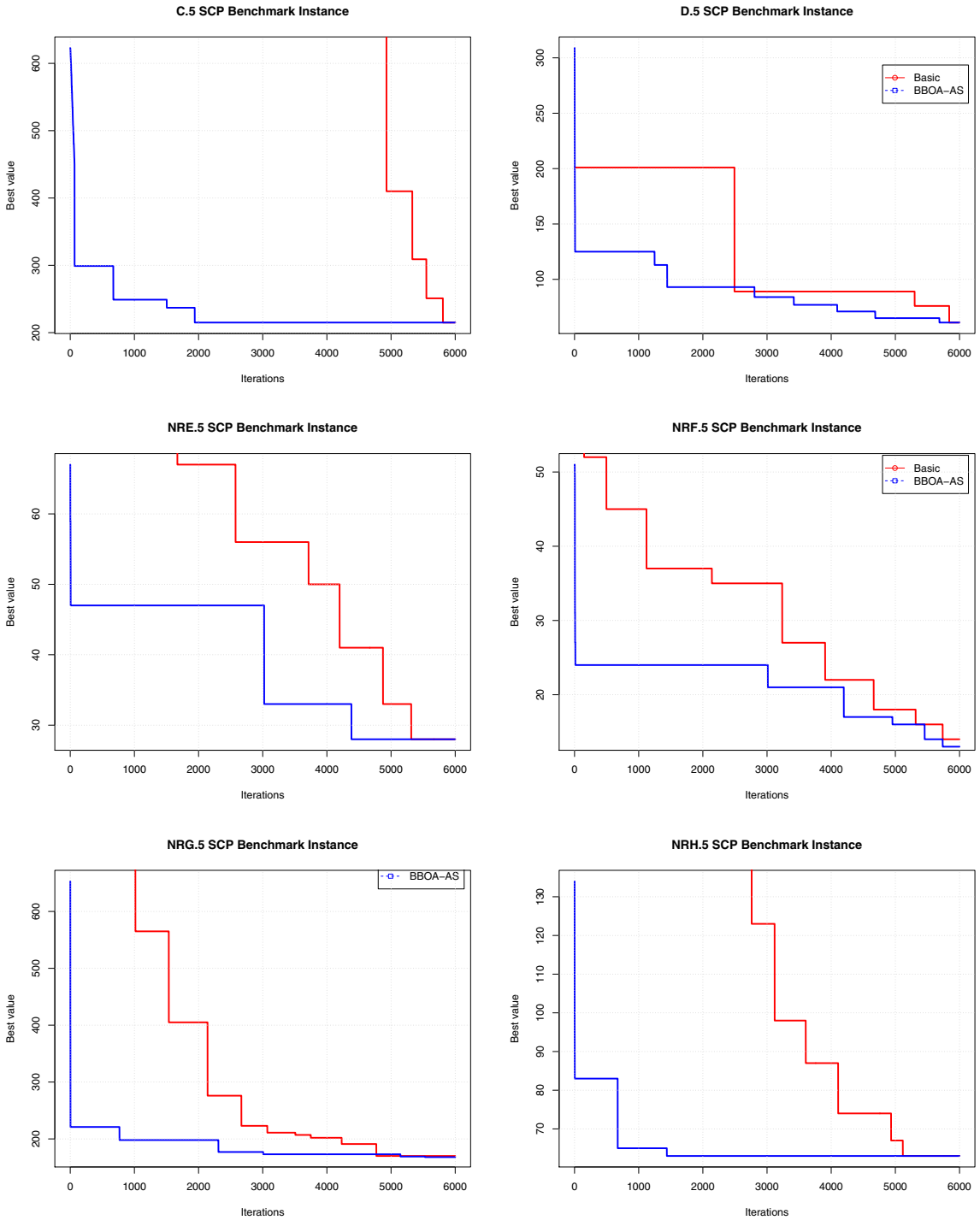

FIGURE 6. Convergence charts for the most difficult instances of the test groups: C, D, NRE, NRF, NRG, and NRG.

TABLE 9. Statistical test: instances of groups 4, 5 and 6.

Instance	4.1	4.2	4.3	4.4	4.5
Basic	**0.04**	–	**0.04**	–	–
BBOA-AS	–	**0.02**	–	**0.03**	**0.03**
Instance	4.6	4.7	4.8	4.9	4.10
Basic	–	–	–	–	–
BBOA-AS	**0.02**	**0.02**	–	**0.05**	–
Instance	5.1	5.2	5.3	5.4	5.5
Basic	**0.05**	–	–	–	–
BBOA-AS	–	–	**0.02**	**0.03**	**0.01**
Instance	5.6	5.7	5.8	5.9	5.10
Basic	–	–	–	–	–
BBOA-AS	**0.01**	–	**0.01**	**0.01**	**0.03**
Instance	6.1	6.2	6.3	6.4	6.5
Basic	–	–	–	–	–
BBOA-AS	–	–	–	–	–

TABLE 10. Statistical test: instances of groups A, B, C, and D.

Instance	A.1	A.2	A.3	A.4	A.5
Basic	–	**0.05**	–	–	–
BBOA-AS	–	–	**0.05**	**0.01**	**0.02**
Instance	B.1	B.2	B.3	B.4	B.5
Basic	–	**0.01**	–	**0.02**	–
BBOA-AS	**0.02**	–	**0.01**	–	**0.02**
Instance	C.1	C.2	C.3	C.4	C.5
Basic	**0.05**	–	–	–	–
BBOA-AS	–	–	–	–	–
Instance	D.1	D.2	D.3	D.4	D.5
Basic	–	**0.01**	**0.01**	–	–
BBOA-AS	–	–	–	**0.04**	–

TABLE 11. Statistical test: instances of groups NRE, NRF, NRG, and NRH.

Instance	NRE.1	NRE.2	NRE.3	NRE.4	NRE.5
Basic	–	–	–	–	–
BBOA-AS	**0.05**	–	–	**0.03**	**0.01**
Instance	NRF.1	NRF.2	NRF.3	NRF.4	NRF.5
Basic	**0.02**	**0.03**	**0.04**	–	–
BBOA-AS	–	–	**0.01**	–	**0.04**
Instance	NRG.1	NRG.2	NRG.3	NRG.4	NRG.5
Basic	–	–	–	–	–
BBOA-AS	–	–	**0.02**	**0.02**	**0.01**
Instance	NRH.1	NRH.2	NRH.3	NRH.4	NRH.5
Basic	–	**0.01**	–	–	–
BBOA-AS	**0.01**	–	**0.02**	**0.01**	–

TABLE 12. Comparison results for instance set of groups 4 and 5. BBOA-AS v/s BCSO and BFO.

Instance	Z_{opt}	BBOA-AS			Optimization algorithms					
					BCSO			BFO		
		Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD
4.1	429	**429**	430	0.000	459	480	0.070	**429**	430	0.000
4.2	512	**512**	514	0.000	570	594	0.113	517	517	0.010
4.3	516	**516**	516	0.000	590	607	0.143	519	522	0.006
4.4	494	**494**	502	0.000	547	578	0.107	495	497	0.002
4.5	512	**512**	515	0.000	545	554	0.064	514	515	0.004
4.6	560	**560**	561	0.000	637	650	0.138	563	565	0.005
4.7	430	**430**	430	0.000	462	467	0.074	**430**	430	0.000
4.8	492	**492**	497	0.000	546	567	0.110	497	499	0.010
4.9	641	**641**	645	0.000	711	725	0.109	655	658	0.022
4.10	514	**514**	516	0.000	537	552	0.045	519	523	0.010
5.1	253	**253**	253	0.000	279	287	0.103	257	260	0.016
5.2	302	**302**	307	0.000	339	340	0.123	309	311	0.023
5.3	226	**226**	229	0.000	247	251	0.093	229	233	0.013
5.4	242	**242**	242	0.000	251	253	0.037	**242**	242	0.000
5.5	211	**211**	213	0.000	230	230	0.090	**211**	213	0.000
5.6	213	**213**	213	0.000	232	243	0.089	**213**	213	0.000
5.7	293	**293**	293	0.000	332	338	0.133	298	301	0.017
5.8	288	**288**	288	0.000	320	330	0.111	291	292	0.010
5.9	279	**279**	281	0.000	295	297	0.057	284	284	0.018
5.10	265	**265**	267	0.000	285	287	0.075	268	270	0.011

TABLE 13. Comparison results for instance set of groups 6, A, B, and C. BBOA-AS v/s BCSO and BFO.

Instance	Z_{opt}	BBOA-AS			Optimization algorithms					
					BCSO			BFO		
		Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD
6.1	138	**138**	138	0.000	151	160	0.094	**138**	140	0.000
6.2	146	**146**	146	0.000	152	157	0.041	147	149	0.007
6.3	145	**145**	148	0.000	160	164	0.103	147	150	0.014
6.4	131	**131**	132	0.000	138	142	0.053	**131**	131	0.000
6.5	161	**161**	163	0.000	169	173	0.050	164	157	0.019
A.1	253	**253**	253	0.000	286	287	0.130	255	256	0.008
A.2	252	**252**	253	0.000	274	276	0.087	259	261	0.028
A.3	232	**232**	234	0.000	257	263	0.108	238	240	0.026
A.4	234	**234**	239	0.000	248	251	0.060	235	237	0.004
A.5	236	**236**	241	0.000	244	244	0.034	**236**	237	0.000
B.1	69	**69**	73	0.000	79	79	0.145	71	72	0.029
B.2	76	**76**	79	0.000	86	89	0.132	78	78	0.026
B.3	80	**80**	84	0.000	85	85	0.063	**80**	80	0.000
B.4	79	**79**	83	0.000	89	89	0.127	80	81	0.013
B.5	72	**72**	72	0.000	73	73	0.014	**72**	73	0.000
C.1	227	**227**	229	0.000	242	242	0.066	230	232	0.013
C.2	219	**219**	219	0.000	240	241	0.096	223	224	0.018
C.3	243	**244**	248	0.004	277	278	0.140	253	254	0.041
C.4	219	**219**	221	0.000	250	250	0.142	225	227	0.027
C.5	215	**215**	217	0.000	243	244	0.130	217	219	0.009

TABLE 14. Comparison results for instance set of groups D, NRE, NRF, and NRG. BBOA-AS v/s BCSO and BFO.

Instance	Z_{opt}	BBOA-AS			Optimization algorithms					
					BCSO			BFO		
		Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD
D.1	60	**60**	63	0.000	65	66	0.083	**60**	61	0.000
D.2	66	**66**	69	0.000	70	70	0.061	68	68	0.030
D.3	72	**72**	77	0.000	79	81	0.097	75	77	0.042
D.4	62	**62**	62	0.000	64	67	0.032	**62**	62	0.000
D.5	61	**61**	61	0.000	65	66	0.066	63	63	0.033
NRE.1	29	**29**	29	0.000	**29**	30	0.000	**29**	31	0.000
NRE.2	30	**30**	31	0.000	34	34	0.133	32	32	0.067
NRE.3	27	28	28	0.037	31	32	0.148	29	30	0.074
NRE.4	28	**28**	28	0.000	32	33	0.143	29	31	0.036
NRE.5	28	**28**	28	0.000	30	30	0.071	29	29	0.036
NRF.1	14	**14**	15	0.000	17	17	0.214	15	17	0.071
NRF.2	15	**15**	15	0.000	18	18	0.200	16	16	0.067
NRF.3	14	**14**	16	0.000	17	17	0.214	16	17	0.143
NRF.4	14	**14**	17	0.000	17	17	0.214	15	18	0.071
NRF.5	13	**13**	14	0.000	15	16	0.154	15	19	0.154
NRG.1	176	**176**	177	0.000	190	193	0.080	185	191	0.051
NRG.2	154	156	156	0.013	165	166	0.071	161	163	0.045
NRG.3	166	**166**	169	0.000	187	188	0.127	175	177	0.054
NRG.4	168	171	171	0.018	179	183	0.065	176	176	0.048
NRG.5	168	169	169	0.006	181	184	0.077	177	181	0.054

TABLE 15. Comparison results for instance set of the group NRH. BBOA-AS v/s BCSO and BFO.

Instance	Z_{opt}	BBOA-AS			Optimization algorithms					
					BCSO			BFO		
		Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD
NRH.1	63	**65**	65	0.032	70	71	0.111	69	70	0.095
NRH.2	63	67	67	0.063	67	67	0.063	**66**	66	0.048
NRH.3	59	**64**	65	0.085	68	70	0.153	65	67	0.102
NRH.4	58	**63**	63	0.086	66	67	0.138	**63**	65	0.086
NRH.5	55	62	62	0.127	61	62	0.109	**59**	60	0.073

the instances are solved to a maximum time of 8 h. If no solution is found at this point the problem is set to *time-out* (t.o.).

Tables 12–23 illustrate that the proposed approach can achieve competitive results in contrast to those modern optimization techniques.

For group 4, the adaptive approach shows outstanding behavior and achieves 100% of the total optimum values, while BFO only identifies two optimum values. BCSO exhibits poor performance by producing 0 optimal values. In the same group of instances, BBOA-AS is better than BELA, BSFLA, and BABC, even when BSFLA reaches 4 good results. On the other hand, MIP shows an excellent performance to identify all optimum values.

Considering the group, 5 we observe that the adaptive biogeography-based optimization algorithm can find all the optimum values. BSFLA is its closest competitor finding 4 optimal values. BFO follows it with 3 optimal values, and BABC with 2 reached optimal results. Finally, BCSO and BELA show that they are not able to solve the instances of the SCP.

TABLE 16. Comparison results for instance set of groups 4 and 5. BBOA-AS v/s BSFLA and BELA.

Instance	Z_{opt}	BBOA-AS			Optimization algorithms					
					BSFLA			BELA		
		Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD
4.1	429	**429**	430	0.000	430	430	0.002	447	448	0.042
4.2	512	**512**	514	0.000	516	518	0.008	559	559	0.092
4.3	516	**516**	516	0.000	520	520	0.008	537	539	0.041
4.4	494	**494**	502	0.000	501	504	0.014	527	530	0.067
4.5	512	**512**	515	0.000	514	514	0.004	527	529	0.029
4.6	560	**560**	561	0.000	563	563	0.005	607	608	0.084
4.7	430	**430**	430	0.000	431	432	0.002	448	449	0.042
4.8	492	**492**	497	0.000	497	499	0.010	509	512	0.035
4.9	641	**641**	645	0.000	656	656	0.023	682	682	0.064
4.10	514	**514**	516	0.000	518	519	0.008	571	571	0.111
5.1	253	**253**	253	0.000	254	255	0.004	280	281	0.107
5.2	302	**302**	307	0.000	307	307	0.017	318	321	0.053
5.3	226	**226**	229	0.000	228	230	0.009	242	240	0.071
5.4	242	**242**	242	0.000	**242**	242	0.000	251	252	0.037
5.5	211	**211**	213	0.000	**211**	213	0.000	225	227	0.066
5.6	213	**213**	213	0.000	**213**	214	0.000	247	248	0.160
5.7	293	**293**	293	0.000	297	299	0.014	316	317	0.078
5.8	288	**288**	288	0.000	291	293	0.010	315	317	0.094
5.9	279	**279**	281	0.000	281	283	0.007	314	315	0.125
5.10	265	**265**	267	0.000	**265**	266	0.000	280	282	0.057

TABLE 17. Comparison results for instance set of groups 6, A, B, and C. BBOA-AS v/s BSFLA and BELA.

Instance	Z_{opt}	BBOA-AS			Optimization algorithms					
					BSFLA			BELA		
		Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD
6.1	138	**138**	138	0.000	140	141	0.014	152	152	0.101
6.2	146	**146**	146	0.000	147	147	0.007	160	161	0.096
6.3	145	**145**	148	0.000	147	148	0.014	160	163	0.103
6.4	131	**131**	132	0.000	**131**	133	0.000	140	142	0.069
6.5	161	**161**	163	0.000	166	169	0.031	184	187	0.143
A.1	253	**253**	253	0.000	255	258	0.008	261	264	0.032
A.2	252	**252**	253	0.000	260	260	0.032	279	281	0.107
A.3	232	**232**	234	0.000	237	239	0.022	252	253	0.086
A.4	234	**234**	239	0.000	235	238	0.004	250	252	0.068
A.5	236	**236**	241	0.000	**236**	239	0.000	241	243	0.021
B.1	69	**69**	73	0.000	70	70	0.014	86	87	0.246
B.2	76	**76**	79	0.000	76	**77**	0.000	88	88	0.158
B.3	80	**80**	84	0.000	80	**80**	0.000	85	87	0.063
B.4	79	**79**	83	0.000	79	**80**	0.000	84	88	0.063
B.5	72	**72**	72	0.000	72	**73**	0.000	78	81	0.083
C.1	227	**227**	229	0.000	229	231	0.009	237	238	0.044
C.2	219	**219**	219	0.000	223	225	0.018	237	239	0.082
C.3	243	244	248	0.004	253	253	0.041	271	271	0.115
C.4	219	**219**	221	0.000	227	228	0.037	246	248	0.123
C.5	215	**215**	217	0.000	217	218	0.009	224	225	0.042

TABLE 18. Comparison results for instance set of groups D, NRE, NRF, and NRG. BBOA-AS v/s BSFLA and BELA.

Instance	Z_{opt}	BBOA-AS			Optimization algorithms					
					BSFLA			BELA		
		Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD
D.1	60	**60**	63	0.000	**60**	62	0.000	62	62	0.033
D.2	66	**66**	69	0.000	67	68	0.015	73	74	0.106
D.3	72	**72**	77	0.000	75	77	0.042	79	81	0.097
D.4	62	**62**	62	0.000	63	65	0.016	67	69	0.081
D.5	61	**61**	61	0.000	63	66	0.033	66	67	0.082
NRE.1	29	**29**	29	0.000	**29**	29	0.000	30	31	0.034
NRE.2	30	**30**	31	0.000	31	32	0.033	35	35	0.167
NRE.3	27	**28**	28	0.037	28	28	0.037	34	34	0.259
NRE.4	28	**28**	28	0.000	29	30	0.036	33	34	0.179
NRE.5	28	**28**	28	0.000	**28**	31	0.000	30	31	0.071
NRF.1	14	**14**	15	0.000	15	15	0.071	17	17	0.214
NRF.2	15	**15**	15	0.000	**15**	15	0.000	18	18	0.200
NRF.3	14	**14**	16	0.000	16	17	0.143	17	18	0.214
NRF.4	14	**14**	17	0.000	15	16	0.071	17	19	0.214
NRF.5	13	**13**	14	0.000	15	17	0.154	16	17	0.231
NRG.1	176	**176**	177	0.000	182	183	0.034	194	196	0.102
NRG.2	154	**156**	156	0.013	161	161	0.045	176	176	0.143
NRG.3	166	**166**	169	0.000	173	174	0.042	184	185	0.108
NRG.4	168	**171**	171	0.018	173	177	0.030	196	197	0.167
NRG.5	168	**169**	169	0.006	174	174	0.036	198	199	0.179

TABLE 19. Comparison results for instance set of the group NRH. BBOA-AS v/s BSFLA and BELA.

Instance	Z_{opt}	BBOA-AS			Optimization algorithms					
					BSFLA			BELA		
		Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD
NRH.1	63	**65**	65	0.032	68	69	0.079	70	71	0.111
NRH.2	63	67	67	0.063	**66**	66	0.048	71	71	0.127
NRH.3	59	64	65	0.085	**62**	63	0.051	68	70	0.153
NRH.4	58	**63**	63	0.086	**63**	64	0.086	70	72	0.207
NRH.5	55	62	62	0.127	**59**	61	0.073	69	69	0.255

For groups 6, A, B, and C, we note that the adaptive biogeography-based optimization algorithm can find all the optimum values, again. We can observe a high superiority compared to BCSO and BELA. BFO, BSFLA, and BABC are not rivals for our adaptive approach due to they only found 6 different optimal values and BBOA-As finds 95% of them (19 of 20). In this stage, MIP begins showing a bad performance being not able to solve form the A-group to head.

For the rest of the instances (hardest instances), the technique behaviors are similars. BBOA-AS follows showing excellent results. By comparing the optimal values, we can no doubt ensure that our approach is better than other metaheuristics.

TABLE 20. Comparison results for instance set of groups 4 and 5. BBOA-AS v/s BABC and MIP.

Instance	Z_{opt}	BBOA-AS			Optimization algorithms					
					BABC			MIP		
		Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD
4.1	429	**429**	430	0.000	430	430	0.002	**429**	429	0.000
4.2	512	**512**	514	0.000	513	513	0.002	**512**	512	0.000
4.3	516	**516**	516	0.000	519	521	0.006	**516**	516	0.000
4.4	494	**494**	502	0.000	495	496	0.002	**494**	494	0.000
4.5	512	**512**	515	0.000	514	517	0.004	**512**	512	0.000
4.6	560	**560**	561	0.000	561	565	0.002	**560**	560	0.000
4.7	430	**430**	430	0.000	431	434	0.002	**430**	430	0.000
4.8	492	**492**	497	0.000	493	494	0.002	**492**	492	0.000
4.9	641	**641**	645	0.000	649	651	0.012	**641**	641	0.000
4.10	514	**514**	516	0.000	517	519	0.006	**514**	514	0.000
5.1	253	**253**	253	0.000	254	255	0.004	**253**	253	0.000
5.2	302	**302**	307	0.000	309	309	0.023	**302**	302	0.000
5.3	226	**226**	229	0.000	229	233	0.013	**226**	226	0.000
5.4	242	**242**	242	0.000	**242**	245	0.000	**242**	242	0.000
5.5	211	**211**	213	0.000	**211**	212	0.000	**211**	211	0.000
5.6	213	**213**	213	0.000	214	214	0.005	**213**	213	0.000
5.7	293	**293**	293	0.000	298	301	0.017	**293**	293	0.000
5.8	288	**288**	288	0.000	289	291	0.003	**288**	288	0.000
5.9	279	**279**	281	0.000	280	281	0.004	**279**	279	0.000
5.10	265	**265**	267	0.000	267	270	0.008	**265**	265	0.000

TABLE 21. Comparison results for instance set of groups 6, A, B, and C. BBOA-AS v/s BABC and MIP.

Instance	Z_{opt}	BBOA-AS			Optimization algorithms					
					BABC			MIP		
		Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD
6.1	138	**138**	138	0.000	142	143	0.029	**138**	138	0.000
6.2	146	**146**	146	0.000	147	150	0.007	**146**	146	0.000
6.3	145	**145**	148	0.000	148	149	0.021	**145**	145	0.000
6.4	131	**131**	132	0.000	**131**	133	0.000	**131**	131	0.000
6.5	161	**161**	163	0.000	165	167	0.025	**161**	161	0.000
A.1	253	**253**	253	0.000	254	254	0.004		t.o.	
A.2	252	**252**	253	0.000	257	259	0.020		t.o.	
A.3	232	**232**	234	0.000	235	238	0.013		t.o.	
A.4	234	**234**	239	0.000	236	237	0.009		t.o.	
A.5	236	**236**	241	0.000	**236**	238	0.000		t.o.	
B.1	69	**69**	73	0.000	70	70	0.014		t.o.	
B.2	76	**76**	79	0.000	78	79	0.026		t.o.	
B.3	80	**80**	84	0.000	**80**	80	0.000		t.o.	
B.4	79	**79**	83	0.000	80	81	0.013		t.o.	
B.5	72	**72**	72	0.000	**72**	74	0.000		t.o.	
C.1	227	**227**	229	0.000	231	233	0.018		t.o.	
C.2	219	**219**	219	0.000	222	223	0.014		t.o.	
C.3	243	244	248	0.004	254	255	0.045		t.o.	
C.4	219	**219**	221	0.000	231	233	0.055		t.o.	
C.5	215	**215**	217	0.000	216	217	0.005		t.o.	

TABLE 22. Comparison results for instance set of groups D, NRE, NRF, and NRG. BBOA-AS v/s BABC and MIP.

Instance	Z_{opt}	BBOA-AS			Optimization algorithms					
					BABC			MIP		
		Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD
D.1	60	**60**	63	0.000	**60**	61	0.000		t.o.	
D.2	66	**66**	69	0.000	68	68	0.030		t.o.	
D.3	72	**72**	77	0.000	76	77	0.056		t.o.	
D.4	62	**62**	62	0.000	63	65	0.016		t.o.	
D.5	61	**61**	61	0.000	63	66	0.033		t.o.	
NRE.1	29	**29**	29	0.000	**29**	33	0.000		t.o.	
NRE.2	30	**30**	31	0.000	32	32	0.067		t.o.	
NRE.3	27	28	28	0.037	29	31	0.074		t.o.	
NRE.4	28	**28**	28	0.000	29	30	0.036		t.o.	
NRE.5	28	**28**	28	0.000	29	32	0.036		t.o.	
NRF.1	14	**14**	15	0.000	**14**	15	0.000		t.o.	
NRF.2	15	**15**	15	0.000	16	16	0.067		t.o.	
NRF.3	14	**14**	16	0.000	16	17	0.143		t.o.	
NRF.4	14	**14**	17	0.000	15	17	0.071		t.o.	
NRF.5	13	**13**	14	0.000	15	16	0.154		t.o.	
NRG.1	176	**176**	177	0.000	183	184	0.040		t.o.	
NRG.2	154	156	156	0.013	162	163	0.052		t.o.	
NRG.3	166	**166**	169	0.000	174	175	0.048		t.o.	
NRG.4	168	171	171	0.018	175	177	0.042		t.o.	
NRG.5	168	169	169	0.006	179	181	0.065		t.o.	

TABLE 23. Comparison results for instance set of the group NRH. BBOA-AS v/s BABC and MIP.

Instance	Z_{opt}	BBOA-AS			Optimization algorithms					
					BABC			MIP		
		Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD
NRH.1	63	**65**	65	0.032	70	71	0.111		t.o.	
NRH.2	63	**67**	67	0.063	69	72	0.095		t.o.	
NRH.3	59	**64**	65	0.085	66	67	0.119		t.o.	
NRH.4	58	**63**	63	0.086	64	64	0.103		t.o.	
NRH.5	55	62	62	0.127	**60**	61	0.091		t.o.	

6. CONCLUSION

In this paper, we presented a self-adaptive approach for a biogeography-based optimization algorithm to solve different instances of the set covering problem. This approach is based on online control for the mutation rate parameter, which is evaluated before the run of the metaheuristic. We added an effective preprocessing process to the core algorithm that allows for filtering and discarding values leading to unfeasible solutions. We also include a set of binarization strategies to adapt the biogeography algorithms to the binary domain. We tested 65 nonunit cost instances from the Beasley's OR-Library where several global optimum values, which were not reached using the basic biogeography-based optimization algorithm, were achieved via the self-adaptive approach. Both approaches were evaluated using a nonparametric statistical test, and the results are conclusive.

In future work, we plan to test self-adaptive approaches in recent bioinspired algorithms and to provide a larger comparison of techniques for solving the set covering problem. The integration of online parameter control

can lead the research toward new topics of study, such as dynamically selecting the best binarization solution strategy according to the performance indicators as analogously studied in [40,43,44].

Acknowledgements. Broderick Crawford is supported by Grant CONICYT/FONDECYT/REGULAR/1171243. Ricardo Soto is supported by Grant CONICYT/FONDECYT/REGULAR/1190129. Rodrigo Olivares is supported by CONICYT/FONDEF/IDeA/ID16I10449, STIC-AMSUD/17STIC-03, and FONDECYT/MEC/MEC80170097, and he is also Postgraduate Grant Pontificia Universidad Católica de Valparaíso (INF-PUCV 2015-2018). Finally, Gino Astorga and Enrique Cortés are supported by Postgraduate Grant Pontificia Universidad Católica de Valparaíso (INF-PUCV 2015-2018).

References

[1] S. Al-Shihabi, M. Arafeh and M. Barghash, An improved hybrid algorithm for the set covering problem. *Comput. Ind. Eng.* **85** (2015) 328–334.

[2] F. Amini and P. Ghaderi, Hybridization of harmony search and ant colony optimization for optimal locating of structural dampers. *App. Soft Comput.* **13** (2013) 2272–2280.

[3] E. Balas and M.C. Carrera, A dynamic subgradient-based branch-and-bound procedure for set covering. *Oper. Res.* **44** (1996) 875–890.

[4] J. Beasley, An algorithm for set covering problem. *Eur. J. Oper. Res.* **31** (1987) 85–93.

[5] J. Beasley and K. Jørnsten, Enhancing an algorithm for set covering problems. *Eur. J. Oper. Res.* **58** (1992) 293–300.

[6] A. Caprara, M. Fischetti, P. Toth, D. Vigo and P.L. Guida, Algorithms for railway crew management. *Math. Program.* **79** (1997) 125–141.

[7] M. Caserta, Tabu search-based metaheuristic algorithm for large-scale set covering problems. In: Metaheuristics. Springer Nature, Basingstoke (2007) 43–63.

[8] B. Crawford, R. Soto, G. Astorga, J. García, C. Castro and F. Paredes, Putting continuous metaheuristics to work in binary search spaces. *Complexity* **2017** (2017) 1–19.

[9] B. Crawford, R. Soto, N. Berríos, F. Johnson and F. Paredes, Solving the set covering problem with binary cat swarm optimization. In: Advances in Swarm and Computational Intelligence. Springer Nature, Basingstoke (2015) 41–48.

[10] B. Crawford, R. Soto, N. Berríos, F. Johnson, F. Paredes, C. Castro and E. Norero, A binary cat swarm optimization algorithm for the non-unicost set covering problem. *Math. Prob. Eng.* **2015** (2015) 1–8.

[11] B. Crawford, R. Soto, R. Cuesta and F. Paredes, Application of the artificial bee colony algorithm for solving the set covering problem. *Sci. World J.* **2014** (2014) 1–8.

[12] B. Crawford, R. Soto, E. Monfroy, W. Palma, C. Castro and F. Paredes, Parameter tuning of a choice-function based hyperheuristic using particle swarm optimization. *Expert Syst. App.* **40** (2013) 1690–1695.

[13] B. Crawford, R. Soto, E. Monfroy, F. Paredes and W. Palma, A hybrid ant algorithm for the set covering problem. *Int. J. Phys. Sci.* **6** (2011) 4667–4673.

[14] B. Crawford, R. Soto, C. Olea, F. Johnson and F. Paredes, Binary bat algorithms for the set covering problem. In: *2015 10th Iberian Conference on Information Systems and Technologies (CISTI)*. Institute of Electrical and Electronics Engineers (IEEE) (2015).

[15] B. Crawford, R. Soto, M. Olivares-Suárez and F. Paredes, A binary firefly algorithm for the set covering problem. In: Advances in Intelligent Systems and Computing. Springer Nature, Basingstoke (2014) 65–73.

[16] B. Crawford, R. Soto, M. Olivares-Suárez and F. Paredes, A binary firefly algorithm for the set covering problem. In: *3rd Computer Science On-line Conference 2014 (CSOC 2014)*. Vol. 285 of Advances in Intelligent Systems and Computing. Springer International Publishing, Cham (2014) 65–73.

[17] B. Crawford, R. Soto, C. Peña, W. Palma, F. Johnson and F. Paredes, Solving the set covering problem with a shuffled frog leapingalgorithm. In: Intelligent Information and Database Systems. Springer Nature, Basingstoke (2015) 41–50.

[18] B. Crawford, R. Soto, C. Torres-Rojas, C. Peña, M. Riquelme-Leiva, S. Misra, F. Johnson and F. Paredes, A binary fruit fly optimization algorithm to solve the set covering problem. In: Computational Science and Its Applications – *ICCSA 2015*. Springer Nature, Basingstoke (2015) 411–420.

[19] R. Cuesta, B. Crawford, R. Soto and F. Paredes, An artificial bee colony algorithm for the set covering problem. In: Advances in Intelligent Systems and Computing. Springer Nature, Basingstoke (2014) 53–63.

[20] M.L. Fisher and P. Kedia, Optimal solution of set covering/partitioning problems using dual heuristics. *Manage. Sci.* **36** (1990) 674–688.

[21] B.A. Foster and D.M. Ryan, An integer programming approach to the vehicle scheduling problem. *Oper. Res. Q. (1970–1977)* **27** (1976) 367.

[22] T.W. Francesca Rossi, P. VanBeek, Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier Science, Amsterdam (2006).

[23] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1979).

[24] L. Han, G. Kendall and P. Cowling, An adaptive length chromosome hyper-heuristic genetic algorithm for a trainer scheduling problem. In: Recent Advances in Simulated Evolution and Learning. World Scientific Pub Co Pte Lt (2004) 506–525.

[25] A. Jaramillo, B. Crawford, R. Soto, S. Misra, E. Olguín, Á.G. Rubio, J. Salas and S.M. Villablanca, An approach to solve the set covering problem with the soccer league competition algorithm. In: Computational Science and Its Applications – *ICCSA 2016*. Springer Nature, Basingstoke (2016) 373–385.

[26] G. Lan, G.W. DePuy and G.E. Whitehouse, An effective and simple heuristic for the set covering problem. *Eur. J. Oper. Res.* **176** (2007) 1387–1403.

[27] A.H. Land and A.G. Doig, An automatic method of solving discrete programming problems. *Econometrica* **28** (1960) 497.

[28] H. Lilliefors, On the kolmogorov-smirnov test for normality with mean and variance unknown. *J. Am. Stat. Assoc.* **62** (1967) 399–402.

[29] H. Ma and D. Simon, Biogeography-based optimization with blended migration for constrained optimization problems. In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. Association for Computing Machinery (ACM) (2010).

[30] H. Ma and D. Simon, Evolutionary Computation with Biogeography-based Optimization. Wiley-ISTE (2017).

[31] H. Mann and W. Donald, On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Stat.* **18** (1947) 50–60.

[32] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer Nature, Basingstoke (1996).

[33] S. Mirjalili and A. Lewis, S-shaped versus v-shaped transfer functions for binary particle swarm optimization. *Swarm Evol. Comput.* **9** (2013) 1–14.

[34] H. Mo and L. Xu, Biogeography migration algorithm for traveling salesman problem. *Int. J. Intel. Comput. Cybern.* **4** (2011) 311–330.

[35] D.N. Mudaliar and N.K. Modi, Unraveling travelling salesman problem by genetic algorithm using m-crossover operator. In: *2013 International Conference on Signal Processing, Image Processing & Pattern Recognition*. Institute of Electrical and Electronics Engineers IEEE (2013).

[36] M.G.R. Panos and M. Pardalos, Handbook of Applied Optimization. Oxford University Press, Oxford (2002).

[37] S.H.A. Rahmati and M. Zandieh, A new biogeography-based optimization (BBO) algorithm for the flexible job shop scheduling problem. *Int. J. Adv. Manuf. Technol.* **58** (2011) 1115–1129.

[38] Á.G. Rubio, B. Crawford, R. Soto, E. Olguín, S. Misra, A. Jaramillo, S.M. Villablanca and J. Salas, Solving the set covering problem with a binary black hole inspired algorithm. In: Computational Science and Its Applications – *ICCSA 2016*. Springer Nature, Basingstoke (2016) 207–219.

[39] B.M. Smith, Impacs – a bus crew scheduling system using integer programming. *Math. Program.* **42** (1988) 181–187.

[40] R. Soto, B. Crawford, S. Misra, W. Palma, E. Monfroy, C. Castro and F. Paredes, Choice functions for autonomous search in constraint programming: GA vs PSO. *Tech. Gazette* **20** (2013) 621–629.

[41] R. Soto, B. Crawford, A. Muñoz, F. Johnson and F. Paredes, Pre-processing, repairing and transfer functions can help binary electromagnetism-like algorithms. In: Advances in Intelligent Systems and Computing. Springer Nature, Basingstoke (2015) 89–97.

[42] R. Soto, B. Crawford, R. Olivares, J. Barraza, F. Johnson and F. Paredes, A binary cuckoo search algorithm for solving the set covering problem. In: *Lecture Notes in Computer Science*. Springer Nature, Basingstoke (2015) 88–97.

[43] R. Soto, B. Crawford, W. Palma, K. Galleguillos, C. Castro, E. Monfroy, F. Johnson and F. Paredes, Boosting autonomous search for CSPs via skylines. *Inf. Sci.* **308** (2015) 8–48.

[44] R. Soto, B. Crawford, W. Palma, E. Monfroy, R. Olivares, C. Castro and F. Paredes, Top-k based adaptive enumeration in constraint programming. *Math. Prob. Eng.* **2015** (2015) 1–12.

[45] R. Soto, B. Crawford, E. Vega and F. Paredes, Solving manufacturing cell design problems using an artificial fish swarm algorithm. In: *Lecture Notes in Computer Science*. Springer Nature, Basingstoke (2015) 282–290.

[46] S. Sundar and A. Singh, A hybrid heuristic for the set covering problem. *Oper. Res.* **12** (2010) 345–365.

[47] G.M. Thompson, A simulated-annealing heuristic for shift scheduling using non-continuously available employees. *Comput. Oper. Res.* **23** (1996) 275–288.

[48] C. Toregas, R. Swain, C. ReVelle and L. Bergman, The location of emergency service facilities. *Oper. Res.* **19** (1971) 1363–1373.

[49] F.J. Vasko, F.E. Wolf and K.L. Stott, A set covering approach to metallurgical grade assignment. *Eur. J. Oper. Res.* **38** (1989) 27–34.

[50] X. Zhang, Q. Kang, J. Cheng and X. Wang, A novel hybrid algorithm based on biogeography-based optimization and grey wolf optimizer. *App. Soft Comput.* **67** (2018) 197–214.

[51] B. Zhao, C. Deng, Y. Yang and H. Peng, Novel binary biogeography-based optimization algorithm for the knapsack problem. In: *Lecture Notes in Computer Science*. Springer Nature, Basingstoke (2012) 217–224.

[52] F. Zhao, S. Qin, Y. Zhang, W. Ma, C. Zhang and H. Song, A two-stage differential biogeography-based optimization algorithm and its performance analysis. *Expert Syst. App.* **115** (2019) 329–345.