mathematics

MDPI

Article

Binary Secretary Bird Optimization Algorithm for the Set Covering Problem

Broderick Crawford [1,*], Felipe Cisternas-Caneo [1], Ricardo Soto [1], Claudio Patricio Toledo Mac-lean [1], José Lara Arce [1], Fabián Solís-Piñones [1], Gino Astorga [2] and Giovanni Giachetti [3]

1 Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso, Avenida Brasil 2241, Valparaíso 2362807, Chile; felipe.cisternas.c@mail.pucv.cl (F.C.-C.); ricardo.soto@pucv.cl (R.S.); claudio.toledo.m@mail.pucv.cl (C.P.T.M.-l.); jose.lara.a01@mail.pucv.cl (J.L.A.); fabian.solis.p@mail.pucv.cl (F.S.-P.)

2 Escuela de Negocios Internacionales, Universidad de Valparaíso, Alcalde Prieto Nieto 452, Viña del Mar 2572048, Chile; gino.astorga@uv.cl

3 Facultad de Ingeniería, Universidad Andres Bello, Antonio Varas 880, Providencia, Santiago 7591538, Chile; giovanni.giachetti@unab.cl

* Correspondence: broderick.crawford@pucv.cl

Abstract

The Set Coverage Problem (SCP) is an important combinatorial optimization problem known to be NP-complete. The use of metaheuristics to solve the SCP includes different algorithms. In particular, binarization techniques have been explored to adapt metaheuristics designed for continuous optimization problems to the binary domain of the SCP. In this work, we present a new approach to solve the SCP based on the Secretary Bird Optimization Algorithm (SBOA). This algorithm is inspired by the natural behavior of the secretary bird, known for its ability to hunt prey and evade predators in its environment. Since the SBOA was originally designed for optimization problems in continuous space and the SCP is a binary problem, this paper proposes the implementation of several binarization techniques to adapt the algorithm to the discrete domain. These techniques include eight transfer functions and five different discretization methods. Taken together, these combinations create multiple SBOA adaptations that effectively balance exploration and exploitation, promoting an adequate distribution in the search space. Experimental results applied to the SCP together with its variant Unicost SCP and compared to Grey Wolf Optimizer and Particle Swarm Optimization suggest that the binary version of SBOA is a robust algorithm capable of producing high quality solutions with low computational cost. Given the promising results obtained, it is proposed as future work to focus on complex and large-scale problems as well as to optimize their performance in terms of time and accuracy.

Keywords: combinatorial optimization; metaheuristic; bio-inspired algorithm; secretary bird optimization algorithm; binarization; set covering problem

MSC: 68T20; 68W25; 90C27; 90C59; 68Q25

check for updates

Academic Editor: Iliya Bouyukliev

Received: 9 June 2025

Revised: 23 July 2025

Accepted: 24 July 2025

Published: 1 August 2025

Citation: Crawford, B.; Cisternas-Caneo, F.; Soto, R.; Mac-lean, C.P.T.; Lara Arce, J.; Solís-Piñones, F.; Astorga, G.; Giachetti, G. Binary Secretary Bird Optimization Algorithm for the Set Covering Problem. *Mathematics* **2025**, *13*, 2482. https://doi.org/10.3390/math13152482

1. Introduction

The use of optimization and, in particular, metaheuristics, is gaining more and more attention in modern industry as it allows to tackle problems that, until some time ago, were not possible to address. Today, it is possible to tackle complex and large-scale problems, with metaheuristics allowing us to obtain good quality solutions in a reasonable processing time [1]. Metaheuristics, in general, allow us to address different types of problems and can also adapt to changes in the environment, being able to work in different domains. All these characteristics make them attractive for solving current real-world problems in a variety of areas, such as logistics [2,3], manufacturing industry [4,5], transport [6,7], health [8,9], and mining [10,11], among others.

The Set Covering Problem (SCP) is a combinatorial optimization problem known to be NP-hard that is ever-present in various industries [12] and one of the 21 problems present in Karp [13]. It involves finding a subset of columns from a binary matrix A of size $m \times n$ that covers all rows at the lowest possible cost. This problem has significant applications in various fields, such as emergency service location [14], crew scheduling in mass transportation systems [15], and other related problems discussed in [16]. There is a special case of SCP where all sets have the same cost in this case the task is to minimize the selection of sets to cover the given universe. Practical applications in the industry are engineering design [17] and vehicle path planning [18], among others.

The use of metaheuristics to solve the SCP includes algorithms such as Genetic Algorithms [19], Particle Swarm Optimization [20], Ant Colony Optimization [21], Tabu Search [22], Electromagnetism metaheuristic [23], Artificial Bee Colony [24], and GRASP [25], among others. In particular, binarization techniques have been explored to adapt metaheuristics designed for continuous optimization problems to the binary domain of the SCP.

In this work, we present a new approach to solving the SCP based on the Secretary Bird Optimization Algorithm (SBOA) [26]. The SBOA was selected for this research for several compelling reasons, drawn directly from its original design and validated performance. First, it is a recent metaheuristic whose authors specifically engineered it to address common shortcomings of existing algorithms, aiming to improve convergence speed, enhance optimization accuracy, and effectively avoid local optima. These characteristics are highly desirable for tackling NP-hard problems like the SCP. Second, the SBOA achieves a robust balance between its exploration and exploitation phases by uniquely combining three different search strategies: differential evolution, Brownian motion, and Lévy flights. Finally, its effectiveness is not merely theoretical; the SBOA has demonstrated superior performance against numerous advanced algorithms on standard benchmark suites (CEC-2017 and CEC-2022) and has been successfully applied to solve twelve different constrained engineering problems.

Given these strengths, the SBOA emerges as a strong and novel candidate for adaptation to the binary domain to solve the SCP. The relevance of finding new and efficient solutions for this problem is underscored by its wide range of applications, which have become even more critical in modern industry. For example, in the post-pandemic era, supply chains and logistics face continuous challenges where the SCP is present [27], including the efficient allocation of resources and crews [28,29], balancing assembly lines, and optimizing the location of distribution centers to meet growing demand.

Since the SBOA was originally designed for optimization problems in continuous spaces and the SCP is a binary problem, this work proposes the implementation of several binarization techniques to adapt the algorithm to the discrete domain. These techniques include eight transfer functions with "S"-shaped and "V"-shaped variants and five different discretization methods. Together, these combinations create multiple adaptations of the SBOA that effectively balance exploration and exploitation, promoting an adequate distribution in the search space. The results suggest that the binary version of the SBOA (BSBOA) is a robust algorithm capable of producing high-quality solutions with low computational cost.

This article is organized as follows: We present a brief description of the Set Coverage Problem in Section 2, how to resolve SCP with continuos metaheuristics in Section 3, and an outline of the Secretary Bird Optimitation Algorithm in Section 4. We explain how and make the binary version of the SBOA in Section 5. Finally, we present our results, discussions, conclusions, and possible future lines of research in Sections 6–8.

2. Set Covering Problem

The Set Covering Problem (SCP) is a combinatorial optimization problem categorized as NP-hard. Its main objective is to find a minimal subset of elements that completely covers a universal set of requirements while minimizing the associated cost.

2.1. Formal Mathematical Formulation

Formally, let $U = \{u_1, u_2, \ldots, u_m\}$ be a universal set composed of m elements, and let $S = \{S_1, S_2, \ldots, S_n\}$ be a collection of n subsets, where each subset S_j has an associated cost c_j. The problem consists of identifying an optimal subset $S^* \subseteq S$ that covers all elements in U at the minimum possible cost.

To model this problem and solve it computationally, the formal set-based definition is translated into a matrix formulation. A binary matrix represents the problem A with m rows and n columns. Each of the m rows corresponds to an element u_i of the universal set U and each of the n columns represents an available subset S_j. The value in the matrix, $a_{i,j}$, is 1 if subset j covers element i (that is, $u_i \in S_j$), and 0 otherwise.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

To decide which columns (subsets) are chosen, the binary decision variable x_j is used, which takes the value of one if column j is selected for the solution, and zero if it is not.

Thus, the main objective is to minimize the total cost of the solution, summing the costs of only the selected columns, as indicated by Equation (1).

$$\min \sum_{j=1}^{n} c_j x_j \tag{1}$$

The main constraint of this problem is the coverage constraint, which ensures that each element (row) is covered by at least one selected column. This is achieved with the following equation, which must hold for every row i:

$$\sum_{j=1}^{n} a_{i,j} x_j \geq 1 \tag{2}$$

In addition, we must ensure that the decision variable is binary; therefore, $x_j \in \{0, 1\}$.

2.2. SCP Practical Example

Let us assume a company needs to form an oversight committee for a new technology project. For the project to be successful, the committee must cover five areas of expertise: financial, legal, technical, marketing, and logistics. The company has identified six external consultants, each with a different set of skills and an associated cost (fee).

The goal is to apply the Set Covering Problem (SCP) to select the committee with the lowest total cost that guarantees coverage of all required areas of expertise.

2.2.1. Problem Definition

1. Elements to be Covered: The five required areas of expertise.
    - R_1: Financial Expertise.
    - R_2: Legal Expertise.
    - R_3: Technical Expertise.
    - R_4: Marketing Expertise.
    - R_5: Logistics Expertise.
2. Available Sets: The six candidates, each with their cost and skills.
    - C_1: Anne (Cost: $4)—Skills: Financial, Legal.
    - C_2: Ben (Cost: $3)—Skills: Technical, Marketing.
    - C_3: Carla (Cost: $6)—Skills: Legal, Technical, Logistics.
    - C_4: David (Cost: $4)—Skills: Marketing, Logistics.
    - C_5: Elena (Cost: $5)—Skills: Financial, Technical.
    - C_6: Frank (Cost: $4)—Skills: Legal, Marketing.

Thus, Table 1 shows the relationship between both previous lists in the so-called Incidence Matrix. This matrix reflects the coverage (columns) of each element (rows) and allows us to validate whether the requirements (problem constraints) are met.

Table 1. Incidence Matrix (A), where $a_{ij} = 1$ if consultant j covers requirement i.

	C1 (Anne)	**C2 (Ben)**	**C3 (Carla)**	**C4 (David)**	**C5 (Elena)**	**C6 (Frank)**
R1 (Fin.)	1	0	0	0	1	0
R2 (Legal)	1	0	1	0	0	1
R3 (Tech.)	0	1	1	0	1	0
R4 (Mkt.)	0	1	0	1	0	1
R5 (Log.)	0	0	1	1	0	0

2.2.2. Mathematical Formulation

Let $x_j = 1$ if consultant j is selected and 0 otherwise. The problem is formulated to minimize the total cost, subject to the constraint that each area of expertise is covered.

- Objective Function (minimize cost):

$$\min(Z) = 4x_1 + 3x_2 + 6x_3 + 4x_4 + 5x_5 + 4x_6 \quad (3)$$

- Constraints (cover each expertise):

$$\begin{aligned} x_1 + x_5 &\geq 1 && \text{(Financial)} \\ x_1 + x_3 + x_6 &\geq 1 && \text{(Legal)} \\ x_2 + x_3 + x_5 &\geq 1 && \text{(Technical)} \\ x_2 + x_4 + x_6 &\geq 1 && \text{(Marketing)} \\ x_3 + x_4 &\geq 1 && \text{(Logistics)} \end{aligned}$$

- Decision Variables:

$$x_j \in \{0, 1\} \quad \forall j \in \{1, \ldots, 6\} \quad (4)$$

2.2.3. Optimal Solution

The optimal solution for this problem is to select Anne (C_1), Ben (C_2), and David (C_4).

- Selected Consultants: {Anne, Ben, David}
- Coverage:
    - Anne covers the Financial and Legal areas.

- Ben covers the Technical and Marketing areas.
- David covers the Marketing and Logistics areas.

- Minimum Total Cost: \$4 (Anne) + \$3 (Ben) + \$4 (David) = \$**11**.

  This combination fulfills all expertise requirements at the lowest possible cost.

2.3. Unicost Set Covering Problem (Unicost SCP)

The unicost problem is a specific variant of the Set Covering Problem (SCP), in which all column costs are equal ($c_j = 1$ for all $j \in J$). The primary objective in this case is to minimize the number of selected columns while ensuring that each row is covered by at least one of them. This variant simplifies the general SCP model by focusing exclusively on the minimum number of columns required to cover all rows, emphasizing structural optimization without considering cost variations.

Mathematically, the model can be formulated as follows:

$$\text{Minimize} \quad \sum_{j \in J} x_j \tag{5}$$

subject to

$$\sum_{j \in J} a_{ij} x_j \geq 1 \quad \text{for all } i \in I, \tag{6}$$

$$_j \in \{0, 1\} \quad \text{for all } j \in J, \tag{7}$$

where a_{ij} is a coefficient indicating whether column j covers row i, x_j is a binary variable that takes the value 1 if column j is selected and 0 otherwise, I is the set of rows, and J is the set of columns.

The unicost problem, like the general SCP, is known to be NP-hard and has been applied in various fields, such as scheduling, logistics, and resource optimization.

3. Continuous Metaheuristics Solving Set Covering Problem

To apply the Secretary Bird Optimization Algorithm (SBOA) to the Set Covering Problem (SCP), it is necessary to adapt the original search approach, designed for a continuous space, to a binary environment. As discussed in Section 1, various metaheuristics have been successfully applied to solve the SCP, including Genetic Algorithms [30], Ant Colony Optimization [31], Particle Swarm Optimization [32], and others. All these metaheuristics have one characteristic in common: they are metaheuristics designed to solve continuous optimization problems that were modified to resolve binary problems like the SCP.

This adaptation is achieved through a Two-Step Technique that converts continuous solutions into binary solutions, ensuring compliance with the SCP requirements. This technique is fundamental for applying bioinspired algorithms to binary combinatorial optimization problems, such as the SCP.

Two-Step Technique

In the literature, there are different ways to binarize continuous metaheuristics [33], but the most widely used is the Two-Step Technique [34]. As its name suggests, the binarization process is performed in two stages:

- Application of a transfer function, which transforms the continuous value into a value within the range $[0, 1]$.
- Application of a binarization rule, which determines the assignment of a 1 or a 0.

In the literature [33], a variety of transfer functions are proposed for this purpose. They are generally categorized into two main families based on their shape and behavior:

S-shaped and V-shaped functions. These two families represent the most common and well-established approaches for mapping continuous search spaces to binary ones.

- S-Shaped Functions: These sigmoidal functions produce a value in the range $[0, 1]$ that represents the probability of a solution's component becoming '1'. An input value close to zero yields a probability near 0.5, while large positive or negative values push the probability towards 1 or 0, respectively. This behavior models a form of probabilistic switch.
- V-Shaped Functions: In contrast, these functions relate the probability of change to the magnitude of the continuous value, rather than its sign. A small step (a value close to zero) results in a low probability of changing the bit, while a large step increases this probability. This is conceptually linked to the notion of velocity or momentum in swarm algorithms, where a larger "move" is more likely to alter the solution's state.

Table 2 and Figure 1 present the standard functions from both families that are commonly used in this research area. The notation d_j^i observed in the table corresponds to the continuous value of the j-th dimension of the i-th individual, resulting from the perturbation performed by the continuous metaheuristic.

Table 2. S-shaped and V-shaped transfer functions.

S-Shaped		V-Shaped	
Name	**Equation**	**Name**	**Equation**
S1	$T(d_j^i) = \frac{1}{1+e^{-2d_j^i}}$	V1	$T(d_j^i) = \left\lvert \operatorname{erf}\left(\sqrt{\frac{\pi}{2}} d_j^i\right) \right\rvert$
S2	$T(d_j^i) = \frac{1}{1+e^{-d_j^i}}$	V2	$T(d_j^i) = \left\lvert \tanh(d_j^i) \right\rvert$
S3	$T(d_j^i) = \frac{1}{1+e^{-d_j^i/2}}$	V3	$T(d_j^i) = \left\lvert \frac{d_j^i}{\sqrt{1+(d_j^i)^2}} \right\rvert$
S4	$T(d_j^i) = \frac{1}{1+e^{-d_j^i/3}}$	V4	$T(d_j^i) = \left\lvert \frac{2}{\pi} \arctan\left(\frac{\pi}{2} d_j^i\right) \right\rvert$

Additionally, in the literature [33], we can find five different binarization rules, of which we can highlight the following:

- Standard (STD): If the condition is satisfied, the standard binarization rule returns 1; otherwise, it returns 0. Mathematically, it is defined as follows:

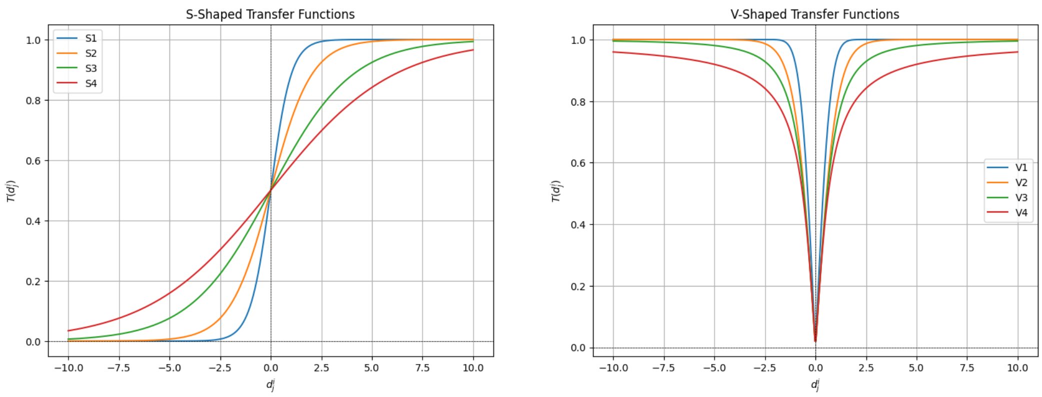
$$X_{\text{new}}^j = \begin{cases} 1 & \text{if rand} \leq T(d_i^j), \\ 0 & \text{else.} \end{cases} \tag{8}$$

- Elitist (ELIT): The best value is assigned if a random value is within the probability; otherwise, a zero value is assigned. Mathematically, it is defined as follows:

$$X_{\text{new}}^j = \begin{cases} X_{\text{Best}}^j & \text{if rand} < T(d_i^j), \\ 0 & \text{else.} \end{cases} \tag{9}$$

- Complement (COM): If the condition is satisfied, the second step operator returns the complement of the actual value.

$$\begin{cases} x_{\text{new}}^j = \text{complement}\left(x_w^j\right) & \text{if rand} \leq T\left(d_w^j\right), \\ 0 & \text{else.} \end{cases} \tag{10}$$



(**a**) S-shaped. (**b**) V-shaped.

Figure 1. S-shaped and V-shaped transfer functions.

4. Secretary Bird Optimization Algorithm

The Secretary Bird Optimization Algorithm (SBOA) is a metaheuristic inspired by the natural behavior of the secretary bird, particularly its unique strategies for hunting and evading predators. It was proposed in 2024 by Youfa Fu, Dan Liu, Jiadui Chen, and Ling He [26]. The algorithm was originally designed to solve continuous optimization problems, leveraging its exploration and exploitation phases to efficiently find optimal solutions.

The operation of the SBOA is divided into two main phases that mimic the bird's survival instincts: exploration, which models its hunting strategy, and exploitation, which models its escape strategy.

4.1. Exploration Phase (Hunting Strategy)

This phase simulates how the secretary bird hunts its prey, like snakes, and corresponds to a global search in the solution space. The process is modeled in three stages that reflect the bird's hunting tactics:

- Searching for Prey: The bird begins by exploring the terrain to locate hidden prey. The algorithm mimics this by creating a new solution based on the difference between two randomly selected solutions from the population. This enhances diversity and allows the algorithm to scan new, unexplored areas of the search space. This movement is modeled by Equation (11).

$$x_{ij}^{\text{new, P1}} = x_{ij} + (x_{\text{random}_1} - x_{\text{random}_2}) \cdot R_1, \tag{11}$$

- Wearing Down the Prey: Once prey is found, the bird does not attack immediately but circles and provokes it to deplete its energy. The algorithm models this by moving towards the best solution found so far (x_{best}), simulating how the bird focuses on its target, while introducing a random component to avoid premature convergence. This is shown in Equation (12).

$$x_{ij}^{\text{new, P1}} = x_{\text{best}} + \exp\left(\left(\frac{t}{T}\right)^4\right) \cdot (RB - 0.5) \cdot (x_{\text{best}} - x_{ij}), \tag{12}$$

- Attacking the Prey: When the prey is exhausted, the bird executes a swift, lethal attack. To simulate this decisive action, the algorithm performs a "jump" toward the best solution using a Lévy flight (RL). This feature models the bird's powerful strike, combining frequent small steps with occasional long jumps to accelerate convergence towards the global optimum. The movement is described in Equation (13).

$$x_{ij}^{\text{new, P1}} = x_{\text{best}} + \left(1 - \frac{t}{T}\right)^{\left(2 \times \frac{t}{T}\right)} \cdot x_{ij} \cdot RL, \tag{13}$$

4.2. Exploitation Phase (Escape Strategy)

This phase models how the secretary bird evades predators, which corresponds to a refined local search for a better solution. The bird chooses one of two strategies with equal probability:

- Camouflage (C_1): The bird may hide in its environment to avoid being detected. The algorithm simulates this by making small, subtle adjustments to its current position, moving locally around the best-known solution to refine it.
- Fleeing (C_2): If camouflage is not an option, the bird runs or flies to escape danger. The algorithm models this with a larger, more random movement that allows it to jump to other regions of the search space, effectively avoiding stagnation.

Both escape strategies are modeled using Equation (14).

$$x_{ij}^{\text{new, evasion}} = \begin{cases} C_1 : x_{\text{best}} + (2 \cdot RB - 1) \cdot \left(1 - \frac{t}{T}\right)^2 \cdot x_{ij}, & \text{if } r < 0.5, \\ C_2 : x_{ij} + R_2 \cdot (x_{\text{random}} - K \cdot x_{ij}), & \text{else.} \end{cases} \tag{14}$$

4.3. Solution Selection

A key aspect of the SBOA is how it determines if a new solution is superior to the current one. The algorithm employs a greedy selection mechanism for this purpose.

The quality of each solution is measured by an objective function (F), which, in the context of the Set Covering Problem, is the total cost to be minimized. After generating a new candidate position in either the exploration or exploitation phase, the algorithm compares the fitness of the new solution ($F_i^{\text{new, P1}}$) with that of the current one (F_i).

The new solution is accepted only if it offers a better fitness value (i.e., a lower cost). Otherwise, the current solution is kept. This process, shown in Equation (15), ensures that the population's quality either improves or remains the same in each iteration, guiding the search effectively towards the optimal solution.

$$X_i = \begin{cases} X_i^{\text{new, P1}} & \text{if } F_i^{\text{new, P1}} < F_i \\ X_i & \text{else} \end{cases} \tag{15}$$

The pseudo-code of the SBOA is detailed in the pseudo-code of Algorithm 1.

Algorithm 1 Secretary Bird Optimization Algorithm.

Input: Problem Setting $(Dim, ub, lb, Pop\_size(N))$, $Max\_Iter(T)$, $Curr\_Iter(t)$
Output: Best solution

Initialize the population randomly.
for $t = 1 \rightarrow T$ **do**
  Update Secretary Bird x_{best}.
  for $i = 1 \rightarrow N$ **do**
    Exploration Phase:
    if $t \leq \frac{1}{3}T$ **then**
      Calculate new status using Searching Stage Equation (11).
      Update the *i*-th Secretary Bird using Update Equation (15).
    else if $\frac{1}{3}T < t < \frac{2}{3}T$ **then**
      Calculate new status using Consuming Stage Equation (12).
      Update the *i*-th Secretary Bird using Update Equation (15).
    else if $\frac{1}{3}T < t < \frac{2}{3}T$ **then**
      Calculate new status using Attacking Stage Equation (13).
      Update the *i*-th Secretary Bird using Update Equation (15).
    end if
    Exploitation Phase:
    if $r < 0.5$ **then**
      Calculate new status using C1 in Exploitation Phase Equation (14).
    else
      Calculate new status using C_2 in Exploitation Phase Equation (14).
    end if
    Update the *i*-th Secretary Bird using Update Equation (15).
  end for
  Output the best solution obtained by SBOA for the given optimization problem.
  Save the best candidate solution so far.
end for
Output the best solution obtained by SBOA for the given optimization problem.

5. Binary Secretary Bird Optimization Algorithm

As explained in Section 4, the Algorithm Optimization based on the Behavior of the Secretary Bird (SBOA) is a metaheuristic designed to solve continuous optimization problems. To address problems such as feature selection, it is necessary to transform the solutions into the binary domain.

Furthermore, in Section 3, it was highlighted that the Two-Step Technique is one of the most widely used approaches for binarizing continuous metaheuristics. In [33,34], eight different transfer functions and five binarization rules are described, which can be applied in this context.

In this work, the binarization scheme was chosen based on established findings in the literature that analyze the relationship between different transfer functions, binarization rules, and the exploration–exploitation balance. For the experiments, we use the V3 transfer function and the Elitist discretization method, considering the work carried out by Lanza-Gutierrez et al. in [35], where they recommend the use of V3 to solve small and medium-sized problems and for a better exploration–exploitation relation recommend Elitist as a discretization technique.

Additionally, we selected a function from the V-shaped family because its behavior is particularly well-suited for swarm intelligence algorithms. Unlike S-shaped functions, V-shaped functions relate the probability of a bit changing to the magnitude (or "velocity") of the agent's move, not its direction. A larger step size corresponds to a higher probability of flipping a bit. Among these, the V3 transfer function, shown in Equation (16), was chosen due to recommendations in the literature for its robust performance on small and medium-sized problems.

$$T(d_j^i) = \left| \frac{d_j^i}{\sqrt{1 + (d_j^i)^2}} \right| \quad (16)$$

To complement the transfer function, the elitist binarization rule was selected, as shown in Equation (17). This rule introduces a strong exploitation pressure by giving a chance to preserve bits from the best-found solution in the population (X_{Best}). This element of elitism is intended to balance the highly explorative nature of the SBOA's hunting phases, thereby creating a more focused and effective search process.

$$X_{\text{new}}^k = \begin{cases} X_{\text{Best}}^k & \text{if rand} < T(d_i^k), \\ 0 & \text{else.} \end{cases} \quad (17)$$

In this way, the Binary Secretary Bird Optimization Algorithm (BSBOA) is constructed. The process begins with the initialization of the solutions in the binary domain. In each iteration, the binary solutions are modified using Equations (11)–(14), which represent the movement equations specific to the SBOA. Once the solutions are perturbed, they exit the binary domain, and a binarization process is applied using Equations (16) and (17). This cycle is repeated until the defined number of iterations is completed.

Algorithm 2 presents the binary version of the SBOA, where the key section is the binarization that occurs in line 23.

Algorithm 2 Binary Secretary Bird Optimization Algorithm.

Input: Problem Setting $(Dim, ub, lb, Pop\_size(N))$, $Max\_Iter(T)$, $Curr\_Iter(t)$
Output: Best solution

1: Initialize the population randomly.
2: **for** $t = 1 \rightarrow T$ **do**
3: Update Secretary Bird x_{best}.
4: **for** $i = 1 \rightarrow N$ **do**
5: **Exploration Phase:**
6: **if** $t \leq \frac{1}{3}T$ **then**
7: Calculate new status using Searching Stage Equation (11).
8: Update the i-th Secretary Bird using Update Equation (15).
9: **else if** $\frac{1}{3}T < t < \frac{2}{3}T$ **then**
10: Calculate new status using Consuming Stage Equation (12).
11: Update the i-th Secretary Bird using Update Equation (15).
12: **else if** $\frac{1}{3}T < t < \frac{2}{3}T$ **then**
13: Calculate new status using Attacking Stage Equation (13).
14: Update the i-th Secretary Bird using Update Equation (15).
15: **end if**
16: **Exploitation Phase:**
17: **if** $r < 0.5$ **then**
18: Calculate new status using C1 in Exploitation Phase Equation (14).
19: **else**
20: Calculate new status using C_2 in Exploitation Phase Equation (14).
21: **end if**
22: Update the i-th Secretary Bird using Update Equation (15).
23: **Binarization of population X**
24: **end for**
25: Save the best candidate solution so far.
26: **end for**
27: Output the best solution obtained by SBOA for the given optimization problem.

Algorithm Complexity Analysis

Each algorithm requires a certain amount of time to perform its optimization tasks, and these can vary for the same problem. Evaluating algorithmic complexity is an effective way to demonstrate performance in terms of runtime. Big O notation is one of the most widely used tools for complexity analysis [36], and we will use it in this paper to analyze the complexity of BSBOA. Let N be the population size, Dim the number of decision variables, and T the maximum number of iterations. Thus, for the initialization process of random solutions, we have a complexity of $O(N)$. During the optimization process, the complexity is O(T × N) + 2 · O(T × N × Dim), which includes the search for the best positions per iteration (O(T × N)), the updating of the positions of all solutions per iteration (O(T × N × Dim)), and the binarization of all solutions per iteration (O(T × N × Dim)). Thus, the algorithmic complexity of our proposal is O(N × (T × Dim + 1)).

6. Experimental Result

To develop the validation of our proposal, we have used the instances offered in OR-library [37] for both the pesos version (SCP) [35] and for the Unicot version (USCP) [17]. Our proposal was compared to two continuous metaheuristics of great relevance, such as Binary Particle Swarm Optimization [38] and Binary Grey Wolf Optimizer [39].

6.1. Parameter Setting

Before performing experimentation, we perform internal tests for parameter configuration. Specifically, we have carried out experimentation with the population size and the number of iterations. For the population size, we tested from [10, 100] in increments of 10, and the population sizes we tested were [20.50, 70, 100, 150, 200, 300, 400, 500, 600, 700, 800, 900, 1000].

Table 3 shows the subset of instances used for parameter configuration. We have used these instances because we consider that they are representative of the entire portfolio of existing instances of OR-library. The table contains the following details: The first column shows the name of the instance, the second column shows the sample to the type of problem that the instance (SCP or USCP) belongs, the third column refers to the amount of restrictions that the instance possesses, the fourth column refers to the amount of decision variables to optimize, the fifth column refers to the density of some that the matrix has in view of Section 3, and the last column refers to the optimal value of the instance. This experiment was conducted in a team using a Windows 10 operating system, an Intel Core i9-10900 K 3.70 GHz Processor, and 64 GB of RAM.

Table 3. Instances used for parameter configuration.

Instance	Type Problem	M	N	Density (%)	Optimum
41	SCP	200	1000	2.00	429
61	SCP	200	2000	5.00	138
b1	SCP	300	3000	5.00	69
d1	SCP	400	4000	5.00	60
clr10	USCP	511	210	12.30	25
clr11	USCP	1023	330	12.40	23
cyc06	USCP	240	192	2.10	60
cyc07	USCP	672	448	0.90	144

Table 4 shows the best configurations for each instance executed, considering computing time and the fitness reached. Thus, it is observed that the ideal population size for this experiment is 10. For the number of iterations, the value changes for each instance;

therefore, we have determined that the average iterations will be used in experimentation. Thus, the number of iterations is 600.

Table 4. Parameter selection.

Pop	Iter	Instance	Best Fitness	Worst Fitness	Average	Time Seconds	Time Minutes
10	500	41	433	433	433	104.18	1.74
10	70	61	141	141	141	10.32	0.17
10	400	b1	69	69	69	233.97	3.9
10	600	clr10	25	27	25.43	51.69	0.86
10	900	clr11	23	26	24.57	185.9	3.1
10	600	cyc06	60	60	60	52.07	0.87
10	500	cyc07	144	154	149.71	240.98	4.02
10	1000	d1	60	61	60.43	1297.6	21.63

Thus, Table 5 shows the final configuration of the experimentation carried out in this work. The global configuration used by all the metaheuristics (All MH) and the parameters of each metaheuristic are highlighted.

Table 5. Configuration of parameters.

Parameter	Value	
All MH	Population size	10
	Iterations	600
	Independent runs	31
	Transfer functions	V3
	Method of discretization	Elitist
PSO	w_{min}	0.1
	w_{max}	0.9
	c_1	2
	c_2	2
GWO	a	linearly decreases from 2 to 0
SBOA	CF	potentially decreases at 0

6.2. SCP and USCP Instances Resolved

Tables 6 and 7 show the instances used to solve the SCP and USCP, respectively. Each table shows the instance name (column Instance), the number of constraints (column M), the number of decision variables (column N), the density of ones in the matrix mentioned in Section 2 (column Density %), and the instance's optimum. It should be noted that the underlined and bold optima are not global optima but the best results reported in the literature. Thus, for the present work, we solved 22 instances for the SCP and 17 instances for the USCP.

Table 6. Instances used for the SCP.

Instance	M	N	Density (%)	Optimum
41	200	1000	2.00	429
42	200	1000	2.00	512
51	200	2000	2.00	253
52	200	2000	2.00	302
61	200	1000	5.00	138
62	200	1000	5.00	146
A1	300	3000	2.00	253
A2	300	3000	2.00	252

Table 6. *Cont.*

Instance	M	N	Density (%)	Optimum
B1	300	3000	5.00	69
B2	300	3000	5.00	76
C1	400	4000	2.00	227
C2	400	4000	2.00	219
D1	400	4000	5.00	60
D2	400	4000	5.00	66
NRE1	500	5000	10.00	29
NRE2	500	5000	10.00	30
NRF1	500	5000	20.00	14
NRF2	500	5000	20.00	15
NRG1	1000	10,000	2.00	**176**
NRG2	1000	10,000	2.00	**154**
NRH1	1000	10,000	5.00	**63**
NRH2	1000	10,000	5.00	**63**

Table 7. Instances used for the USCP.

Instance	M	N	Density (%)	Optimum
U41	200	1000	2.00	38
U51	200	2000	2.00	34
U61	200	1000	5.00	21
UA1	300	3000	2.00	39
UB1	300	3000	5.00	22
UC1	400	4000	2.00	43
UD1	400	4000	5.00	24
UNRE1	500	5000	10.00	17
UNRF1	500	5000	20.00	10
UNRG1	1000	10,000	2.00	**61**
UNRH1	1000	10,000	5.00	**34**
CLR10	511	210	12.30	**25**
CLR11	1023	330	12.40	**23**
CLR12	2047	495	12.50	**23**
CYC06	240	192	2.10	**60**
CYC07	672	448	0.90	**144**
CYC08	1792	1024	0.40	**344**

6.3. Results of SCP

This subsection shows the results of running the SBOA with the previously established parameters and compared with the GWO and PSO algorithms. In Tables 8 and 9, the results are shown when running the different instances of OR-library. For each algorithm and instance, the best known optimal value, the best value achieved, the worst value achieved, the average, and the standard deviation of the solutions obtained of the 30 executions are shown in the tables [40].

Table 8. Fitness results per SCP instance.

MH	Instance	Opt.	Best	Worst	Avg. Fitness	Std. Fitness
SBOA			433.0	433.0	433.0	0.0
GWO	41	429	433.0	438.0	433.419	1.29
PSO			433.0	437.0	433.258	0.983
SBOA			525.0	527.0	525.968	0.999
GWO	42	512	525.0	527.0	525.935	0.982
PSO			525.0	527.0	526.29	0.957

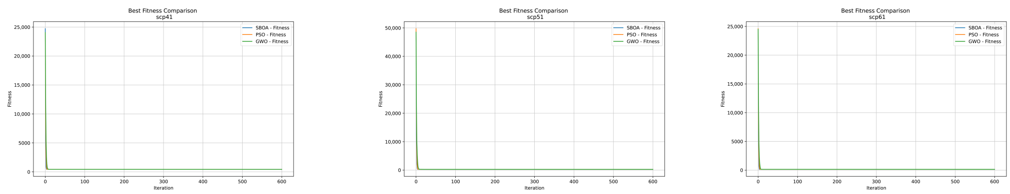
Table 8. *Cont.*

MH	Instance	Opt.	Best	Worst	Avg. Fitness	Std. Fitness
SBOA GWO PSO	51	253	267.0 267.0 267.0	269.0 268.0 269.0	267.29 267.194 267.323	0.52 0.395 0.59
SBOA GWO PSO	52	302	315.0 315.0 315.0	322.0 323.0 323.0	318.387 319.677 318.742	1.979 2.74 2.17
SBOA GWO PSO	61	138	141.0 141.0 141.0	145.0 145.0 145.0	141.935 142.968 141.839	1.605 1.926 1.568
SBOA GWO PSO	62	146	148.0 148.0 148.0	154.0 154.0 154.0	149.677 150.323 149.387	2.277 2.085 2.058
SBOA GWO PSO	A1	253	257.0 257.0 257.0	257.0 257.0 257.0	257.0 257.0 257.0	0.0 0.0 0.0
SBOA GWO PSO	A2	252	258.0 256.0 258.0	265.0 265.0 265.0	260.387 260.452 261.355	1.58 1.997 1.976
SBOA GWO PSO	B1	69	69.0 69.0 69.0	71.0 71.0 71.0	69.387 69.645 69.387	0.79 0.9 0.79
SBOA GWO PSO	B2	76	76.0 76.0 76.0	77.0 82.0 77.0	76.129 76.419 76.161	0.335 1.185 0.368
SBOA GWO PSO	C1	227	231.0 231.0 231.0	234.0 235.0 234.0	233.065 232.806 232.968	0.504 0.858 0.647
SBOA GWO PSO	C2	219	221.0 221.0 221.0	226.0 228.0 229.0	222.387 222.516 222.645	1.559 1.949 2.088
SBOA GWO PSO	D1	60	60.0 60.0 60.0	65.0 63.0 63.0	61.387 61.161 61.323	1.006 0.919 1.028
SBOA GWO PSO	D2	66	67.0 67.0 67.0	69.0 69.0 69.0	67.71 67.774 67.677	0.727 0.791 0.69
SBOA GWO PSO	NRE1	29	29.0 29.0 29.0	29.0 30.0 29.0	29.0 29.032 29.0	0.0 0.177 0.0
SBOA GWO PSO	NRE2	30	30.0 30.0 30.0	32.0 33.0 32.0	31.0 31.065 30.71	0.803 0.878 0.632
SBOA GWO PSO	NRF1	14	14.0 14.0 14.0	14.0 15.0 14.0	14.0 14.032 14.0	0.0 0.177 0.0
SBOA GWO PSO	NRF2	15	15.0 15.0 15.0	15.0 16.0 15.0	15.0 15.032 15.0	0.0 0.177 0.0

A convergence graph represents how the metaheuristic finds progressively better and better solutions as iterations increase. In order to find good solutions in a reasonable time to provide answers to real-world problems, it is expected that the number of iterations will not be excessive in order to avoid the excessive use of computational resources. As described in Crawford et al. [41] and Lemus-Romani et al. [42], graphs were used to document the optimization process while considering the fitness achieved as the iterations progressed. This is shown in Figures 2–5, where the relationship between the number of iterations and the fitness achieved is shown, corresponding to the x and y axes, respectively, where it can be seen that there is good convergence without the algorithm being trapped in a local optimum [43].

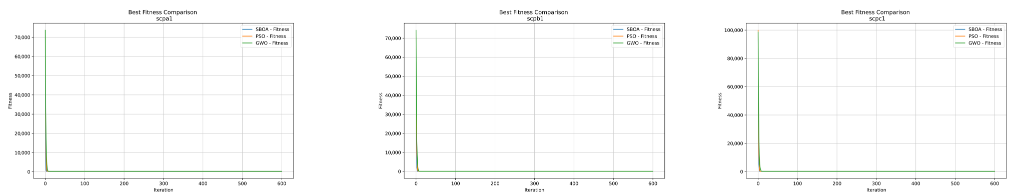
Table 9. Fitness results per SCP instance.

MH	Instance	Opt.	Best	Worst	Avg. Fitness	Std. Fitness
SBOA			178.0	184.0	180.968	1.492
GWO	NRG1	176	179.0	185.0	181.484	1.72
PSO			179.0	184.0	181.0	1.666
SBOA			158.0	161.0	159.355	0.598
GWO	NRG2	154	158.0	162.0	159.677	0.929
PSO			158.0	162.0	159.516	1.043
SBOA			64.0	67.0	65.0	1.016
GWO	NRH1	63	64.0	67.0	65.581	0.943
PSO			64.0	67.0	65.484	0.875
SBOA			64.0	67.0	65.065	1.014
GWO	NRH2	63	64.0	67.0	65.161	0.954
PSO			64.0	66.0	65.194	0.895



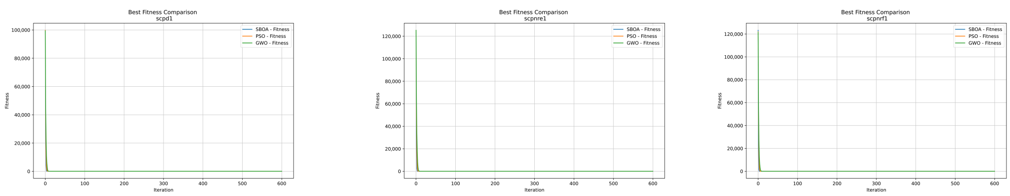
(**a**) Scp 41 (**b**) Scp 51 (**c**) Scp 61

Figure 2. Convergence analysis of the instances Scp 41, Scp 51 and Scp 61.



(**a**) Scp a1 (**b**) Scp b1 (**c**) Scp c1

Figure 3. Convergence analysis of the instances Scp a1, Scp b1 and Scp c1.



(**a**) Scp d1 (**b**) Scp nre1 (**c**) Scp nrf1

Figure 4. Convergence analysis of the instances Scp d1, Scp nre1 and Scp nrf1.

(**a**) Scp nrg1 (**b**) SCP nrh1

Figure 5. Convergence analysis of the instances Scp nrg1 and Scp nrh1.

On the other hand, in Tables 10 and 11, the execution of the different instances for the three Algorithms, the SBOA, GWO and PSO, are shown, considering their execution time.

Table 10. Time per SCP instance.

MH	Instance	Min. Time (s)	Max. Time (s)	Avg. Time (s)	Std. Time (s)
SBOA GWO PSO	41	157.704 153.028 154.711	213.129 187.257 170.525	183.116 173.556 165.33	16.07 9.494 4.352
SBOA GWO PSO	42	142.817 158.63 158.201	218.333 182.134 167.463	174.875 171.814 164.051	19.872 6.122 2.481
SBOA GWO PSO	51	253.613 236.581 237.17	374.187 282.194 271.079	313.787 267.529 259.171	30.033 14.559 8.931
SBOA GWO PSO	52	253.449 233.655 231.334	379.583 283.615 271.309	314.932 266.88 258.577	34.542 13.466 10.615
SBOA GWO PSO	61	98.435 110.113 102.526	157.754 123.04 116.973	122.681 116.26 109.84	13.947 3.258 2.943
SBOA GWO PSO	62	97.503 107.432 104.62	142.421 121.659 115.671	119.435 114.606 110.325	13.442 3.808 3.257
SBOA GWO PSO	A1	776.617 548.174 547.932	992.96 603.27 583.488	886.21 583.829 564.821	52.953 19.38 7.163
SBOA GWO PSO	A2	760.15 559.968 540.209	978.413 613.854 587.692	885.575 589.377 569.523	58.469 14.932 11.293
SBOA GWO PSO	B1	492.081 340.665 331.606	598.676 377.974 369.327	549.83 355.276 355.788	29.989 7.95 7.879
SBOA GWO PSO	B2	509.619 345.028 339.963	609.138 399.848 369.457	566.107 368.475 350.85	27.889 9.286 6.382
SBOA GWO PSO	C1	1624.884 1055.171 1057.741	2006.59 1174.378 1215.157	1800.822 1109.42 1083.767	104.886 26.687 32.262
SBOA GWO PSO	C2	1494.119 1019.127 1034.299	1967.28 1169.176 1112.982	1785.954 1117.342 1075.741	136.993 30.362 15.09

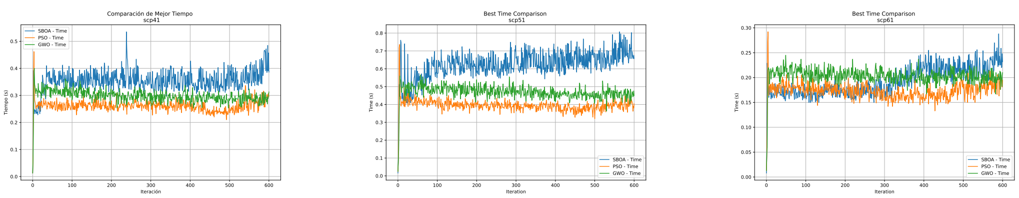
Table 10. *Cont.*

MH	Instance	Min. Time (s)	Max. Time (s)	Avg. Time (s)	Std. Time (s)
SBOA		932.527	1160.315	1073.955	65.504
GWO	D1	640.738	702.894	661.201	13.484
PSO		605.31	667.894	638.656	13.116
SBOA		943.902	1147.318	1060.202	52.619
GWO	D2	608.824	691.086	646.158	16.23
PSO		594.241	664.729	628.357	20.317

Table 11. Time per SCP instance.

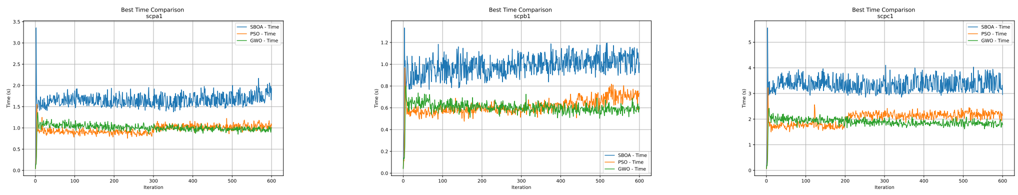
MH	Instance	Min. Time (s)	Max. Time (s)	Avg. Time (s)	Std. Time (s)
SBOA		1163.322	1343.931	1301.288	43.887
GWO	NRE1	709.169	820.318	756.539	27.69
PSO		681.815	749.788	714.205	14.854
SBOA		1111.717	1419.179	1314.766	74.668
GWO	NRE2	683.767	845.554	754.179	40.734
PSO		669.604	855.131	758.741	49.429
SBOA		1032.868	1327.712	1225.842	70.654
GWO	NRF1	565.83	681.723	619.619	26.624
PSO		546.366	661.886	604.704	28.136
SBOA		1099.421	1391.446	1272.068	76.519
GWO	NRF2	573.592	701.44	633.946	30.842
PSO		588.031	713.309	635.212	26.661
SBOA		17,984.637	54,355.97	26,594.345	12,842.352
GWO	NRG1	30,569.108	62,689.422	53,273.186	11,121.198
PSO		32,644.973	61,491.48	55,582.678	6414.388
SBOA		17,185.131	52,717.745	27,342.828	9971.869
GWO	NRG2	25,719.309	61,376.891	53,300.762	10,461.981
PSO		31,367.547	59,373.087	54,333.041	5643.738
SBOA		10,512.106	32,739.219	16,587.815	6416.691
GWO	NRH1	15,165.461	34,183.717	30,025.251	4855.541
PSO		21,114.651	35,256.654	31,207.508	2997.95
SBOA		10,276.136	29,402.972	15,774.859	5539.228
GWO	NRH2	14,111.582	34,673.103	29,343.299	5608.793
PSO		19,751.687	34,721.174	30,023.669	3445.378

Figures 6–9 show graphs with the performance of the three metaheuristics (SBOA, PSO and GWO), where it can be seen that although the times of the SBOA in relation to the iterations were not better than PSO and GWO, they were promising.



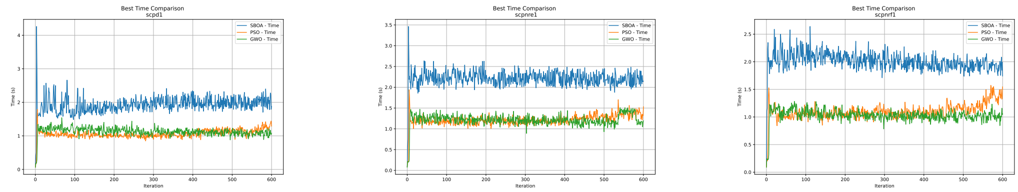
(**a**) Scp 41 (**b**) Scp 51 (**c**) Scp 61

Figure 6. Time analysis of the instances Scp 41, Scp 51 and Scp 61.



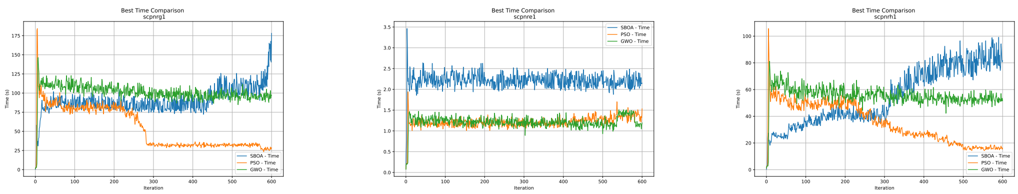
(**a**) Scp a1 (**b**) Scp b1 (**c**) Scp c1

Figure 7. Time analysis of the instances Scp a1, Scp b1 and Scp c1.



(**a**) Scp d1 (**b**) Scp nre1 (**c**) Scp nrf1

Figure 8. Time analysis of the instances Scp d1, Scp nre1 and Scp nrf1.



(**a**) Scp nrg1 (**b**) Scp nre1 (**c**) Scp nrh1

Figure 9. Time analysis of the instances Scp nrg1, Scp nre1 and Scp nrh1.

6.4. Results of USCP

Table 12 shows the results obtained when running the different instances of the USCP for the three metaheuristics investigated in this work (SBOA, GWO and PSO).

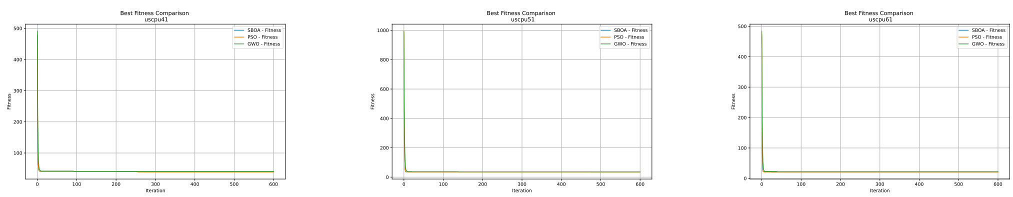
Table 12. Fitness results per USCP instance.

MH	Instance	Opt.	Best	Worst	Avg. Fitness	Std. Fitness
SBOA			39.0	41.0	39.839	0.627
GWO	U41	38	39.0	43.0	40.871	1.07
PSO			39.0	41.0	39.645	0.598
SBOA			35.0	37.0	35.581	0.555
GWO	U51	34	35.0	38.0	36.29	0.681
PSO			35.0	36.0	35.613	0.487
SBOA			21.0	22.0	21.29	0.454
GWO	U61	21	21.0	23.0	21.839	0.514
PSO			21.0	22.0	21.29	0.454
SBOA			39.0	41.0	40.323	0.642
GWO	UA1	39	40.0	43.0	41.387	0.656
PSO			40.0	41.0	40.452	0.498
SBOA			22.0	24.0	22.806	0.591
GWO	UB1	22	22.0	24.0	23.387	0.605
PSO			22.0	23.0	22.774	0.418
SBOA			43.0	45.0	44.323	0.532
GWO	UC1	43	44.0	48.0	46.387	0.79
PSO			44.0	46.0	44.774	0.551

Table 12. *Cont.*

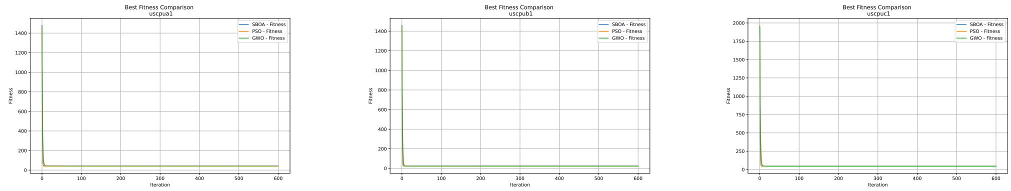
MH	Instance	Opt.	Best	Worst	Avg. Fitness	Std. Fitness
SBOA GWO PSO	UD1	24	25.0 25.0 25.0	26.0 27.0 26.0	25.226 26.194 25.065	0.418 0.47 0.246
SBOA GWO PSO	UNRE1	17	17.0 17.0 17.0	18.0 18.0 17.0	17.516 17.774 17.0	0.5 0.418 0.0
SBOA GWO PSO	UNRF1	10	10.0 10.0 10.0	11.0 11.0 11.0	10.903 10.774 10.29	0.296 0.418 0.454
SBOA GWO PSO	UNRG1	61	62.0 63.0 62.0	63.0 68.0 64.0	62.484 66.194 63.29	0.5 0.895 0.579
SBOA GWO PSO	UNRH1	34	34.0 35.0 34.0	35.0 37.0 35.0	34.323 35.839 34.935	0.467 0.514 0.246
SBOA GWO PSO	CLR10	24	25.0 25.0 25.0	27.0 27.0 27.0	25.097 25.387 25.29	0.39 0.748 0.632
SBOA GWO PSO	CLR11	23	23.0 23.0 23.0	23.0 27.0 27.0	23.0 24.839 23.387	0.0 1.919 1.183
SBOA GWO PSO	CLR12	23	23.0 23.0 23.0	28.0 29.0 26.0	24.258 24.774 23.968	1.586 2.121 1.257
SBOA GWO PSO	CYC06	60	60.0 60.0 60.0	62.0 63.0 62.0	61.097 61.452 61.226	0.817 0.978 0.831
SBOA GWO PSO	CYC07	144	147.0 147.0 144.0	153.0 154.0 153.0	150.871 151.903 150.484	1.362 1.399 1.563
SBOA GWO PSO	CYC08	344	357.0 358.0 358.0	365.0 372.0 367.0	361.129 363.419 362.258	2.324 2.721 1.883

Figures 10–15 show the convergence plots of the unicost instances, where it can be observed that, in general, the SBOA achieves good performance with few iterations without being trapped in a local optimum.



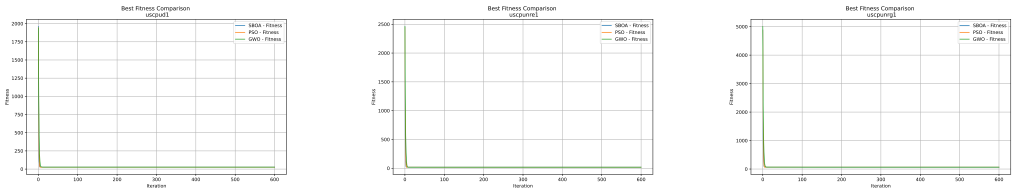
(**a**) UScp 41 (**b**) UScp 51 (**c**) UScp 61

Figure 10. Convergence analysis of the instances UScp 41, UScp 51 and UScp 61.



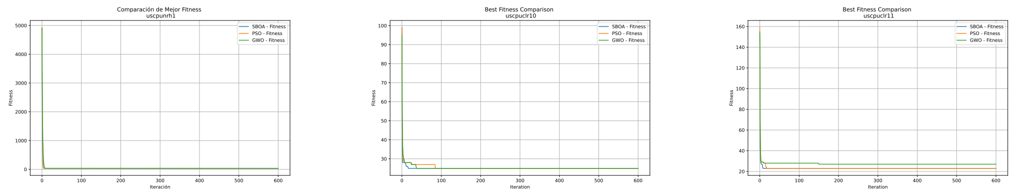
(**a**) UScp a1 (**b**) UScp b1 (**c**) UScp c1

Figure 11. Convergence analysis of the instances UScp a1, UScp b1 and UScp c1.



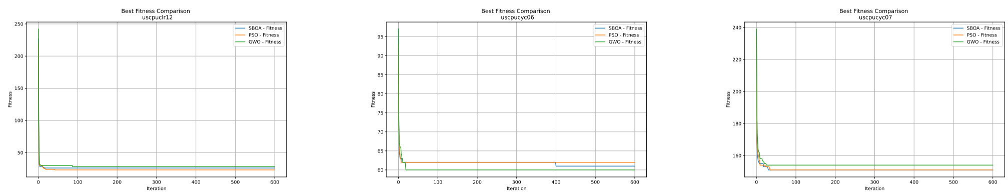
(**a**) UScp d1 (**b**) UScp unre1 (**c**) UScp unrg1

Figure 12. Convergence analysis of the instances UScp d1, UScp unre1 and UScp unrg1.



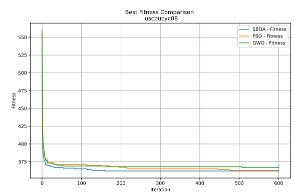
(**a**) UScp unrh1 (**b**) UScp clr10 (**c**) UScp clr11

Figure 13. Convergence analysis of the instances UScp unrh1, UScp clr10 and UScp clr11.



(**a**) UScp clr12 (**b**) UScp cyc06 (**c**) UScp cyc07

Figure 14. Convergence analysis of the instances UScp clr12, UScp cyc06 and UScp cyc07.



UScp cyc08

Figure 15. Convergence analysis of the instance UScp cyc08.

Tables 13 and 14 show the times used for the different USCP instances for the three algorithms studied.

Table 13. Time per USCP instance.

MH	Instance	Min. Time (s)	Max. Time (s)	Avg. Time (s)	Std. Time (s)
SBOA	U41	127.133	165.55	144.041	9.186
GWO		122.253	138.701	128.183	3.852
PSO		117.762	144.135	128.764	5.967
SBOA	U51	240.841	298.561	261.01	14.843
GWO		212.336	229.391	219.389	4.053
PSO		219.169	248.563	233.324	8.576
SBOA	U61	87.424	119.14	98.912	7.76
GWO		68.666	86.004	77.182	4.104
PSO		62.444	78.625	73.111	3.677
SBOA	UA1	596.009	988.713	718.132	81.884
GWO		271.85	342.57	299.673	21.205
PSO		155.901	331.492	281.74	35.986
SBOA	UB1	261.626	572.229	354.131	96.36
GWO		163.543	178.737	169.919	3.133
PSO		163.947	189.483	177.069	6.403
SBOA	UC1	43.0	45.0	44.323	0.532
GWO		44.0	48.0	46.387	0.79
PSO		44.0	46.0	44.774	0.551
SBOA	UD1	466.161	1065.89	691.041	212.326
GWO		280.281	314.57	292.719	6.333
PSO		280.569	329.153	302.325	12.853
SBOA	UNRE1	863.663	1732.248	1337.399	330.758
GWO		410.564	705.947	434.391	50.715
PSO		423.553	497.961	460.517	18.005
SBOA	UNRF1	1070.355	1650.344	1551.32	130.705
GWO		424.606	683.387	589.47	95.124
PSO		449.775	535.65	485.254	25.765
SBOA	UNRG1	5414.287	14,309.31	7441.461	3000.263
GWO		8737.818	18,034.46	15,638.192	3071.343
PSO		13,479.719	21,729.841	18,346.111	2374.816
SBOA	UNRH1	4168.711	16,435.724	7474.938	3805.651
GWO		4775.536	11,231.733	9150.029	1921.657
PSO		6517.542	14,954.869	11,776.28	1780.385

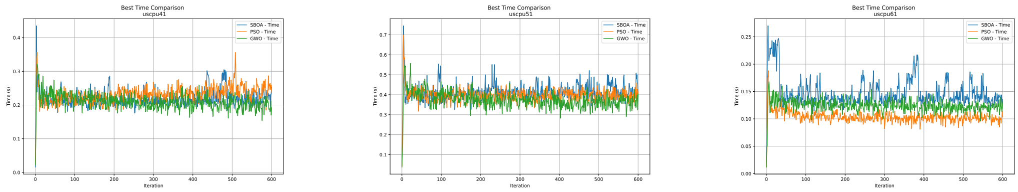
Table 14. Time per USCP instance.

Metaheuristic	Instance	Min. Time (s)	Max. Time (s)	Avg. Time (s)	Std. Time (s)
SBOA	CLR10	58.212	86.825	75.066	6.622
GWO		23.831	29.506	26.374	1.434
PSO		29.711	36.123	32.689	1.824
SBOA	CLR11	126.482	196.981	141.176	11.534
GWO		53.558	69.384	61.066	4.716
PSO		66.438	85.256	71.301	4.728
SBOA	CLR12	604.751	1064.573	801.739	149.713
GWO		338.341	500.013	407.826	43.52
PSO		501.697	604.357	550.129	22.986
SBOA	CYC06	53.307	93.846	68.63	8.254
GWO		22.638	26.964	24.317	1.085
PSO		23.312	28.313	25.79	1.229
SBOA	CYC07	282.102	319.146	306.65	6.968
GWO		138.408	163.015	148.045	6.521
PSO		146.28	175.9	158.426	5.986

Table 14. *Cont.*

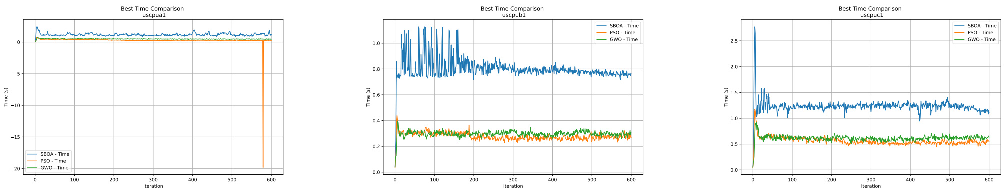
Metaheuristic	Instance	Min. Time (s)	Max. Time (s)	Avg. Time (s)	Std. Time (s)
SBOA			5163.509	4655.015	226.937
GWO	CYC08	3080.423	3591.295	3391.635	126.319
PSO		3562.642	3902.347	3747.505	84.342

Figures 16–21 show graphs with the performance of the three metaheuristics (SBOA, PSO and GWO), enabling us to compare it with the times achieved by the SBOA.



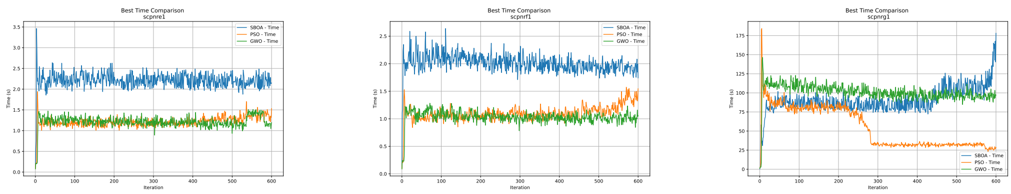
(**a**) UScp 41 (**b**) UScp 51 (**c**) UScp 61

Figure 16. Time analysis of the instances UScp 41, UScp 51 and UScp 61.



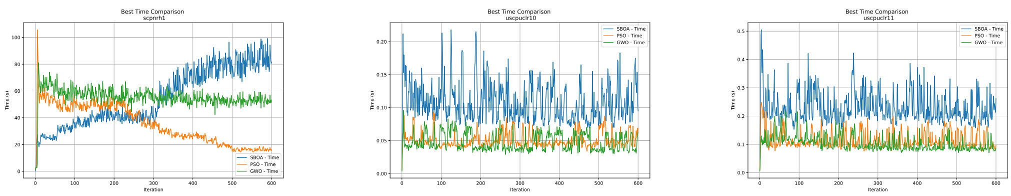
(**a**) UScp a1 (**b**) UScp b1 (**c**) UScp c1

Figure 17. Time analysis of the instances UScp a1, UScp b1 and UScp c1.



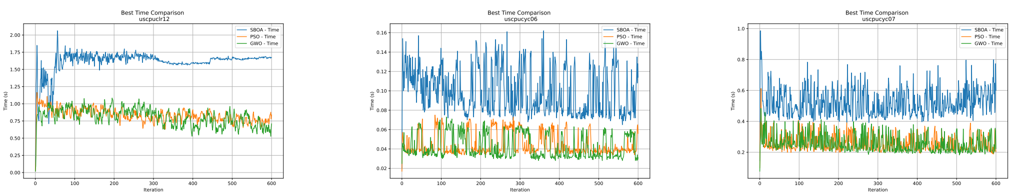
(**a**) UScp nre1 (**b**) UScp nrf1 (**c**) UScp nrg1

Figure 18. Time analysis of the instances UScp nre1, UScp nrf1 and UScp nrg1.



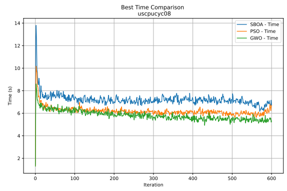
(**a**) UScp nrh1 (**b**) UScp clr10 (**c**) UScp clr11

Figure 19. Time analysis of the instances UScp nrh1, UScp clr10 and UScp clr11.



(**a**) UScp clr12 (**b**) UScp cyc06 (**c**) UScp cyc07

Figure 20. Time analysis of the instances UScp clr12, UScp cyc06 and UScp cyc07.



UScp cyc08

Figure 21. Time analysis of the instance cyc08.

Statistical Tests

To validate our work, Tables 15–18 show the statistical significance tests for the algorithms worked on: SBP, PSO and GWO. This table shows the *p*-values that act as indicators of statistical significance. We use the nonparametric Wilcoxon–Mann–Whitney test to perform this validation [44] since we have two independent samples and we cannot assume normality for at least one of them. The hypotheses considered are the following:

$$H_0 : \mu_A \geq \mu_B,$$

$$H_1 : \mu_A < \mu_B$$

where μ_A and μ_B represent the average value delivered by algorithms A and B. We consider that if the *p*-value is less than 0.05, the hypothesis H_0 will be rejected, with hypothesis H_1 being accepted and these cases are highlighted in bold and underlined in the Tables.

Table 15. Average *p*-value of the SBOA compared with PSO and GWO for SCP41, ScP42, SCP51, SCP52, SCP61 and SCP62.

	4.1			**4.2**			**5.1**			**5.2**			**6.1**			**6.2**		
	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO
SBO	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	**0.03**	-	≥0.05	≥0.05
PSO	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	**0.02**	≥0.05	-	≥0.05
GWO	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	**0.03**	**0.02**	-	≥0.05	≥0.05	-

Table 16. Average *p*-value of the SBOA compared with PSO and GWO for a1, a2, b1, b2, c1, c2, d1 and d2.

	a.1			**a.2**			**b.1**			**b.2**		
	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO
SBO	-	≥0.05	≥0.05	-	0.05	-	-	≥0.05	≥0.05	-	≥0.05	≥0.05
PSO	≥0.05	-	≥0.05	≥0.05	-	≥0.05	1.0	-	≥0.05	≥0.05	-	≥0.05
GWO	≥0.05	≥0.05	-	-	≥0.05	-	0.2	≥0.05	-	≥0.05	≥0.05	-
	c.1			c.2			d.1			d.2		
	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO
SBO	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05
PSO	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05
GWO	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-

The BSBOA demonstrated remarkable ability in solving the Set Covering Problem (SCP) and its unicost variant (USCP) using the V3 transfer function and elitist rule. For the SCP, it stood out by generating high-quality solutions with lower costs compared to metaheuristics, like GWO and PSO; in the USCP, where costs are homogeneous, it maintained competitive performance with less pronounced differences. Additionally, the BSBOA achieved favorable execution times, establishing itself as an efficient and versatile option for combinatorial optimization problems.

Table 17. Average *p*-value of the SBOA compared with PSO and GWO for nre1, nre2, nrf1, nrf2, nrg1, nrg2, nrh1 and nrh2.

		nre1			**nre2**			**nrf1**			**nrf2**	
	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO
SBO	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05
PSO	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05
GWO	≥0.05	≥0.05	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	0.33	≥0.05	-
		nrg1			nrg2			nrh1			nrh2	
	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO
SBO	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-	**0.003**	**0.02**	-	≥0.05	≥0.05
PSO	≥0.05	-	≥0.05	≥0.05	-	≥0.05	**0.03**	-	≥0.05	≥0.05	-	≥0.05
GWO	≥0.05	≥0.05	-	≥0.05	≥0.05	-	**0.02**	≥0.05	-	≥0.05	≥0.05	-

Table 18. Average *p*-value of the SBOA compared with PSO and GWO for u41, u51, u61, ua1, ub1, uc1, uclr10, uclr12, ucyc06, ucyc07, ud1, unre1, unrf1, unre1 and unrh1.

		u41			**u51**			**u61**			**ua1**	
	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO
SBO	-	≥0.05	**0.0**	-	≥0.05	0.0	-	≥0.05	0.0	-	≥0.05	0.0
PSO	≥0.05	-	**0.0**	≥0.05	-	**0.0**	≥0.05	-	**0.0**	≥0.05	-	**0.0**
GWO	**0.0**	**0.0**	-	**0.0**	**0.0**	-	**0.0**	**0.0**	-	0.0	**0.0**	-
		ub1			uc1			uclr10			uclr12	
	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO
SBO	-	≥0.05	**0.0**	-	**0.0**	**0.0**	-	≥0.05	≥0.05	-	0.5	≥0.05
PSO	≥0.05	-	**0.0**	**0.0**	-	**0.0**	≥0.05	-	≥0.05	≥0.05	-	≥0.05
GWO	**0.0**	**0.0**	-	**0.0**	**0.0**	-	≥0.05	≥0.05	-	≥0.05	≥0.05	-
		ucyc06			ucyc07			ucyc07			ud1	
	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO
SBO	-	≥0.05	0.12	-	≥0.05	**0.0**	-	≥0.05	**0.0**	-	≥0.05	**0.0**
PSO	≥0.05	-	≥0.05	≥0.05	-	**0.0**	≥0.05	-	**0.0**	≥0.05	-	**0.0**
GWO	≥0.05	≥0.05	-	**0.0**	**0.0**	-	**0.0**	**0.0**	-	**0.0**	**0.0**	-
		unre1			unrf1			unre1			unrh1	
	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO	SBO	PSO	GWO
SBO	-	0.0	**0.04**	-	**0.0**	≥0.05	-	**0.0**	**0.0**	-	**0.0**	**0.0**
PSO	**0.0**	-	**0.0**	**0.0**	-	**0.0**	**0.0**	-	**0.0**	0.0	-	**0.0**
GWO	**0.04**	**0.0**	-	≥0.05	**0.0**	-	**0.0**	**0.0**	-	**0.0**	**0.0**	-

7. Discussion

The experimental outcomes consistently show BSBOA's ability to yield high-quality solutions, often surpassing GWO and PSO in solution value and robustness (lower standard deviations) for the SCP, while maintaining competitive performance for the USCP. The computational analysis confirms BSBOA's operational efficiency with a generally low computational burden.

BSBOA's effectiveness stems from SBOA's core design, inspired by the secretary bird's natural behaviors of hunting and evasion, fostering a well-balanced interplay between exploration and exploitation crucial for NP-hard problems. Its systematic adaptation from continuous to binary domains, involving a rigorous exploration of eight transfer functions and five discretization methods, was pivotal. The selection of the V3 transfer function and the Elitist rule, based on prior research for optimal exploration–exploitation balance in medium-sized problems, ensures this effective translation and harmonious interaction with the discrete solution space.

However, it is crucial to acknowledge certain trade-offs and constraints. The iterative nature of the binarization process, applying transfer functions and rules in each cycle, introduces a computational overhead that could be significant for extremely large-scale problems or real-time systems, despite current observed efficiencies. Moreover, while our systematic validation identified an optimal configuration (V3 and Elitist) for the tested instances, the generalizability of this specific setup across all possible SCP and USCP variations, particularly those with distinct structural properties or much higher dimensions, requires further extensive validation. The current reliance on standard benchmark instances, while a common practice, also represents a limitation in fully assessing its performance in the intricate, diverse conditions of real-world applications.

These considerations underscore the ongoing need for refinement and broader validation. Such endeavors are vital to fully realize BSBOA's potential for industrial and engineering challenges.

8. Conclusions

In this work, a solution to the SCP was provided using the SBOA, which is inspired by the natural behavior of the secretary bird. On the one hand, the SBOA was originally designed to work on continuous space optimization problems; on the other hand, the SCP is a binary problem. Several binarization techniques were proposed to adapt the algorithm to the discrete domain. Combinations of eight transfer functions were used, along with five discretization methods. Subsequently, the binary version of the SBOA was compared with the GWO and PSO algorithms, providing a solution to the SCP and its USCP variant. The BSBOA is positioned as a robust and effective tool for resolving both the SCP and USCP, excelling in solution quality and execution efficiency compared to GWO and PSO. Its ability to handle different levels of complexity and balance exploration and exploitation makes it ideal for practical optimization applications. Future work is to extend its applicability to more complex and large-scale problems, as well as to optimize its performance in terms of time and accuracy.

Author Contributions: Conceptualization, B.C., F.C.-C. and R.S.; methodology, B.C., F.C.-C., C.P.T.M.-l., J.L.A. and F.S.-P.; software, F.C.-C., C.P.T.M.-l., J.L.A. and F.S.-P.; validation, B.C., F.C.-C., R.S., G.A. and G.G.; formal analysis, C.P.T.M.-l., J.L.A. and F.S.-P.; investigation, B.C., F.C.-C., R.S., C.P.T.M.-l., J.L.A., F.S.-P., G.A. and G.G.; resources, C.P.T.M.-l., J.L.A., F.S.-P. and G.A.; writing—original draft preparation, C.P.T.M.-l., J.L.A., F.S.-P. and G.A.; writing—review and editing, B.C., F.C.-C., R.S. and G.G.; supervision, B.C., F.C.-C. and R.S.; funding acquisition, B.C. and R.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article; further inquiries can be directed to the corresponding authors.

Acknowledgments: Felipe Cisternas-Caneo was supported by the National Agency for Research and Development ANID BECAS/DOCTORADO NACIONAL 21230203.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SBOA	Secretary Bird Optimization Algorithm
BSBOA	Binary Secretary Bird Optimization Algorithm
PSO	Particle Swarm Optimization
GWO	Grey Wolf Optimizer
ACO	Ant Colony Optimization
GA	Genetic Algorithm
BCSO	Binary Cat Swarm Optimization
BBHA	Binary Black Hole Algorithm
CS	Cuckoo Search
TS	Tabu Search
SCP	Set Covering Problem
USCP	Unicost Set Covering Problem

References

1. Coelho, P.; Silva, C. Parallel Metaheuristics for shop scheduling: Enabling industry 4.0. *Procedia Comput. Sci.* **2021**, *180*, 778–786. [CrossRef]
2. Ghotb, S.; Sowlati, T.; Mortyn, J. Scheduling of log logistics using a metaheuristic approach. *Expert Syst. Appl.* **2024**, *238*, 122008. [CrossRef]
3. Núñez-López, J.M.; Segovia-Hernández, J.G.; Sánchez-Ramírez, E.; Ponce-Ortega, J.M. Integrating metaheuristic methods and deterministic strategies for optimizing supply chain equipment design in process engineering. *Chem. Eng. Res. Des.* **2025**, *214*, 93–104. [CrossRef]
4. Dhouib, S.; Zouari, A. Adaptive iterated stochastic metaheuristic to optimize holes drilling path in manufacturing industry: The Adaptive-Dhouib-Matrix-3 (A-DM3). *Eng. Appl. Artif. Intell.* **2023**, *120*, 105898. [CrossRef]
5. Yang, C.L.; Yilma, A.A.; Sutrisno, H.; Woldegiorgis, B.H.; Nguyen, T.P.Q. LSTMbased framework with metaheuristic optimizer for manufacturing process monitoring. *Alex. Eng. J.* **2023**, *83*, 43–52. [CrossRef]
6. Huang, B.; Tang, L.; Baldacci, R.; Wang, G.; Sun, D. A metaheuristic algorithm for a locomotive routing problem arising in the steel industry. *Eur. J. Oper. Res.* **2023**, *308*, 385–399. [CrossRef]
7. Zuhanda, M.K.; Hasibuan, S.A.R.S.; Napitupulu, Y.Y.; Hartono. An exact and metaheuristic optimization framework for solving Vehicle Routing Problems with Shipment Consolidation using population-based and Swarm Intelligence. *Decis. Anal. J.* **2024**, *13*, 100517. [CrossRef]
8. Lopes, J.; Guimarães, T.; Duarte, J.; Santos, M. Enhancing Surgery Scheduling in Health Care Settings with Metaheuristic Optimization Models: Algorithm Validation Study. *JMIR Med. Inform.* **2025**, *13*, e57231. [CrossRef]
9. Mzili, T.; Mzili, I.; Riffi, M.E.; Kurdi, M.; Ali, A.H.; Pamucar, D.; Abualigah, L. Enhancing COVID-19 vaccination and medication distribution routing strategies in rural regions of Morocco: A comparative metaheuristics analysis. *Inform. Med. Unlocked* **2024**, *46*, 101467. [CrossRef]
10. Mostafaei, K.; Yousefi, M.; Kreuzer, O.; Kianpour, M.N. Simulation-based mineral prospectivity modeling and Gray Wolf optimization algorithm for delimiting exploration targets. *Ore Geol. Rev.* **2025**, *177*, 106458. [CrossRef]
11. Canales Bustos, L.; Santibañez González, E.; Candia Véjar, A. A multi objective optimization model for the design of an effective decarbonized supply chain in mining. *Int. J. Prod. Econ.* **2017**, *193*, 449–464. [CrossRef]
12. Moradi, N.; Mafakheri, F.; Wang, C. Set covering routing problems: A review and classification scheme. *Comput. Ind. Eng.* **2024**, *198*, 110730. [CrossRef]
13. Karp, R.M. On the computational complexity of combinatorial problems. *Networks* **1975**, *5*, 45–68. [CrossRef]
14. Šarac, D.; Kopić, M.; Mostarac, K.; Kujačić, M.; Jovanović, B. Application of set covering location problem for organizing the public postal network. *PROMET-Traffic Transp.* **2016**, *28*, 403–413. [CrossRef]
15. Mesquita, M.; Paias, A. Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem. *Comput. Oper. Res.* **2008**, *35*, 1562–1575. [CrossRef]
16. Crawford, B.; Soto, R.; Monfroy, E. Cultural algorithms for the set covering problem. In Proceedings of the Advances in Swarm Intelligence: 4th International Conference, ICSI 2013, Harbin, China, 12–15 June 2013; Proceedings, Part II 4; Springer: Berlin/Heidelberg, Germany, 2013; pp. 27–34.
17. Wang, Y.; Pan, S.; Al-Shihabi, S.; Zhou, J.; Yang, N.; Yin, M. An improved configuration checking-based algorithm for the unicost set covering problem. *Eur. J. Oper. Res.* **2021**, *294*, 476–491. [CrossRef]

18. Wu, C.; Murray, A.T. Optimizing public transit quality and system access: The multiple-route, maximal covering/shortest-path problem. *Environ. Plan. B Plan. Des.* **2005**, *32*, 163–178. [CrossRef]
19. Wang, R.L.; Okazaki, K. An improved genetic algorithm with conditional genetic operators and its application to set-covering problem. *Soft Comput.* **2007**, *11*, 687–694. [CrossRef]
20. Crawford, B.; Soto, R.; Monfroy, E.; Palma, W.; Castro, C.; Paredes, F. Parameter tuning of a choice-function based hyperheuristic using particle swarm optimization. *Expert Syst. Appl.* **2013**, *40*, 1690–1695. [CrossRef]
21. Ren, Z.G.; Feng, Z.R.; Ke, L.J.; Zhang, Z.J. New ideas for applying ant colony optimization to the set covering problem. *Comput. Ind. Eng.* **2010**, *58*, 774–784. [CrossRef]
22. Caserta, M. Tabu search-based metaheuristic algorithm for large-scale set covering problems. In *Metaheuristics: Progress in Complex Systems Optimization*; Springer: Boston, MA, USA, 2007; pp. 43–63.
23. Naji-Azimi, Z.; Toth, P.; Galli, L. An electromagnetism metaheuristic for the unicost set covering problem. *Eur. J. Oper. Res.* **2010**, *205*, 290–300. [CrossRef]
24. Sundar, S.; Singh, A. A hybrid heuristic for the set covering problem. *Oper. Res.* **2012**, *12*, 345–365. [CrossRef]
25. Reyes, V.; Araya, I. A GRASP-based scheme for the set covering problem. *Oper. Res.* **2021**, *21*, 2391–2408. [CrossRef]
26. Fu, Y.; Liu, D.; Chen, J.; He, L. Secretary bird optimization algorithm: A new metaheuristic for solving global optimization problems. *Artif. Intell. Rev.* **2024**, *57*, 1–102. [CrossRef]
27. Cochran, J.K.; Uribe, A.M. A set covering formulation for agile capacity planning within supply chains. *Int. J. Prod. Econ.* **2005**, *95*, 139–149. [CrossRef]
28. Nam, S.; Shen, H.; Ryu, C.; Shin, J.G. SCP-Matrix based shipyard APS design: Application to long-term production plan. *Int. J. Nav. Archit. Ocean Eng.* **2018**, *10*, 741–761. [CrossRef]
29. Mol, G.; Ermiş, M. Solving the Large-Scale Crew Pairing Problem in the Airline Industry Using the Column Generation Method. In Proceedings of the International Symposium for Production Research, Budva, Montenegro, 9–11 October 2024; Springer: Cham, Switzerland, 2024; pp. 333–348.
30. Borne, P.; Tangour, F. Metaheuristics for the Optimization in Planning and Scheduling. *IFAC Proc. Vol.* **2007**, *40*, 1–7. [CrossRef]
31. Niu, B.; Wang, Y.; Liu, J.; Yue, G.X.G. Path planning for unmanned aerial vehicles in complex environment based on an improved continuous ant colony optimisation. *Comput. Electr. Eng.* **2025**, *123*, 110034. [CrossRef]
32. Yang, X.; Li, H.; Huang, Y. An adaptive dynamic multi-swarm particle swarm optimization with stagnation detection and spatial exclusion for solving continuous optimization problems. *Eng. Appl. Artif. Intell.* **2023**, *123*, 106215. [CrossRef]
33. Becerra-Rozas, M.; Lemus-Romani, J.; Cisternas-Caneo, F.; Crawford, B.; Soto, R.; Astorga, G.; Castro, C.; García, J. Continuous metaheuristics for binary optimization problems: An updated systematic literature review. *Mathematics* **2022**, *11*, 129. [CrossRef]
34. Crawford, B.; Soto, R.; Astorga, G.; García, J.; Castro, C.; Paredes, F. Putting continuous metaheuristics to work in binary search spaces. *Complexity* **2017**, *2017*, 8404231. [CrossRef]
35. Lanza-Gutierrez, J.M.; Crawford, B.; Soto, R.; Berrios, N.; Gomez-Pulido, J.A.; Paredes, F. Analyzing the effects of binarization techniques when solving the set covering problem through swarm optimization. *Expert Syst. Appl.* **2017**, *70*, 67–82. [CrossRef]
36. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*; MIT Press: Cambridge, MA, USA, 2022.
37. Beasley, J.E.; Jörnsten, K. Enhancing an algorithm for set covering problems. *Eur. J. Oper. Res.* **1992**, *58*, 293–300. [CrossRef]
38. Kennedy, J.; Eberhart, R.C. A discrete binary version of the particle swarm algorithm. In Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Orlando, FL, USA, 12–15 October 1997; IEEE: Piscataway, NJ, USA, 1997; Volume 5, pp. 4104–4108.
39. Emary, E.; Zawbaa, H.M.; Hassanien, A.E. Binary grey wolf optimization approaches for feature selection. *Neurocomputing* **2016**, *172*, 371–381. [CrossRef]
40. Houssein, E.H.; Saeed, M.K.; Hu, G.; Al-Sayed, M.M. Metaheuristics for solving global and engineering optimization problems: Review, applications, open issues and challenges. *Arch. Comput. Methods Eng.* **2024**, *31*, 4485–4519. [CrossRef]
41. Crawford, B.; Soto, R.; Lemus-Romani, J.; Becerra-Rozas, M.; Lanza-Gutiérrez, J.M.; Caballé, N.; Castillo, M.; Tapia, D.; Cisternas-Caneo, F.; García, J.; et al. Q-learnheuristics: Towards data-driven balanced metaheuristics. *Mathematics* **2021**, *9*, 1839. [CrossRef]
42. Lemus-Romani, J.; Becerra-Rozas, M.; Crawford, B.; Soto, R.; Cisternas-Caneo, F.; Vega, E.; Castillo, M.; Tapia, D.; Astorga, G.; Palma, W.; et al. A novel learning-based binarization scheme selector for swarm algorithms solving combinatorial problems. *Mathematics* **2021**, *9*, 2887. [CrossRef]

43. Singh, P. The Fast Forward Quantum Optimization Algorithm: A study of convergence and novel unconstrained optimization. *Comput. Methods Appl. Mech. Eng.* **2025**, *443*, 118039. [CrossRef]
44. Mann, H.B.; Whitney, D.R. On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Stat.* **1947**, *18*, 50–60. [CrossRef]