Research Article

Adaptive Black Hole Algorithm for Solving the Set Covering Problem

Ricardo Soto ¹, Broderick Crawford ¹, Rodrigo Olivares ^{1,2}, Carla Taramasco ²,
Ignacio Figueroa¹, Álvaro Gómez¹, Carlos Castro ³, and Fernando Paredes ⁴

¹Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile

²Escuela de Ingeniería Civil Informática, Universidad de Valparaíso, Valparaíso, Chile

³Universidad Técnica Federico Santa María, Valparaíso, Chile

⁴Escuela de Ingeniería Industrial, Universidad Diego Portales, Santiago, Chile

Correspondence should be addressed to Rodrigo Olivares; rodrigo.olivares@uv.cl

Received 5 December 2017; Revised 5 August 2018; Accepted 3 September 2018; Published 16 October 2018

Academic Editor: Georgios Dounias

Copyright © 2018 Ricardo Soto et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Evolutionary algorithms have been used to solve several optimization problems, showing an efficient performance. Nevertheless, when these algorithms are applied they present the difficulty to decide on the appropriate values of their parameters. Typically, parameters are specified before the algorithm is run and include population size, selection rate, and operator probabilities. This process is known as offline control and is even considered as an optimization problem in itself. On the other hand, parameter settings or control online is a variation of the algorithm original version. The main idea is to vary the parameters so that the algorithm of interest can provide the best convergence rate and thus may achieve the best performance. In this paper, we propose an adaptive black hole algorithm able to dynamically adapt its population according to solving performance. For that, we use autonomous search which appeared as a new technique that enables the problem solver to control and adapt its own parameters and heuristics during solving in order to be more efficient without the knowledge of an expert user. In order to test this approach, we resolve the set covering problem which is a classical optimization benchmark with many industrial applications such as line balancing production, crew scheduling, service installation, and databases, among several others. We illustrate encouraging experimental results, where the proposed approach is able to reach various global optimums for a well-known instance set from Beasley's OR-Library, while improving various modern metaheuristics.

1. Introduction

Nature inspired algorithms are inspired by natural phenomena and behaviours that are met in nature. These algorithms are a subcategory of evolutionary algorithms, which are algorithmic schemes that evolve the solution in each iteration of the algorithm [1]. Evolutionary algorithms use the intelligence of the previous solution for the new one to be generated [2]. In innumerable works, the efficiency of these techniques has been tested. However, it is well known that the performance of metaheuristics depends largely on their correct parameter settings. In fact, finding the appropriate values for the parameters of an algorithm is considered to be a nontrivial task. Previous research has addressed this task that can be classified into two major forms of parameter setting

(see Figure 1): parameter tuning and parameter control. Parameter tuning is known as the process of finding good parameter values before the run of the solver—by using, for instance, training instances—, and then launching the algorithm using these values, which remain static during solving. On the contrary, parameter control launches the algorithm with initial parameter values which are updated during solving time. Now, parameter control is divided into three branches according to the degree of autonomy of the strategies. The control is deterministic when the parameter value is modified by some fixed, predetermined rule using no information from the search. The control is called adaptive, when feedback from the search is employed to determine the variation of a given parameter value, for instance, the population size. Finally, self-adaptive is known as when the

FIGURE 1: Classification of parameter setting.

parameters are encoded into individuals in order to evolve conjointly with the other variables of the problem. This applies to some metaheuristics; for instance, in evolutionary algorithms the better values of these encoded parameters may lead to better individuals, which in turn have more chances to propagate these good configurations to next generations.

This paper proposes an adaptive approach for parameter control by using autonomous search which is a particular case of adaptive systems that improve their solving performance by modifying and adjusting themselves to the problem at hand, either by adaptation or supervised adaptation [3]. Autonomous Search is applied on the black hole algorithm that is a population-based metaheuristic inspired by the black hole phenomenon. The behaviour of this algorithm is principally controlled by two elements: number of solutions and event horizon. The main idea is to automatically adapt the population according to performance exhibited by the algorithm; this population variation indirectly influences on the event horizon.

To evaluate the adaptive approach on the black hole algorithm, we test the set covering problem which is one of the well-known Karp's 21 NP-complete problems. Interesting experimental results are obtained, where the proposed approach is able to obtain various global optimums for a set of 65 well-known set covering problem instances, outperforming also several recently reported techniques.

The remainder of this paper is organized as follows: The related work is introduced in the next section. A brief description of the set covering problem is presented in Section 3. Section 4 details the metaheuristic employed in this work, including their adaptive variation, the binary representation, and the repair of unfeasible solutions. Section 5 illustrates the experimental results, and finally, we conclude and suggest some lines of future research in Section 6.

2. Related Work

In the evolutionary computing field, many optimization problems have widely been treated by using evolutionary

algorithms, such as [4–6] and also [7–9]. In this research line, the parameter setting approach [10] is one of the most studied challenges [11–17]. However, a more reduced work can be observed in the literature when other metaheuristics are involved. For instance, a parameter adaptation study on ant colony optimization is reported in [18], a firefly algorithm for solving the optimal capacitor placement problem is described [19], in [20] a self-adaptive artificial bee colony for constrained numerical optimization is presented, and a modified cuckoo search algorithm for solving engineering problems can be encountered in [21, 22]. Now, considering more similar works, we find a variation of genetic algorithm that adjusts its population size in [23]. The basic idea is to adapt the actual population size depending on the difficulty of the algorithm in its ultimate goal to generate new child chromosomes that outperform their parents. Similarly, [24] presents a version of cuckoo search algorithm that adapts their parameter p_a reflecting the probability whether the nest will be abandoned or updated. In both cases, the adaptive approach depends of quality solution. Finally, in [25] we can observe a variation of the artificial bee colony that consists in controlling the perturbation frequency.

On the other hand, the set covering problem has widely been explored in the optimization and mathematical programming sphere [26]. First works have used exact techniques, for instance branch-and-bound and branch-and-cut algorithms [27–30]. Greedy algorithms have been proposed as a real alternative for solving the set covering problem; nevertheless their deterministic nature hinders the generation of high quality solutions [31]. Under this line, it is well known that exact methods are in general unable to tackle large instances of NP-complete problems, consequently much research work has been devoted to the study of efficient metaheuristics to solve hard instances of the set covering problem in an amount of bounded time. For instance, ant colony optimization [32–35], simulated annealing [36], tabu search [37], and genetic algorithms [38] have extensively been proposed to tackle the classic set covering problem. In recent years, research has been driven towards solving this problem by using recent bioinspired algorithms, such as

teaching-learning based optimization [39] firefly optimization [40], cat swarm optimization [41], shuffled frog leaping [42], artificial bee colony [43], cuckoo search [44], and black hole algorithm [44], among others.

Finally, we can say that an adaptive approach for bio-inspired algorithms to solve the set covering problem—or other optimization problem—is an interesting research line that has not yet been highly exploited.

3. Set Covering Problem

In this section we present the set covering problem. The set covering problem consists in finding a set of solutions at the lowest possible cost to cover a set of needs. Line balancing production [45], crew scheduling [46–48], service installation [49, 50], databases [51], production flow-lines optimization [52], and Boolean expressions [53] are some practical examples of set covering problem. Formally, we define the problem as follows: Let $A = (a_{ij})$ be a binary matrix with M -rows $\times$ N -columns, 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 can observe that column ($j \in N$) cover a row i that exists in M , if $a_{ij} = 1$. The corresponding mathematical is stated in the following:

$$\text{minimize} \quad \sum_{j=1}^N c_j x_j \quad (1)$$

subject to

$$\sum_{j=1}^N a_{ij} x_j \geq 1, \quad \forall i = \{1, \dots, M\} \quad (2)$$

$$x_j \in \{0, 1\}, \quad \forall j = \{1, \dots, N\} \quad (3)$$

The aim is to minimize the summation of column costs, where $x_j = 1$ if component j is part of the solution and $x_j = 0$ otherwise. The constraints of the set covering problem guarantee that every row i is covered by at least one column j .

4. Black Hole Algorithm

The black hole algorithm is a population-based algorithm inspired on the black hole phenomenon [54, 55]. A black hole is a zone of space that has so much mass concentrated in it that there is no way for a nearby object to escape its gravitational force. Anything falling into a black hole, including light, cannot escape.

Similar to other population-based algorithms, the black hole algorithm begins with an initial population of potential solutions, called “stars”. At each iteration of the black hole algorithm, the best solution is chosen to be the black hole, which then starts attracting other candidates around it. If the fitness of a star crosses the event horizon of the black hole (Eq. (5)), it will be swallowed and it is gone forever. In such case, a new star (potential solution) is randomly generated and placed in the search space and starts a new search.

The black hole has the ability to absorb the stars that surround it. After initializing the black hole and stars, the black hole begins by absorbing the stars around it and all the stars start moving towards the black hole. The absorption of stars by the black hole is formulated as follows:

$$x_i^j(t+1) = x_i^j(t) + r [x_{BH}^j - x_i^j(t)], \quad (4)$$

$$\forall i \in \{1, \dots, popSize\}$$

where $x_i^j(t)$ and $x_i^j(t+1)$ are the components of a solution i at iterations t and $(t+1)$, respectively. x_{BH}^j is the component of best solution (black hole) in the search space. r is a random number in the interval $[0, 1]$. Finally, $popSize$ represents the number of stars (potential solutions).

While moving towards the black hole, a star may reach a location with lower cost than the black hole. In such case, the black hole moves to the location of that star and vice versa. Then the algorithm will continue with the black hole in the new location and then stars start moving towards this new location.

In addition, there is the probability of crossing the event horizon during moving stars towards the black hole. Every star (or potential solution) that crosses the event horizon of the black hole will be swallowed by the black hole. Every time a candidate is sucked in by the black hole, another potential solution (star) is born and distributed randomly in the search space and starts a new search. This is done to keep the number of potential solutions constant. The next iteration takes place after all the stars have been moved.

The radius of the event horizon in the black hole algorithm is calculated using the following equation:

$$R = \frac{f_{BH}}{\sum_{i=1}^{popSize} f_i} \quad (5)$$

where f_{BH} is the fitness value of the black hole, f_i is the fitness value of the i -th star, and $popSize$ is the number of stars (potential solutions).

When the distance between a star and the black hole (best candidate) is less than R , that candidate is collapsed and a new star is created and distributed randomly in the search space. Based on the above description the main steps in the black hole algorithm are described in detail in Algorithm 1.

The black hole algorithm begins with the loading and processing phases. Then, at Lines 3-8, an initial population of $popSize$ stars is randomly generated as a vector of N binary values that corresponds to one star or potential solution (Line 5). For each star, the fitness value is calculated by evaluating the objective function *cost* (Line 7).

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 11-37). As previously mentioned, the set covering problem is a minimization problem. This evaluation is handled in comparison presented at Line 13. If the new minimum value is less than minimum global, the minimum global is changed by the new minimum value and the best solution is stored in $\hat{x}$ (black hole).

Require: Problem input data, $popSize$, T
Ensure: The best solution that resolves the set covering problem

```

1:  $\{N, c_j\} \leftarrow loadProblemData()$ 
2: {Produce the first generation of  $popSize$  stars}
3: for all star  $x_i$ , ( $\forall i = 1, \dots, popSize$ ) do
4:   for all dimension  $j$ , ( $\forall j = 1, \dots, N$ ) do
5:      $x_i^j \leftarrow Random\{0, 1\}$ 
6:   end for
7:    $f_i \leftarrow cost(x_i)$ 
8: end for
9:  $globalfit \leftarrow +\infty$ 
10: {Produce  $T$ -generations of  $popSize$  stars}
11: while  $t < T$  do
12:    $\{minfit, minindex\} \leftarrow min(f)$ 
13:   if  $minfit < globalfit$  then
14:      $globalfit \leftarrow minfit$ 
15:     for all dimension  $j$ , ( $\forall j = 1, \dots, N$ ) do
16:        $\hat{x}^j(t) \leftarrow x_{minindex}^j(t)$ 
17:     end for
18:   end if
19:   for all star  $x_i$ , ( $\forall i = 1, \dots, popSize$ ) do
20:     {Calculate the event horizon through Eq. (5) and distance between the current solution with respect to the best solution}
21:      $R \leftarrow \frac{f_{BH}}{\sum_{i=1}^{popSize} cost(x_i)}$ 
22:      $rand \leftarrow Random[0, 1]$ 
23:     if  $R > rand$  then
24:       for all dimension  $j$ , ( $\forall j = 1, \dots, N$ ) do
25:          $x_i^j(t) \leftarrow Random\{0, 1\}$ 
26:       end for
27:     end if
28:   end for
29:   for all star  $x_i$ , ( $\forall i = 1, \dots, popSize$ ) do
30:     for all dimension  $j$ , ( $\forall j = 1, \dots, N$ ) do
31:       {Generate new solutions through Eq. (4)}
32:        $x_i^j(t+1) = x_i^j(t) + r[x_{BH}^j - x_i^j(t)]$ 
33:       {In previous step, the value generated belongs to the real domain and it must be brought to a binary domain}
34:        $x_i^j(t+1) \leftarrow g(x_i^j(t+1))$ 
35:     end for
36:   end for
37: end while
38: return Postprocess results and visualization;

```

ALGORITHM 1: Black hole algorithm.

If the fitness of a star crosses the event horizon of the black hole, replace it with a new star in a random location in the search space. This process is described in the for loop statement at Lines 19–28.

Finally, the last for loop statement (Lines 29–36) generates new solutions according to Equation (4), for each dimension j . This value belongs to real domain and it must be brought to a binary domain; thus a T function is used again to transform the real value in a binary one.

4.1. Adaptive Black Hole Algorithm. The basic version of the black hole algorithm has not control parameters. If we observe the parameter $popSize$, we can see that the population size is valued before the run of the metaheuristic. At each iteration of the algorithm, the solutions are altered

updating their position or when they are absorbed by the black hole, in which case a new random solution—or star—is created. However, the total number of solutions is preserved. The second part of the metaheuristic uses an absorption process, called event horizon that is a stochastic procedure and it is influenced by the best solution, discarding bad solutions that seem to lead to poor results. This is not necessarily a correct idea, since often a bad solution can be drive the search to an optimal one.

These concerns are very important for the performance of the algorithm, reason why we have decided to take the event horizon logic to calculate the value of the population size parameter in autonomous way. For that, we use autonomous search which is a particular case of adaptive systems that improve their solving performance by modifying

ith solution	1	0	1	...	0	1
x_i	x_i^1	x_i^2	x_i^3	...	$x_i^{(j-1)}$	x_i^j

FIGURE 2: Binary solution representation.

and adjusting themselves to the problem at hand, either by adaptation or supervised adaptation (please see [3, 56] for details). This approach has been successfully applied in constraint programming using bioinspired algorithms for controlling the process resolution of solver tools [57, 58]. The objective of autonomous search is to allow the metaheuristic to self-adapt the value of the parameter *popSize* during the run, according to the algorithm convergence.

The literature presents several works illustrating how to improve evolutionary algorithms by dynamically controlling parameters during solving time [13–17, 22]. We have decided to modify the original black hole algorithm considering the autonomous search principles for building a procedure that adaptively vary the number of solutions (population size parameter), according to the performance exhibited during search.

We are inspired by this approach to improve the local search process of black hole algorithm. The main objective is to give the algorithm the ability to adjust its population size. When algorithm is stagnant some solutions are improved according to the best solution. In opposition, if solutions are far away from the best solution, the worst stars are eliminated and new ones are randomly generated.

We vary only the population parameter since is the only one available to be modified during solving time, this make the implementation easier and the solving process improved. We believe this approach also provide know-how for future experiments involving adaptive population. The procedure is described in Algorithm 2.

Inputs of the procedure are the value of population size *popSize*, current iteration *t*, and number of stagnation *ls* that represents the number of iterations where the best solution does not improve, commonly called “local search”.

Reached this point, we calculate the distance between best and worst solutions according to cost of each one and then it divides by sum of all costs (Line 2). This value describes the percentage of maximum separation between solutions. A low percentage value indicates that solutions are homogeneous skewed to the best solution, in which case the procedure creates new solutions (Lines 5-18). To create these solutions, we reuse the percentage to determinate if they are cloned from best solution (Lines 8-10) or they are randomly generated (Lines 12-16). On the other hand, if *prob* > *rand* means that solutions are heterogeneous; thus the procedure remove the worst solutions (Lines 20-22).

In both cases, the percentages calculated can also be seen as a mechanism to support the exploration and exploitation phases. If solution are homogeneous, this procedure allows to explore towards new solutions, while if solutions are heterogeneous, the procedure converges towards a set of similar solutions itself.

TABLE 1: $\mathcal{S}$ -Shape and $\mathcal{V}$ -Shape transfer functions.

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

For computational experiments we use *popSize* = 20 potential solution and *T* = 5000 number of generations. Although the approach proposed is adaptive, it is necessary to define an initial value for *popSize*.

4.2. Binary Approaches. It is known that variables of the set covering problem are limited to binary values, namely, $x_i^j \in \{0, 1\}$, $\forall j \in \{1, \dots, N\}$: for this reason, we use a binary representation for each solution given by the black hole algorithm, as shown in Figure 2, where

$$x_i^j = \begin{cases} 1, & \text{if the dimension } j \text{ is part of the solution} \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

In this paper, we use the standard black hole algorithm [59], where each star represent a candidate solution.

However, the standard version of the black hole 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. In this work, we tested 32 different binarization strategies.

We evaluate different functions, separated into two families [60, 61]: $\mathcal{S}$ -Shape and $\mathcal{V}$ -Shape (see Table 1).

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

- (1) Standard: if condition is satisfied, standard method returns 1; otherwise, return 0.

$$x_i^j = \begin{cases} 1, & \text{if } rand \leq g(x_i^j) \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

Require: Population size $popSize$, current iteration t , number of stagnation ls , and size of instance N .

Ensure: New value for the parameter $popSize$.

```

1: if  $t$  achieves  $ls$  then
2:    $prob \leftarrow \frac{|cost(x_{bh}) - cost(x_{wrong})|}{\sum_{i=1}^{popSize} cost(x_i)}$ 
3:    $rand \leftarrow Random[0, 1]$ 
4:   if  $rand > prob$  then
5:      $b \leftarrow roundup(popSize \times prob)$ 
6:      $rand \leftarrow Random[0, 1]$ 
7:     if  $rand > prob$  then
8:       for all star  $x_i, (\forall i = 1, \dots, b)$  do
9:          $x_{n+i} \leftarrow x_{bh}$ 
10:      end for
11:    else
12:      for all star  $x_i, (\forall i = 1, \dots, b)$  do
13:        for all dimension  $j, (\forall j = 1, \dots, N)$  do
14:           $x_{popSize+i}^j \leftarrow Random\{0, 1\}$ 
15:        end for
16:      end for
17:    end if
18:     $popSize \leftarrow popSize + b$ 
19:  else
20:     $w \leftarrow roundup(popSize \times prob)$ 
21:    if  $popSize \geq w$  then
22:      {Remove  $w$  worst solutions}
23:       $popSize \leftarrow popSize - w$ 
24:    end if
25:  end if
26: end if

```

ALGORITHM 2: Adaptive approach for the population size parameter ($popSize$).

- (2) Complement: if condition is satisfied, standard method returns the complement value.

$$x_i^j = \begin{cases} \overline{x_i^j}, & \text{if } rand \leq g(x_i^j) \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

- (3) Static probability: a probability is generated and it is evaluated with a transfer function.

$$x_i^j = \begin{cases} 0, & \text{if } g(x_i^j) \leq \alpha \\ x_i^j, & \text{if } \alpha < g(x_i^j) \leq \frac{1}{2}(1 + \alpha) \\ 1, & \text{if } g(x_i^j) \geq \frac{1}{2}(1 + \alpha) \end{cases} \quad (9)$$

- (4) Elitist: discretization method Elitist Roulette, also known as Monte Carlo, is to select randomly among the best individuals of the population, with a probability proportional to its fitness.

$$x_i^j = \begin{cases} x_i^j, & \text{if } rand \leq g(x_i^j) \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

4.3. Heuristic Feasibility Operator. Generally, metaheuristics may provide solutions that violate the constraints of the problem. For instance, a new set covering problem solution owning uncovered rows, clearly violates a subset of constraints. In order 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 feasible solutions, 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

$$\frac{cost(column_j)}{coveredRows(column_j)} \quad (11)$$

The unfeasible solutions are repaired by covering 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 from 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 is “greedy” heuristic. On the one hand, between the instructions 8 and 12, the columns with lower ratio are added to the solution. On the other hand, between Lines 13 and 18,

```

1:  $\alpha_i \leftarrow$  The set of columns that cover row  $i, i \in M$ 
2:  $\beta_j \leftarrow$  The set of rows covered by column  $j, j \in N$ 
3:  $S \leftarrow$  The set of columns in a solution
4:  $w_i \leftarrow$  The number of columns that cover row  $i, i \in M$ . For this,  $w_i \leftarrow |S \cap \alpha_i|, \forall i \in M$ 
5:  $U \leftarrow$  The set of uncovered rows. For this,  $U = \{i \mid w_i = 0, \forall i \in M\}$ 
6: for all row  $i \in U$  (in increasing order of  $i$ ) do
7:   Find the first column  $j$  in increasing order of  $j \in \alpha_i$  that minimizes  $c_j/|U \cap \beta_j|$ 
8:   Add  $j$  to  $S$  and set  $w_i \leftarrow w_i + 1, \forall i \in \beta_j$ 
9:   Set  $U \leftarrow U - \beta_j$ 
10: end for
11: for all column  $j \in S$  (in decreasing order of  $j$ ) do
12:   if  $w_i \geq 2$  then
13:      $S \leftarrow S - j$ 
14:      $w_i \leftarrow w_i - 1, \forall i \in \beta_j$ 
15:   end if
16: end for
17: return  $S$  is now a feasible solution for the set covering problem that contains no redundant columns;

```

ALGORITHM 3: Heuristic feasibility operator.

the redundant columns with higher costs are deleted while the solution is feasible.

5. Experimental Results

After to apply the adaptive approach, we have analyzed the time complexity of the basic algorithm to illustrate that our proposal does not affect its performance. If we observe each flow control, statements and expressions from the basic black hole algorithm, we can determine that time complexity is given by $O(T \times popSize \times N)$, where T is a constant representing the maximum of iterations and it is avoided, $popSize$ value is the number of solutions, and N is the dimension of each solution. In worst case, the basic algorithm is upper bounded by $O(kn^2)$.

Now, evaluating time complexity about our adaptive approaches, we determine that the upper bound is given by $O(b \times N)$, where b is the new number of solutions and N is the dimension of each solution. This procedures are activated when the number of iterations achieve the local search criteria ls , therefore they are independent of the main algorithm and as a consequence its performance and complexity time are not altered.

With respect to space analysis about our approach, we considerate the space used in principal memory. In this context, every time that the adaptive algorithms are launched, we observe that the usage of the principal memory is not outperformed to 8%. The basic algorithm use a similar percentage of principal memory.

The performance of the adaptive black hole algorithm is experimentally evaluated by using 65 instances of the set covering problem organized in 11 sets taken from the Beasley's OR-library [62]. Table 2 describes instance set, number of rows or constraints M , number of columns or variables (dimension) N , range of costs, and density (percentage of nonzeros in the matrix).

TABLE 2: Instances from Beasley's OR-Library.

Instance group	M	N	Cost range	Density (%)
4	200	1000	[1,100]	2
5	200	2000	[1,100]	2
6	200	1000	[1,100]	5
A	300	3000	[1,100]	2
B	300	3000	[1,100]	5
C	400	4000	[1,100]	2
D	400	4000	[1,100]	5
E	500	5000	[1,100]	10
F	500	5000	[1,100]	20
G	1000	10000	[1,100]	2
H	1000	10000	[1,100]	5

In order to reduce the instance size of set covering problem, we have used a preprocessed instances set. Different preprocessing methods have particularly been proposed in [63]. We employ two of them, which have been proved to be the most effective ones:

- (i) *Column Domination*. The nonunicost set covering problem holds different column costs, then once a set of rows I_j is covered by another column j' and $c_{j'} < c_j$, we say that column j is dominated by j' , then column j is removed from the solution.
- (ii) *Column Inclusion*. If a row is covered by only one column after the above domination, it means that there is no better column to cover those rows; consequently this column must be included in optimal solution.

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} that in our case is the best known value for each instance and it is calculated as follows:

$$RPD = \left(\frac{Z_{min} - Z_{opt}}{Z_{opt}} \right) \quad (12)$$

The minimum (Min), maximum (Max), and average (Avg) of the solutions obtained were achieved running 30 runs over each one of set the 65 test instances. We test all the combinations of transfer functions and discretization methods over all these instances. Before to test all instances, we perform a sampling phase to evaluate which binarization strategy would show the best performance. In appendix section we include preliminary results in the sampling phase (see Tables 9 and 10 and Figures 7 and 8). The binarization strategy that achieved the best results for the black hole algorithm is $\mathcal{S}_4 + Elitist$.

5.1. Black Hole Algorithms Comparison. In this Section, we compare the proposed adaptive black hole algorithm with the basic black hole one. To the best of our knowledge, no autonomous search implementation on top of black hole has been proposed yet to include in the comparison.

The algorithm implementation has been done using Java SE 7 and the experiments have been launched on a 2.3 Ghz Intel Core i3 with 4 GB RAM machine running Windows 7 (available at <https://goo.gl/AEYXJq>). The initial parameter setting used is as follows: a population size of 40, 4000 iterations, a local search of 40 and 30 executions for each instance. Bold font and underlined row are used to present when the adaptive black hole algorithm outperforms their basic version.

Table 3 illustrates the results obtained for all instances from groups 4, 5, 6, A, B, C, and D. Regarding instance sets 4 and 6, the proposed adaptive approach in this work (BH-AS) improves dramatically the results in 50%, reaching 4 global optimums. Considering instance sets 5 and 6, we can observe that the basic black hole algorithm exhibits great performance achieving 80% of the optimal values. However, the adaptive process of the population size parameter improves results obtaining 2 more optimal values. Now, if we analyze instances from groups A, B, C, and D, we can see that again the basic black hole algorithm shows an outstanding efficiency to solve the set covering problem, solving to optimally 15 of 20 instances. Nevertheless, the adaptive approach shows that it can improve even more, finding one more optimal value for the instance C.1.

Table 4 presents the results obtained for instances from groups NRE, NRF, NRG, and NRH. If we consider that the basic black hole algorithm reaches 3 optimal values from 20, there is a wide margin to improve. Taking into account the fact that the autoadaptive methods outperform the basic BH algorithm in 11 of 17 possible instances and 2 of them are global optima, we can say that BH-AS performs significantly better than the basic version.

Figure 3 illustrates convergence charts for the hardest instances of each instance group. Here, we observe that

for the 4 group the convergence of the BH-AS is clearly faster than the basic algorithm. For the group 5, the BH-AS begins with a bad quality solution but at the middle of the process improves its performance outperforming the basic BHA similar behaviour can be seen for the NRE and NRH group. The performance of instances from groups 6, A, B, C, and D is similar keeping all of them an early convergence. Finally, for benchmarks from groups NRF and NRG, the behaviour of BH-AS is clearly earlier than its competitor.

Now, if we regard solving times required for reaching the solutions, we may observe that times are very similar for the two algorithms. However, we must consider that BH-AS need the computation of the adaptive process and is able to outperform the basic BH in terms of optimum values reached. We can observe also a small difference in terms of solving times in favor of the BH-AS with respect to BH.

5.2. BH-AS v/s Original Approach. After to analyze the exhibited efficiency for BH-AS and the basic algorithm for solving the set covering problem, we have observed that the variation is minimal. It is not possible to state which technique is the best. However, in order to show a significant difference between the adaptive approach and the original black hole algorithm, we perform a contrast statistical test for each instance through the *Kolmogorov-Smirnov-Lilliefors* to determine the independence of samples [64] and *Wilcoxon's signed rank* [65] to compare statistically the results.

For both tests, we consider a hypothesis evaluation, which is analyzed assuming a *p-value* of 0.05—accuracy—i.e., smaller values than 0.05, determines that the corresponding hypothesis cannot be assumed. Both tests were conducted using *GNU Octave* (available at <http://goo.gl/jtHn8i>).

The first test allows us to analyze the independence of samples. For that, we run the algorithm 30 times for each instance and then, we apply the test. According to obtained values, we can decide if samples follow a normal distribution or they are independent. To proceed, we propose the following hypotheses:

- (i) H_0 states that Z_{min} follows a normal distribution.
- (ii) H_1 states the opposite.

The conducted test has yielded *p-value* lower than 0.05; therefore H_0 cannot be assumed.

Then, as samples are independent and cannot be assumed that 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 compare the medians of two distributions. To proceed, we propose the following new hypotheses:

- (i) H_0 : $\bar{Z}_{min}$ achieved by basic black hole algorithm < $\bar{Z}_{min}$ achieved by BH-AS.
- (ii) H_1 states the opposite.

Table 5 compares the approaches for all tested instances via the *Wilcoxon's signed rank* test. As the significance level is also established to 0.05, smaller values that 0.05 defines that

TABLE 3: Computational results of groups 4, 5, 6, A, B, C, and D.

Instance	Z_{opt}	Black Hole Algorithm					
		Basic			BH-AS		
		Z_{min}	RPD	times (ms)	Z_{min}	RPD	times (ms)
4.1	429	430	0.002	1011.3	430	0.002	1181.2
4.2	512	512	0.000	1867.0	512	0.000	1926.0
4.3	516	516	0.000	2007.5	516	0.000	1957.1
4.4	494	495	0.002	1998.2	494	0.000	2108.1
4.5	512	514	0.004	2010.1	512	0.000	1992.3
4.6	560	560	0.000	2021.3	560	0.000	2102.2
4.7	430	430	0.000	2014.5	430	0.000	2032.2
4.8	492	494	0.004	2311.0	492	0.000	2202.6
4.9	641	644	0.005	2099.1	643	0.003	2109.9
4.10	514	515	0.002	1981.7	514	0.000	2017.1
5.1	253	253	0.000	2171.9	253	0.000	2001.7
5.2	302	305	0.010	1992.4	302	0.000	2071.2
5.3	226	228	0.000	2221.8	228	0.000	2221.8
5.4	242	242	0.000	2008.1	242	0.000	2191.2
5.5	211	211	0.000	2001.4	211	0.000	2127.4
5.6	213	213	0.000	2091.1	213	0.000	2222.1
5.7	293	293	0.000	2132.3	293	0.000	2332.1
5.8	288	288	0.000	2199.7	288	0.000	2339.2
5.9	279	279	0.000	2351.6	279	0.000	2271.2
5.10	265	265	0.000	1989.4	265	0.000	2057.7
6.1	138	140	0.014	2915.7	138	0.000	2915.7
6.2	146	147	0.007	2910.1	147	0.007	2791.6
6.3	145	145	0.000	2717.1	145	0.000	2751.7
6.4	131	131	0.000	2981.3	131	0.000	3003.7
6.5	161	161	0.000	3081.2	161	0.000	3038.3
A.1	253	253	0.000	3009.5	253	0.000	2967.1
A.2	252	253	0.004	3033.2	253	0.004	2983.2
A.3	232	233	0.004	2991.9	233	0.004	3104.9
A.4	234	234	0.000	3091.2	234	0.000	3111.1
A.5	236	236	0.000	3001.4	236	0.000	3012.3
B.1	69	69	0.000	3223.5	69	0.000	3122.1
B.2	76	76	0.000	3011.3	76	0.000	3183.6
B.3	80	80	0.000	3253.3	80	0.000	3452.0
B.4	79	79	0.000	3212.1	79	0.000	3419.1
B.5	72	72	0.000	3433.3	72	0.000	3501.0
C.1	227	229	0.009	3271.1	227	0.000	3397.9
C.2	219	219	0.000	3090.4	219	0.000	3291.1
C.3	243	245	0.008	3194.6	245	0.008	3229.2
C.4	219	219	0.000	3230.2	219	0.000	3323.1
C.5	215	215	0.000	2979.6	215	0.000	3111.8
D.1	60	60	0.000	5821.3	60	0.000	6021.0
D.2	66	67	0.015	6010.2	67	0.015	6094.1
D.3	72	73	0.014	6071.2	73	0.014	6147.7
D.4	62	62	0.000	6313.4	62	0.000	6511.6
D.5	61	61	0.000	6831.4	61	0.000	6712.2

TABLE 4: Computational results of groups NRE, NRF, NRG, and NRH.

Instance	Z_{opt}	Black Hole Algorithm					
		Basic			BH-AS		
		Z_{min}	RPD	times (ms)	Z_{min}	RPD	times (ms)
NRE.1	29	30	0.034	6200.0	<u>29</u>	<u>0.000</u>	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	<u>28</u>	<u>0.000</u>	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	<u>16</u>	<u>0.143</u>	9293.8
NRF.4	14	16	0.143	7746.3	14	0.000	8023.3
NRF.5	13	14	0.077	7021.1	14	0.077	7220.2
NRG.1	176	182	0.034	7143.5	<u>179</u>	<u>0.017</u>	7644.8
NRG.2	154	160	0.039	7948.0	<u>158</u>	<u>0.026</u>	8584.5
NRG.3	166	170	0.024	8771.8	<u>169</u>	<u>0.018</u>	9874.0
NRG.4	168	170	0.012	8123.7	<u>170</u>	<u>0.012</u>	8919.2
NRG.5	168	170	0.012	9013.4	<u>168</u>	<u>0.000</u>	9921.2
NRH.1	63	66	0.048	15114.6	<u>65</u>	<u>0.032</u>	17911.2
NRH.2	63	67	0.063	16343.4	<u>67</u>	<u>0.063</u>	19733.4
NRH.3	59	64	0.085	19243.7	<u>64</u>	<u>0.085</u>	21840.3
NRH.4	58	64	0.103	18241.6	<u>63</u>	<u>0.086</u>	21421.7
NRH.5	55	63	0.145	24442.6	<u>62</u>	<u>0.127</u>	22242.5

TABLE 5: Statistical test.

Instance	4.1	4.2	4.3	4.4	4.5	4.6	4.7	4.8	4.9	4.10
Basic	–	–	<u>0.01</u>	–	–	–	<u>0.02</u>	–	–	–
BH-AS	<u>0.01</u>	–	–	<u>0.01</u>	<u>0.01</u>	–	–	<u>0.04</u>	<u>0.01</u>	<u>0.01</u>
Instance	5.1	5.2	5.3	5.4	5.5	5.6	5.7	5.8	5.9	5.10
Basic	–	–	<u>0.01</u>	–	–	–	–	–	–	<u>0.04</u>
BH-AS	–	<u>0.02</u>	–	–	<u>0.03</u>	<u>0.01</u>	<u>0.05</u>	–	–	–
Instance	6.1	6.2	6.3	6.4	6.5	A.1	A.2	A.3	A.4	A.5
Basic	–	–	–	–	–	<u>0.02</u>	<u>0.05</u>	–	–	–
BH-AS	<u>0.01</u>	–	<u>0.01</u>	–	<u>0.01</u>	–	–	<u>0.01</u>	<u>0.01</u>	<u>0.01</u>
Instance	B.1	B.2	B.3	B.4	B.5	C.1	C.2	C.3	C.4	C.5
Basic	–	–	–	<u>0.03</u>	<u>0.02</u>	–	–	–	<u>0.01</u>	–
BH-AS	–	<u>0.01</u>	<u>0.01</u>	–	–	–	<u>0.05</u>	<u>0.01</u>	–	<u>0.04</u>
Instance	D.1	D.2	D.3	D.4	D.5	NRE.1	NRE.2	NRE.3	NRE.4	NRE.5
Basic	–	–	–	–	–	–	<u>0.03</u>	–	–	<u>0.01</u>
BH-AS	<u>0.02</u>	–	–	<u>0.01</u>	<u>0.03</u>	<u>0.01</u>	–	<u>0.01</u>	<u>0.04</u>	–
Instance	NRF.1	NRF.2	NRF.3	NRF.4	NRF.5	NRG.1	NRG.2	NRG.3	NRG.4	NRG.5
Basic	–	<u>0.01</u>	–	–	–	–	–	–	–	–
BH-AS	<u>0.03</u>	–	<u>0.03</u>	<u>0.01</u>	<u>0.03</u>	<u>0.01</u>	<u>0.03</u>	<u>0.01</u>	<u>0.01</u>	<u>0.02</u>
Instance	NRH.1		NRH.2		NRH.3		NRH.4		NRH.5	
Basic	–		–		–		–		–	
BH-AS	<u>0.03</u>		–		<u>0.01</u>		<u>0.03</u>		<u>0.05</u>

FIGURE 3: Continued.

FIGURE 3: Convergence charts for the hardest instances of each instance group.

TABLE 6: Comparison results for instance set of groups 4, 5, 6, and A.

Instance	4.1	4.2	4.3	4.4	4.5	4.6	4.7	4.8	4.9	4.10
Z_{opt}	429	512	516	494	512	560	430	492	641	514
BH-AS	Z_{min}	430	512	516	494	512	560	430	492	643
	Z_{avg}	430	512	516	501	519	562	432	495	648
	RPD	0.002	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.003
	Time (ms)	1811.2	1926.0	1957.1	2108.1	1992.3	2102.2	2032.2	2202.6	2109.9
BCSO	Z_{min}	459	570	590	547	545	637	462	546	711
	Z_{avg}	480	594	607	578	554	650	467	567	725
	RPD	7	11.3	14.3	10.7	6.4	13.8	7.4	11	10.9
BFO	Z_{min}	429	517	519	495	514	563	430	497	655
	Z_{avg}	430	517	522	497	515	565	430	499	658
	RPD	0	0.97	0.58	0.2	0.39	0.53	0	1.01	2.18
BSFLA	Z_{min}	430	516	520	501	514	563	431	497	656
	Z_{avg}	430	518	520	504	514	563	432	499	656
	RPD	0.23	0.78	0.78	1.42	0.39	0.54	0.23	1.02	2.34
BELA	Z_{min}	447	559	537	527	527	607	448	509	682
	Z_{avg}	448	559	539	530	529	608	449	512	682
	RPD	4.20	9.18	4.07	6.68	2.93	8.39	4.19	3.46	6.40
BABC	Z_{min}	430	513	519	495	514	561	431	493	649
	Z_{avg}	430	513	521	496	517	565	434	494	651
	RPD	0.23	0.20	0.58	0.20	0.39	0.18	0.23	0.20	0.93
MIP	Z_{min}	429	512	516	494	512	560	430	492	641
	Time (ms)	705	912	788	934	654	701	693	712	917
Instance	5.1	5.2	5.3	5.4	5.5	5.6	5.7	5.8	5.9	5.10
Z_{opt}	253	302	226	242	211	213	293	288	279	265
BH-AS	Z_{min}	253	302	226	242	211	213	293	288	279
	Z_{avg}	253	302	226	242	211	213	293	288	279
	RPD	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Time (ms)	2001.7	2071.2	2221.8	2191.2	2127.4	2222.1	2332.1	2339.2	2271.2
BCSO	Z_{min}	279	339	247	251	230	232	332	320	295
	Z_{avg}	287	340	251	253	230	243	338	330	297
	RPD	10.3	12.3	9.3	3.7	9	8.9	13.3	11.1	5.7
BFO	Z_{min}	257	309	229	242	211	213	298	291	284
	Z_{avg}	260	311	233	242	213	213	301	292	284
	RPD	1.58	2.31	1.32	0	0	0	1.7	1.04	1.79
BSFLA	Z_{min}	254	307	228	242	211	213	297	291	281
	Z_{avg}	255	307	230	242	213	214	299	293	283
	RPD	0.4	1.66	0.88	0	0	0	1.37	1.04	0.72
BELA	Z_{min}	280	318	242	251	225	247	316	315	314
	Z_{avg}	281	321	240	252	227	248	317	317	315
	RPD	10.67	5.30	7.08	3.72	6.64	15.96	7.85	9.38	12.54
BABC	Z_{min}	254	309	229	242	211	214	298	289	280
	Z_{avg}	255	309	233	245	212	214	301	291	281
	RPD	0.40	2.32	1.33	0	0	0.47	1.71	0.35	0.36
MIP	Z_{min}	253	302	226	242	211	213	293	288	279
	Time (ms)	4776	3882	4332	4899	4771	4329	3887	5191	4659
Instance	6.1	6.2	6.3	6.4	6.5	A.1	A.2	A.3	A.4	A.5
Z_{opt}	138	146	145	131	161	253	252	232	234	236

TABLE 7: Continued.

Instance		D.1	D.2	D.3	D.4	D.5	NRE.1	NRE.2	NRE.3	NRE.4	NRE.5
Z_{opt}		60	66	72	62	61	29	30	27	28	28
BH-AS	Z_{min}	60	67	73	62	61	29	31	28	29	28
	Z_{avg}	60	67	73	62	61	29	31	28	28	28
	RPD	0.000	0.015	0.014	0.000	0.000	0.000	0.033	0.037	0.000	0.000
	Time	6021.0	6094.1	6147.7	6511.6	6712.2	6200.0	6115.1	5991.9	7081.0	7191.1
BCSO	Z_{min}	65	70	79	64	65	29	34	31	32	30
	Z_{avg}	66	70	81	67	66	30	34	32	33	30
	RPD	8.3	6.1	9.7	3.2	6.6	0	13.3	14.8	14.3	7.1
BFO	Z_{min}	60	68	75	62	63	29	32	29	29	29
	Z_{avg}	61	68	77	62	63	31	32	30	31	29
	RPD	0	3.03	4.16	0	3.27	0	6.66	7.4	3.57	3.57
BSFLA	Z_{min}	60	67	75	63	63	29	31	28	29	28
	Z_{avg}	62	68	77	65	66	29	32	28	30	31
	RPD	0	1.52	4.17	1.61	3.28	0	3.33	3.7	3.57	0
BELA	Z_{min}	62	73	79	67	66	30	35	34	33	30
	Z_{avg}	62	74	81	69	67	31	35	34	34	31
	RPD	3.33	10.61	9.72	8.06	8.20	3.45	16.67	25.93	17.86	7.14
BABC	Z_{min}	60	68	76	63	63	29	32	29	29	29
	Z_{avg}	61	68	77	65	66	33	32	31	30	32
	RPD	0	3.03	5.56	1.61	3.28	0	6.67	7.41	3.57	3.57
MIP	Z_{min}	–	–	–	–	–	–	–	–	–	–
	Time (ms)	t.o.	t.o.	t.o.	t.o.	t.o.	t.o.	t.o.	t.o.	t.o.	t.o.
Instance		NRE.1	NRE.2	NRE.3	NRE.4	NRE.5	NRG.1	NRG.2	NRG.3	NRG.4	NRG.5
Z_{opt}		14	15	14	14	13	176	154	166	168	168
BH-AS	Z_{min}	14	15	16	14	14	179	158	169	170	168
	Z_{avg}	15	16	16	16	15	181	160	169	171	169
	RPD	0.000	0.000	0.143	0.000	0.077	0.017	0.026	0.018	0.012	0.000
	Time	7118.5	8212.1	9293.8	8023.3	7220.2	7494.4	8584.5	9874.0	8919.2	9921.2
BCSO	Z_{min}	17	18	17	17	15	190	165	187	179	181
	Z_{avg}	17	18	17	17	16	193	166	188	183	184
	RPD	21.4	20	21.4	21.4	15.4	8	7.1	20.6	6.5	7.7
BFO	Z_{min}	15	16	16	15	15	185	161	175	176	177
	Z_{avg}	17	16	17	18	19	191	163	177	176	181
	RPD	7.14	6.66	14.28	7.14	15.38	5.11	4.54	5.42	4.76	5.35
BSFLA	Z_{min}	15	15	16	15	15	182	161	173	173	174
	Z_{avg}	15	15	17	16	17	183	161	174	177	174
	RPD	7.14	0	14.29	7.14	15.38	3.41	4.55	4.22	2.98	3.57
BELA	Z_{min}	17	18	17	17	16	194	176	184	196	198
	Z_{avg}	17	18	18	19	17	196	176	185	197	199
	RPD	21.43	20	21.43	21.43	23.08	10.23	14.29	10.84	16.67	17.86
BABC	Z_{min}	14	16	16	15	15	183	162	174	175	179
	Z_{avg}	15	16	17	17	16	184	163	175	177	181
	RPD	0	6.67	14.29	7.14	15.38	3.98	5.19	4.82	4.17	6.55
MIP	Z_{min}	–	–	–	–	–	–	–	–	–	–
	Time (ms)	t.o.	t.o.	t.o.	t.o.	t.o.	t.o.	t.o.	t.o.	t.o.	t.o.

FIGURE 4: Algorithm comparisons for instances 4.10, 5.10, and 6.5.

FIGURE 5: Algorithm comparisons for instances A.5, B.5, C.5, and D.5.

FIGURE 6: Algorithm comparisons for instances NRE.5, NRE.5, NRG.5, and NRH.5.

TABLE 8: Computational results for instances of group NRH.

Instance		NRH.1	NRH.2	NRH.3	NRH.4	NRH.5
Z_{opt}		63	63	59	58	55
BH-AS	Z_{min}	65	67	64	63	62
	Z_{avg}	67	68	65	64	62
	RPD	0.032	0.063	0.085	0.086	0.127
	$Time$	17911.2	19733.4	21840.3	21421.7	22242.5
BCSO	Z_{min}	70	67	68	66	61
	Z_{avg}	71	67	70	67	62
	RPD	11.1	6.3	15.3	13.8	10.9
BFO	Z_{min}	69	66	65	63	59
	Z_{avg}	70	66	67	65	60
	RPD	9.52	4.76	10.16	6.77	7.27
BSFLA	Z_{min}	68	66	62	63	59
	Z_{avg}	69	66	63	64	61
	RPD	7.94	4.76	5.08	8.62	7.27
BELA	Z_{min}	70	71	68	70	69
	Z_{avg}	71	71	70	72	69
	RPD	11.11	12.70	15.25	20.69	25.45
BABC	Z_{min}	70	69	66	64	60
	Z_{avg}	71	72	67	64	61
	RPD	11.11	9.52	11.86	10.34	9.09
MIP	Z_{min}	–	–	–	–	–
	$Time (ms)$	t.o.	t.o.	t.o.	t.o.	t.o.

H_0 cannot be assumed. Again, bold font and underlined row are used for a winner value of the metaheuristic stated in the column of the table, e.g., for instance 4.1, BH-AS is better than basic algorithm as its value is lower than 0.05, then H_0 cannot be assumed.

According to results, for p -values lower than 0.05 for black hole optimization are 12 and for BH-AS are 40. The rest does not provide significant information. This results illustrating that the performance of BH-AS is better than the original approach.

5.3. BH-AS v/s Other Optimization Techniques. To evidence the performance of our adaptive approach, we perform a comparison with different approximation techniques: binary cat swarm optimization (BCSO) [41], binary firefly optimization (BFO) [40], binary shuffled frog leaping algorithm (BSFLA) [42], binary artificial bee colony (BABC) [43], and binary electromagnetism-like algorithm (BELA) [66].

We additionally incorporate a comparative by using Mixed Integer Linear Programming (MIP) as exact solving method implemented on MiniZinc G12 MIP. With the solver, the instances are solved to a maximum time of 8 hours. If no solution is found at this point the problem is set to t.o. (time-out).

Tables 6, 7, and 8 illustrate that the proposed approach is able to reach competitive results in contrast to those modern optimization techniques.

For group 4, the adaptive approach shows outstanding behaviour reaching 80% of the total optimum values, while

BFO is the only one finding optimum values. However, if we compare the adaptive black hole with MIP, results illustrate the exact method is more efficient than the adapted algorithm in terms of solving time.

Considering groups 5 and 6, we can observe the adaptive black hole algorithm is able of find all optimum values, in reduce solving times. Comparing results with others metaheuristics, we can state that our approach is better than them, outperforming the MIP in terms of solving time.

Finally, for biggest instances—from group A to group NRH—the exact method MIP becomes incapable to find the solution (time-out). Figures 4, 5, and 6 illustrate the comparative among the adaptive approach and approximate algorithms when they are solving hardest instances of each group.

If we observe instances 4.10 and 6.5, we can see the BH-AS is better than BCSO and BELA. Now, for instance D.5, BH-AS is better than all studied approximates techniques. Now, to solve NRG.5 results indicate that the BH-AS method is the best alternative. Last, if we compare BH-AS v/s MIP, we can say that our approach overcomes the exact method due to this is not able to solve the hardest groups.

6. Conclusions and Future Work

In this paper, we have presented an adaptive approach for black hole algorithm to solve different instances of the set covering problem. This approach is based on adaptive for population size parameter which is valued before the

TABLE 9: Sampling phase: hardest instances for groups 4, 5, 6, A, and B.

ID	Hardest instances		Discretization methods			
	Optimum values	Transfer functions	Standard	Complement	Static probability	Elitist
4.10	514	$\mathcal{S}_1$	515	514	516	515
		$\mathcal{S}_2$	516	515	515	517
		$\mathcal{S}_3$	516	516	517	514
		$\mathcal{S}_4$	515	515	515	514
		$\mathcal{V}_1$	515	516	517	516
		$\mathcal{V}_2$	514	517	515	515
		$\mathcal{V}_3$	516	517	516	515
		$\mathcal{V}_4$	514	515	516	515
5.10	265	$\mathcal{S}_1$	265	267	266	266
		$\mathcal{S}_2$	265	267	266	266
		$\mathcal{S}_3$	266	266	265	265
		$\mathcal{S}_4$	265	266	265	265
		$\mathcal{V}_1$	265	267	265	268
		$\mathcal{V}_2$	265	265	267	268
		$\mathcal{V}_3$	265	268	265	266
		$\mathcal{V}_4$	265	267	267	266
6.5	161	$\mathcal{S}_1$	164	163	162	162
		$\mathcal{S}_2$	162	163	163	164
		$\mathcal{S}_3$	162	163	161	164
		$\mathcal{S}_4$	164	161	162	163
		$\mathcal{V}_1$	161	164	163	161
		$\mathcal{V}_2$	161	162	163	163
		$\mathcal{V}_3$	162	163	161	163
		$\mathcal{V}_4$	162	162	161	163
A.5	236	$\mathcal{S}_1$	239	238	239	238
		$\mathcal{S}_2$	238	236	238	237
		$\mathcal{S}_3$	239	237	237	237
		$\mathcal{S}_4$	236	238	236	237
		$\mathcal{V}_1$	239	236	237	239
		$\mathcal{V}_2$	236	239	237	238
		$\mathcal{V}_3$	236	237	239	237
		$\mathcal{V}_4$	238	239	239	237

run of the metaheuristic. To this end, we use autonomous search which is a particular case of adaptive systems that improve their solving performance by modifying and adjusting themselves to the problem at hand, either by adaptation or supervised adaptation. We have added to the core algorithm an effective preprocessing process that allows filtering and discarding values leading to unfeasible solutions. On the other hand, we include a set of binarization strategies to adapt the black hole algorithm to binary domain. We have tested 65 nonunicost instances from the Beasley's OR-Library where several global optimum values which were not reached using by the basic black algorithm were achieved via the autoadaptive approach. We have also compared the proposed adaptive approach by using a nonparametric statistical tests and the results are conclusive.

As future work, we plan to experiment autoadaptive approaches in recent bioinspired algorithms and to provide a

larger comparison of techniques for solving the set covering problem. The integration of autonomous search can led the research towards new study lines, such as dynamically selecting the best binarization strategy during solving according to performance indicators as analogously studied in [67, 68].

Appendix

Sampling phase allows us to know in a preliminary way which binarization strategy exhibits the better result. Tables 9 and 10 show results when we tested all binarization strategy. Moreover, we include a set of charts (Figures 7 and 8) that summarize the obtained computational experiments. If we observe, the combination $\mathcal{S}_1$ and Elitist has a greater number of convergences towards the lowest known value. For this reason, we have decided to use this binarization strategy for all experiments.

TABLE 10: Sampling phase: hardest instances for groups C, D, NRE, NRF, BRG, and NRH.

Hardest instances		Transfer functions	Discretization methods			
ID	Optimum values		Standard	Complement	Static probability	Elitist
B.5	72	$\mathcal{S}_1$	75	75	73	74
		$\mathcal{S}_2$	73	74	75	72
		$\mathcal{S}_3$	75	74	74	73
		$\mathcal{S}_4$	75	73	72	74
		$\mathcal{V}_1$	73	73	74	75
		$\mathcal{V}_2$	74	72	72	74
		$\mathcal{V}_3$	74	74	72	75
		$\mathcal{V}_4$	75	74	72	75
C.5	215	$\mathcal{S}_1$	217	215	216	216
		$\mathcal{S}_2$	217	217	218	218
		$\mathcal{S}_3$	216	216	218	217
		$\mathcal{S}_4$	215	218	217	217
		$\mathcal{V}_1$	216	216	218	217
		$\mathcal{V}_2$	215	216	217	216
		$\mathcal{V}_3$	216	217	215	216
		$\mathcal{V}_4$	216	218	215	217
D.5	61	$\mathcal{S}_1$	61	61	61	63
		$\mathcal{S}_2$	62	61	63	64
		$\mathcal{S}_3$	64	63	64	63
		$\mathcal{S}_4$	64	61	63	62
		$\mathcal{V}_1$	61	61	61	64
		$\mathcal{V}_2$	63	62	64	64
		$\mathcal{V}_3$	61	63	62	62
		$\mathcal{V}_4$	62	63	62	62
NRE.5	28	$\mathcal{S}_1$	30	29	31	30
		$\mathcal{S}_2$	29	31	28	31
		$\mathcal{S}_3$	31	28	30	28
		$\mathcal{S}_4$	28	31	28	28
		$\mathcal{V}_1$	30	31	31	29
		$\mathcal{V}_2$	29	29	29	28
		$\mathcal{V}_3$	29	30	28	28
		$\mathcal{V}_4$	31	30	30	28
NRF.5	13	$\mathcal{S}_1$	14	15	15	14
		$\mathcal{S}_2$	15	16	14	14
		$\mathcal{S}_3$	16	13	14	15
		$\mathcal{S}_4$	14	16	14	15
		$\mathcal{V}_1$	16	14	16	16
		$\mathcal{V}_2$	15	16	15	14
		$\mathcal{V}_3$	13	16	14	14
		$\mathcal{V}_4$	13	14	15	13
NRG.5	168	$\mathcal{S}_1$	170	171	169	171
		$\mathcal{S}_2$	170	171	171	171
		$\mathcal{S}_3$	169	168	168	169
		$\mathcal{S}_4$	170	171	171	169
		$\mathcal{V}_1$	168	169	171	169
		$\mathcal{V}_2$	170	169	170	171
		$\mathcal{V}_3$	170	171	171	168
		$\mathcal{V}_4$	171	169	170	169
NRH.5	55	$\mathcal{S}_1$	58	57	58	56
		$\mathcal{S}_2$	56	58	58	55
		$\mathcal{S}_3$	58	56	57	57
		$\mathcal{S}_4$	57	57	55	57
		$\mathcal{V}_1$	58	57	58	55
		$\mathcal{V}_2$	55	55	56	57
		$\mathcal{V}_3$	55	58	56	58
		$\mathcal{V}_4$	56	56	56	56

FIGURE 7: Sampling phase. Hardest instances for groups 4, 5, 6, A, B, and C.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

Ricardo Soto is supported by Grant CONICYT/FONDECYT/REGULAR/1160455. Broderick Crawford is supported

FIGURE 8: Sampling phase. Hardest instances for groups D, NRE, NRE, NRG, and NRH.

by Grant CONICYT/FONDECYT/REGULAR/1171243. Carla Taramasco and Rodrigo Olivares are supported by CONICYT/FONDEF/IDeA/ID16110449, FONDECYT/STIC-AMSUD/17STIC-03, FONDECYT/MEC/MEC80170097, and

CORFO-CENS 16CTTS-66390 through the National Center on Health Information Systems. Moreover, Rodrigo Olivares is supported by Postgraduate Grant Pontificia Universidad Católica de Valparaíso (INF-PUCV 2015-2017).

References

- [1] X. Yang, "Nature-inspired metaheuristic algorithms: success and new challenges," *Journal of Computer Engineering and Information Technology*, vol. 1, no. 1, 2012.
- [2] A. P. Piotrowski, M. J. Napiorkowski, J. J. Napiorkowski, and P. M. Rowinski, "Swarm Intelligence and Evolutionary Algorithms: Performance versus speed," *Information Sciences*, vol. 384, pp. 34–85, 2017.
- [3] Y. Hamadi, E. Monfroy, and F. Saubion, "What is autonomous search?" in *Hybrid optimization*, vol. 45 of *Springer Optimization and Its Applications*, pp. 357–391, Springer, New York, NY, USA, 2011.
- [4] D. Gong, X. Ji, J. Sun, and X. Sun, "Interactive evolutionary algorithms with decision-maker's preferences for solving interval multi-objective optimization problems," in *Communications in Computer and Information Science*, pp. 23–29, Springer, Berlin, Germany, 2012.
- [5] Y. Liu, D. Gong, J. Sun, and Y. Jin, "A Many-Objective Evolutionary Algorithm Using A One-by-One Selection Strategy," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2689–2702, 2017.
- [6] J. Sun, D. Gong, X. Zeng, and N. Geng, "An ensemble framework for assessing solutions of interval programming problems," *Information Sciences*, vol. 436/437, pp. 146–161, 2018.
- [7] D. Gong, J. Sun, and Z. Miao, "A set-based genetic algorithm for interval many-objective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 47–60, 2018.
- [8] J. Sun, D. Gong, and X. Sun, "Solving interval multi-objective optimization problems using evolutionary algorithms with preference polyhedron," in *Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference (GECCO '11)*, vol. 233, pp. 729–736, ACM Press, Dublin, Ireland, July 2011.
- [9] D.-W. Gong, N.-N. Qin, and X.-Y. Sun, "Evolutionary algorithms for optimization problems with uncertainties and hybrid indices," *Information Sciences*, vol. 181, no. 19, pp. 4124–4138, 2011.
- [10] T. Roeper and E. Williams, "The theory of parameters and syntactic development," in *Parameter Setting*, pp. 191–215, Springer, Netherlands, 1987.
- [11] Á. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.
- [12] G. Karafotias, M. Hoogendoorn, and A. E. Eiben, "Parameter Control in Evolutionary Algorithms: Trends and Challenges," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 167–187, 2015.
- [13] C. Salto and E. Alba, "Designing heterogeneous distributed GAs by efficiently self-adapting the migration period," *Applied Intelligence*, vol. 36, no. 4, pp. 800–808, 2012.
- [14] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 1785–1791, Edinburgh, Scotland, UK, September 2005.
- [15] W. Yi, L. Gao, X. Li, and Y. Zhou, "A new differential evolution algorithm with a hybrid mutation operator and self-adapting control parameters for global optimization problems," *Applied Intelligence*, vol. 42, no. 4, pp. 642–660, 2015.
- [16] M. Han, S. Liao, J. Chang, and C. Lin, "Dynamic group-based differential evolution using a self-adaptive strategy for global optimization problems," *Applied Intelligence*, vol. 39, no. 1, pp. 41–56, 2013.
- [17] K. Liang, X. Yao, and C. S. Newton, "Adapting self-adaptive parameters in evolutionary algorithms," *Applied Intelligence*, vol. 15, no. 3, pp. 171–180, 2001.
- [18] T. Stutzle, M. Lopez-Ibanez, P. Pellegrini et al., "Parameter adaptation in ant colony optimization," in *What Is Autonomous Search?* pp. 191–215, Springer, Berlin, Germany, 2012.
- [19] J. Olamaei, M. Moradi, and T. Kaboodi, "A new adaptive modified firefly algorithm to solve optimal capacitor placement problem," in *Proceedings of the 18th Electric Power Distribution Network Conference, EPDC 2013*, Kermanshah, Iran, May 2013.
- [20] X. Li and M. Yin, "Self-adaptive constrained artificial bee colony for constrained numerical optimization," *Neural Computing and Applications*, vol. 24, no. 3–4, pp. 723–734, 2014.
- [21] S. Mahmoudi and S. Lotfi, "Modified cuckoo optimization algorithm (MCOA) to solve graph coloring problem," *Applied Soft Computing*, vol. 33, pp. 48–64, 2015.
- [22] T. T. Nguyen and D. N. Vo, "Modified cuckoo search algorithm for short-term hydrothermal scheduling," *International Journal of Electrical Power & Energy Systems*, vol. 65, pp. 271–281, 2015.
- [23] M. Affenzeller, S. Wagner, and S. Winkler, "Self-adaptive population size adjustment for genetic algorithms," in *Computer Aided Systems Theory EUROCAST 2007*, pp. 820–828, Springer, Berlin, Germany, 2007.
- [24] X. Li and M. Yin, "Modified cuckoo search algorithm with self adaptive parameter method," *Information Sciences*, vol. 298, pp. 80–97, 2015.
- [25] B. Akay and D. Karaboga, "A modified Artificial Bee Colony algorithm for real-parameter optimization," *Information Sciences*, vol. 192, pp. 120–142, 2012.
- [26] A. Caprara, M. Fischetti, and P. Toth, "Algorithms for the set covering problem," *Annals of Operations Research*, vol. 98, pp. 353–371, 2000.
- [27] E. Balas, "A dynamic subgradient-based branch-and-bound procedure for set covering," *Location Science*, vol. 5, no. 3, article 203, 1997.
- [28] J. E. Beasley, "An algorithm for set covering problem," *European Journal of Operational Research*, vol. 31, no. 1, pp. 85–93, 1987.
- [29] B. Yelbay, S. I. Birbil, and K. Bülbül, "The set covering problem revisited: an empirical study of the value of dual information," *Journal of Industrial and Management Optimization*, vol. 11, no. 2, pp. 575–594, 2015.
- [30] R. A. Rushmeier and G. L. Nemhauser, "Experiments with parallel branch-and-bound algorithms for the set covering problem," *Operations Research Letters*, vol. 13, no. 5, pp. 277–285, 1993.
- [31] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979.
- [32] B. Crawford, R. Soto, E. Monfroy, F. Paredes, and W. Palma, "A hybrid Ant algorithm for the set covering problem," *International Journal of Physical Sciences*, vol. 6, no. 19, pp. 4667–4673, 2011.
- [33] Z.-G. Ren, Z.-R. Feng, L.-J. Ke, and Z.-J. Zhang, "New ideas for applying ant colony optimization to the set covering problem," *Computers & Industrial Engineering*, vol. 58, no. 4, pp. 774–784, 2010.
- [34] R. Hadji, M. Rahoual, E. Talbi, and V. Bachelet, "Ant colonies for the set covering problem," in *Proceedings of ANTS 2000*, pp. 63–66, 2015.
- [35] L. Lessing, I. Dumitrescu, and T. Stutzle, "A comparison between ACO algorithms for the set covering problem," in *Ant*

- Colony Optimization and Swarm Intelligence*, vol. 3172 of *Lecture Notes in Computer Science*, pp. 1–12, Springer, Berlin, Germany, 2004.
- [36] M. J. Brusco, L. W. Jacobs, and G. M. Thompson, “A morphing procedure to supplement a simulated annealing heuristic for cost- and coverage-correlated set-covering problems,” *Annals of Operations Research*, vol. 86, pp. 611–627, 1999.
 - [37] M. Caserta, “Tabu search-based metaheuristic algorithm for large-scale set covering problems,” in *Metaheuristics: Progress in Complex Systems Optimization*, pp. 43–63, Springer, Boston, Mass, USA, 2007.
 - [38] J. E. Beasley and P. C. Chu, “A genetic algorithm for the set covering problem,” *European Journal of Operational Research*, vol. 94, no. 2, pp. 392–404, 1996.
 - [39] Y. Lu and F. J. Vasko, “An OR Practitioner’s Solution Approach for the Set Covering Problem,” *International Journal of Applied Metaheuristic Computing*, vol. 6, no. 4, pp. 1–13, 2015.
 - [40] B. Crawford, R. Soto, M. Olivares-Suárez, and F. Paredes, “A binary firefly algorithm for the set covering problem,” *Advances in Intelligent Systems and Computing*, vol. 285, pp. 65–73, 2014.
 - [41] B. Crawford, R. Soto, N. Berríos et al., “A binary cat swarm optimization algorithm for the non-unicost set covering problem,” *Mathematical Problems in Engineering*, vol. 2015, Article ID 578541, 8 pages, 2015.
 - [42] B. Crawford, R. Soto, C. Peña, W. Palma, F. Johnson, and F. Paredes, “Solving the Set Covering Problem with a Shuffled Frog Leaping Algorithm,” in *Intelligent Information and Database Systems*, vol. 9012 of *Lecture Notes in Computer Science*, pp. 41–50, Springer International Publishing, Cham, Switzerland, 2015.
 - [43] R. Cuesta, B. Crawford, R. Soto, and F. Paredes, “An Artificial Bee Colony Algorithm for the Set Covering Problem,” in *Modern Trends and Techniques in Computer Science*, vol. 285 of *Advances in Intelligent Systems and Computing*, pp. 53–63, Springer International Publishing, Cham, Switzerland, 2014.
 - [44] R. Soto, B. Crawford, R. Olivares et al., “Solving the non-unicost set covering problem by using cuckoo search and black hole optimization,” *Natural Computing*, vol. 16, no. 2, pp. 213–229, 2017.
 - [45] M. Salveson, “The assembly line balancing problem,” *Journal of Industrial Engineering*, vol. 6, pp. 18–25, 1955.
 - [46] E. K. Baker, L. D. Bodin, W. F. Finnegan, and R. J. Ponder, “Efficient Heuristic Solutions to an Airline Crew Scheduling Problem,” *A I I E Transactions*, vol. 11, no. 2, pp. 79–85, 2007.
 - [47] I. Bartholdi, “A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering,” *Operations Research*, vol. 29, no. 3, pp. 501–510, 1981.
 - [48] J. Rubin, “Technique for the solution of massive set covering problems, with application to airline crew scheduling,” *Transportation Science*, vol. 7, no. 1, pp. 34–48, 1973.
 - [49] C. Toregas, R. Swain, C. ReVelle, and L. Bergman, “The location of emergency service facilities,” *Operations Research*, vol. 19, no. 6, pp. 1363–1373, 1971.
 - [50] W. Walker, “Using the set-covering problem to assign fire companies to fire houses,” *Operations Research*, vol. 22, no. 2, pp. 275–277, 1974.
 - [51] K. Munagala, S. Babu, R. Motwani, and J. Widom, “The pipelined set cover problem,” in *Database Theory—ICDT 2005*, vol. 3363 of *Lecture Notes in Computer Science*, pp. 83–98, Springer, Berlin, Germany, 2005.
 - [52] S. Helber, K. Schimmelpfeng, R. Stolletz, and S. Lagershausen, “Using linear programming to analyze and optimize stochastic flow lines,” *Annals of Operations Research*, vol. 182, pp. 193–211, 2011.
 - [53] M. A. Breuer, “Simplification of the covering problem with application to Boolean expressions,” *Journal of the ACM*, vol. 17, pp. 166–181, 1970.
 - [54] A. Hatamlou, “Black hole: a new heuristic optimization approach for data clustering,” *Information Sciences*, vol. 222, pp. 175–184, 2013.
 - [55] S. Kumar, D. Datta, and S. K. Singh, “Black hole algorithm and its applications,” in *Studies in Computational Intelligence*, vol. 575, pp. 147–170, Springer International Publishing, Cham, Switzerland, 2015.
 - [56] Y. Hamadi, E. Monfroy, and F. Saubion, *Autonomous Search*, Springer, Berlin, Germany, 2012.
 - [57] R. Soto, B. Crawford, R. Olivares et al., “Online control of enumeration strategies via bat algorithm and black hole optimization,” *Natural Computing*, vol. 16, no. 2, pp. 241–257, 2017.
 - [58] R. Soto, B. Crawford, R. Olivares et al., “Using autonomous search for solving constraint satisfaction problems via new modern approaches,” *Swarm and Evolutionary Computation*, vol. 30, pp. 64–77, 2016.
 - [59] M. Nemati, H. Momeni, and N. Bazrkar, “Binary Black Holes Algorithm,” *International Journal of Computer Applications*, vol. 79, no. 6, pp. 36–42, 2013.
 - [60] S. Mirjalili and A. Lewis, “S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization,” *Swarm and Evolutionary Computation*, vol. 9, pp. 1–14, 2013.
 - [61] Broderick Crawford, Ricardo Soto, Gino Astorga, José García, Carlos Castro, and Fernando Paredes, “Putting Continuous Metaheuristics to Work in Binary Search Spaces,” *Complexity*, vol. 2017, Article ID 8404231, 19 pages, 2017.
 - [62] J. Beasley, “OR-Library,” 1990, <https://goo.gl/IO1UQ6>.
 - [63] M. L. Fisher and P. Kedia, “Optimal solution of set covering/partitioning problems using dual heuristics,” *Management Science*, vol. 36, no. 6, pp. 674–688, 1990.
 - [64] H. W. Lilliefors, “On the Kolmogorov-Smirnov test for normality with mean and variance unknown,” *Journal of the American Statistical Association*, vol. 62, no. 318, pp. 399–402, 1967.
 - [65] H. B. Mann and D. R. Whitney, “On a test of whether one of two random variables is stochastically larger than the other,” *Annals of Mathematical Statistics*, vol. 18, pp. 50–60, 1947.
 - [66] R. Soto, B. Crawford, A. Muñoz, F. Johnson, and F. Paredes, “Pre-processing, repairing and transfer functions can help binary electromagnetism-like algorithms,” in *Artificial intelligence perspectives and applications*, vol. 347 of *Advances in Intelligent Systems and Computing*, pp. 89–97, Springer, Cham, Switzerland, 2015.
 - [67] R. Soto, B. Crawford, S. Misra et al., “Choice functions for autonomous search in constraint programming: GA vs. PSO,” *Tehnički vjesnik*, vol. 20, no. 4, pp. 621–627, 2013.
 - [68] R. Soto, B. Crawford, W. Palma et al., “Boosting autonomous search for CSPs via skylines,” *Information Sciences*, vol. 308, pp. 38–48, 2015.