

Article

An Experimental Study of Transfer Functions and Binarization Strategies in Binary Arithmetic Optimization Algorithms for the Set Covering Problem

Broderick Crawford ^{1,*} , Ricardo Soto ¹ , Hugo Caballero ¹ , Gino Astorga ² , Felipe Cisternas-Caneo ¹ ,
Fabián Solís-Piñones ¹  and Giovanni Giachetti ³ 

¹ Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso, Avenida Brasil 2241, Valparaíso 2362807, Chile; ricardo.soto@pucv.cl (R.S.); hugo.caballero.c@mail.pucv.cl (H.C.); felipe.cisternas.c@mail.pucv.cl (F.C.-C.); 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

Metaheuristics have proven to be effective in solving large-scale combinatorial problems by combining global exploration with local exploitation, all within a reasonably short time. The balance between these phases is crucial to avoid slow or premature convergence. We propose binary variants of the Arithmetic Optimization Algorithm for the set cover problem, integrating a two-step binarization scheme based on transfer functions with binarization rules and a greedy repair operator to ensure feasibility. We evaluate the proposed solution using forty-five instances from OR-Beasley and compare it with representative approaches, including genetic algorithms, path-relinking strategies, and Lagrangian-based heuristics. The quality of the solution is evaluated using relative percentage deviation and stability with the coefficient of variation. The results show competitive deviations and consistently low variation, confirming that our approach is a robust alternative with a solid balance between exploration and exploitation.

Keywords: arithmetic optimization algorithm; binary metaheuristics; binarization techniques; set covering problem; combinatorial optimization; exploration–exploitation balance

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



Academic Editor: Janez Žerovnik

Received: 2 August 2025

Revised: 22 September 2025

Accepted: 27 September 2025

Published: 30 September 2025

Citation: Crawford, B.; Soto, R.; Caballero, H.; Astorga, G.; Cisternas-Caneo, F.; Solís-Piñones, F.; Giachetti, G. An Experimental Study of Transfer Functions and Binarization Strategies in Binary Arithmetic Optimization Algorithms for the Set Covering Problem. *Mathematics* **2025**, *13*, 3129. <https://doi.org/10.3390/math13193129>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The Set Covering Problem [1,2] is a widely studied NP-hard combinatorial optimization problem of great industrial relevance, including applications in service planning, facility location, and network optimization [3]. Despite its straightforward formulation, its computational complexity has led to the development of numerous exact and approximate methods, including metaheuristics such as genetic algorithms, ant colony optimization, GRASP, and chemical reaction algorithms. However, no metaheuristic consistently outperforms others across all instances, highlighting the need for tailored approaches.

In this context, we selected the Arithmetic Optimization Algorithm [4] due to its simplicity, strong balance between exploration and exploitation, and recent success in continuous optimization tasks. Although originally designed for continuous domains,

its potential for discrete problems such as SCPs remains largely unexplored. This study presents an adaptation of the AOA to the SCP, incorporating a two-step binarization [5] process that preserves the fundamental structure of the original algorithm while integrating well-established transfer functions and binarization rules used in previous SCP research.

The rest of this article is organized as follows: Section 2 presents the Set Covering Problem (SCP) along with its state of the art and formalization; Section 3 describes the Arithmetic Optimization Algorithm (AOA) metaheuristic, including its main parameters, exploration and exploitation components, and original algorithm; Section 4 introduces the two-step binarization mechanism, transfer functions, and binarization rules along with the reasons for their selection; Section 5 details our proposal, specifically, adaptation of the AOA to a binary version that incorporates a mechanism for validating and repairing candidate solutions; Section 6 reports the experimental results as evaluated using Relative Percentage Deviation (RPD) and Coefficient of Variation (CV), presents comparative analyses with reference techniques, and discusses the effect of different combinations of transfer functions and binarization rules and the statistical analysis of the results; Section 7 reports the computational overhead along with our method's limitations; Section 8 presents the novel elements and main achievements of our proposal; finally, Section 9 presents the conclusions along with future work that could be developed based on the results of this study.

2. Set Covering Problem

The Set Covering Problem is a classical NP-complete combinatorial optimization problem with broad applications in areas such as facility location, scheduling, and resource allocation. A notable case is crew scheduling, where the goal is to select a cost-efficient set of crews to cover all required trips [6]. Due to its computational complexity, the problem is commonly addressed using exact methods for small instances and heuristic or metaheuristic approaches for larger ones [7].

In recent years, nature-inspired metaheuristics such as genetic algorithms, ACO, and hybrid methods have shown strong performance on this problem [8,9]. However, most of these approaches rely on well-established binary frameworks, leaving room to explore newer algorithms that have not yet been fully adapted to discrete spaces.

Formally, the SCP can be defined as follows. Let $A = (a_{ij})$ be a binary matrix of size $m \times n$, where $a_{ij} \in \{0, 1\}$, and let $C = (c_1, c_2, \dots, c_n)$ be a non-negative cost vector associated with the n columns. Define $I = \{1, 2, \dots, m\}$ as the set of rows and $J = \{1, 2, \dots, n\}$ as the set of columns. Each $c_j > 0$ for $j \in J$ represents the cost of selecting column j , which covers row i if $a_{ij} = 1$. The objective is to find a subset $S \subseteq J$ such that every row $i \in I$ is covered by at least one column $j \in S$ and the total cost of the selected columns is minimized. This formulation is commonly referred to as the column-based representation.

$$\text{Minimize } Z = \sum_{j=1}^n c_j x_j, \quad (1)$$

subject to

$$\sum_{j=1}^n a_{ij} x_j \geq 1, \quad \forall i \in I, \quad (2)$$

$$x_j \in \{0, 1\}, \quad \forall j \in J. \quad (3)$$

where x_j is a binary decision variable such that $x_j = 1$ if column j is selected in the solution and $x_j = 0$ otherwise. The constraints [10] ensure that each row i is covered by at least one selected column j .

Equations (1)–(3) describe the column-based representation of the SCP, where the solution is encoded as a binary string of length n corresponding to the number of columns.

The objective is to find a minimum-cost subset $S \subseteq J$ such that every row $i \in I$ is covered by at least one column $j \in S$.

3. Arithmetic Optimization Algorithm

The Arithmetic Optimization Algorithm (AOA) is a population-based metaheuristic introduced by Abualigah et al. [4], which was designed to solve complex optimization problems using a mathematical framework based on the fundamental arithmetic operations: addition, subtraction, multiplication, and division. Unlike many metaheuristics inspired by biological or physical phenomena, the AOA relies exclusively on these operators, which are applied probabilistically to update candidate solutions and balance the exploration and exploitation phases [11].

3.1. Core Components of the Arithmetic Optimization Algorithm

3.1.1. Random Population Initialization

$$X_0 = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{bmatrix} \quad (4)$$

The initial population matrix is defined as $X_0 \in \mathbb{R}^{m \times n}$, where m is the number of solutions and n is the number of variables. It is initialized randomly and updated in each iteration according to the exploration and exploitation mechanisms of the Arithmetic Optimization Algorithm.

3.1.2. Math Optimizer Accelerated

The Math Optimizer Accelerated (MOA) is an adaptive parameter that controls the balance between exploration and exploitation during the execution of the AOA algorithm, which is recalculated in each iteration. This parameter is typically set high enough to ensure exploration when starting the process, then decreases to ensure exploitation. The MOA is defined by Equation (5). The parameters *Min* and *Max* are defined in Table 1.

$$MOA(C_Iter) = Min + C_Iter \times \left(\frac{Max - Min}{M_Iter} \right) \quad (5)$$

Table 1. Parameters used in the AOA.

Parameter	Description
<i>LB</i>	Lower Bound. Minimum value that a decision variable can take. Defines the lower limit of the search space.
<i>UB</i>	Upper Bound. Maximum value that a decision variable can take. Defines the upper limit of the search space.
<i>C_Iter</i>	Current iteration number. Indicates the current generation in the optimization process.
<i>M_Iter</i>	Maximum number of iterations. Determines when the algorithm stops.
<i>Min</i>	Predefined constant value, typically 0.2
<i>Max</i>	Predefined constant value, typically 1.0

Table 1. Cont.

Parameter	Description
α	Curvature control parameter used in the computation of <i>MOP</i> . It controls the rate at which <i>MOP</i> decreases. Typical value: $\alpha = 5$
r_1	A random number in $[0, 1]$ to decide whether to perform exploration or exploitation.
r_2	A random number in $[0, 1]$ used in exploration phase to select between division and multiplication.
r_3	A random number in $[0, 1]$ used in exploitation phase to select between subtraction and addition.
μ	Parameter random in $[0, 1]$. It is used as a stochastic scaling factor to alter the size of the change applied to each variable during the update process.
ε	Small positive value to avoid division by zero, typically 10^{-8} .
X_{best}	The best solution found so far. Used to guide the update of current solutions.

3.1.3. Exploration Phase

The exploration phase is responsible for searching for solutions in different regions of the search space, with the goal of avoiding becoming trapped in local optima. This is accomplished using the multiplication and division operators, according to Equation (6) and conditioned on the MOA value:

$$x_{ij}(C_Iter + 1) = \begin{cases} \text{best}(x_j) \div (MOP + \varepsilon) \times ((UB_j - LB_j) \times \mu + LB_j), & \text{if } r_2 < 0.5 \\ \text{best}(x_j) \times MOP \times ((UB_j - LB_j) \times \mu + LB_j), & \text{otherwise.} \end{cases} \quad (6)$$

3.1.4. Mathematical Optimizer Probability

The Mathematical Optimizer Probability (MP) coefficient is adaptive, being defined in terms of *C_Iter*, and works as a control factor that regulates the transition between the exploration and exploitation stages of the optimization process.

$$MOP(C_Iter) = 1 - \frac{C_Iter^{1/\alpha}}{M_Iter^{1/\alpha}} \quad (7)$$

3.1.5. Exploitation Phase

The exploitation phase is activated when a random number $r_1 \leq \text{MOA}$, marking the transition from global exploration to local improvement. In this stage, the AOA applies the Subtraction and Addition operators using Equation (8) in order to intensify the search around the current optimum, while a stochastic component is introduced in each iteration to preserve some exploration, especially at the beginning and end:

$$x_{ij}(C_Iter + 1) = \begin{cases} \text{best}(x_j) - MOP \times ((UB_j - LB_j) \times \mu + LB_j), & r_3 < 0.5 \\ \text{best}(x_j) + MOP \times ((UB_j - LB_j) \times \mu + LB_j), & \text{otherwise} \end{cases} \quad (8)$$

where $\text{best}(x_j)$ denotes the *j*-th component of the current best solution, while UB_j, LB_j denote the upper and lower bounds of dimension *j*. For each dimension *j*, UB_j, LB_j are defined as constant values before the optimization process begins. These bounds delimit the feasible search space for each variable x_j , ensuring that the generated solutions remain within problem-specific limits.

3.1.6. Pseudocode of the Original Arithmetic Optimization Algorithm

The AOA Algorithm 1 is presented below. In the exploratory stage, the algorithm applies multiplication and division operations. In the exploitation phase, it employs addition and subtraction to focus the search around the most promising solutions, gradually improving the quality of the results. This balanced dynamic enables the AOA to effectively guide the optimization process, combining global diversification and local convergence within a numerically simple yet highly efficient framework.

Algorithm 1 Arithmetic Optimization Algorithm.

```

1: Initialize the AOA parameters in Table 1
2: Initialize positions of all ( $n$ ) randomly (Solutions:  $i = 1, \dots, m$ ), in Equation (4)
3: while  $C_{Iter} < M_{Iter}$  do
4:   Calculate the objective function for the given solutions
5:   find best ( $best$ ) solution
6:   Update MOA using Equation (5)
7:   Update MOP using Equation (7)
8:   for  $i = 1$  to  $N$  do
9:     for  $j = 1$  to  $d$  do
10:      Generate random numbers  $r_1, r_2, r_3 \in [0, 1]$ 
11:      if  $r_1 > MOA$  then
12:        if  $r_2 > 0.5$  then
13:          Apply Division operator ( $\div$ ).
14:          Update position using Equation (6), Rule 1
15:        else
16:          Apply Multiplication operator ( $\times$ ).
17:          Update position using Equation (6), Rule 2
18:        end if
19:      else
20:        if  $r_3 > 0.5$  then
21:          Apply Subtraction operator ( $-$ ).
22:          Update position using Equation (8), Rule 1
23:        else
24:          Apply Addition operator ( $+$ ).
25:          Update position using Equation (8), Rule 2
26:        end if
27:      end if
28:    end for
29:  end for
30:   $C_{Iter} \leftarrow C_{Iter} + 1$ 
31: end while
32: return Best solution  $x$ 
  
```

4. Two-Step Binarization Scheme

This process consists of two main phases:

- (a) Transfer phase: The transformation of a real-valued variable into a value within the range $[0, 1]$. These functions are shown in Table 2
- (b) Binarization phase: The mapping of this probabilistic value into a binary value in $\{0, 1\}$. These functions are shown in Tables 3 and 4.

Though structurally simple, these two components play a critical role in shaping the exploration and exploitation behavior of a binary metaheuristic, and as such have a significant impact on the overall performance of the algorithm [12].

Here, we adopt S-shaped and V-shaped transfer functions together with the binarization rules STD, COM, PS, ELIT and ELITR. This is because they capture two key behaviors: cases where mapping continuous values to binary decisions cases where changing the

sign should flip the decision (S-shaped, with $T(0) \approx 0.5$), and cases where the magnitude matters regardless of the sign (V-shaped, as a function of $|d|$). Here, d denotes the continuous value prior to binarization in the current iteration t ; applying $T(\cdot)$ yields a probability in $[0, 1]$ of the binarization rules mapping to a binary decision (0/1). These families are monotone, smooth, bounded, and computationally efficient, allowing the exploration–exploitation balance to be adjusted gradually over iterations. For the SCP, the results of the OR-Library instances support these choices; comparative analyses show that the choice of binarization is crucial to performance [13], while systematic studies report that the binarization rule explains more performance variability than the transfer function family.

Table 2. Transfer functions (S-shaped and V-shaped); $T(x) \in [0, 1]$ is a probability, while x is the individual’s continuous position.

Name	Formula	Rationale (Reason/Justification)
S1	$T(x) = \frac{1}{1 + e^{-2x}}$	Steeper logistic: More decisive mapping of the sign and magnitude of x ; yields higher flip probabilities for moderate $ x $ and accelerates early exploration without saturating too quickly.
S2	$T(x) = \frac{1}{1 + e^{-x}}$	Standard logistic: Balanced and stable mapping commonly used as a baseline; provides moderate flip probabilities across a wide range of $ x $ and promotes steady convergence.
S3	$T(x) = \frac{1}{1 + e^{-x/2}}$	Smoother logistic: Gentler slope that damps abrupt changes in probability; reduces over-correction and oscillations, which helps to avoid spurious bit flips in noisy updates.
S4	$T(x) = \frac{1}{1 + e^{-x/\sqrt{3}}}$	More conservative: Slower growth that delays bit fixing and mitigates premature convergence; favors exploitation in later stages while keeping low probability for small $ x $.
V1	$T(x) = \left \operatorname{erf}\left(\sqrt{\frac{2}{\pi}} x\right) \right $	Very smooth in $ x $: Near-linear response around the origin with gradual saturation; allows fine-grained probability modulation for small and medium $ x $.
V2	$T(x) = \tanh(x) $	Fast rise and saturation: Probability increases quickly once $ x $ grows and then plateaus; useful for decisive updates and rapid transitions from exploration to exploitation.
V3	$T(x) = \frac{ x }{\sqrt{1 + x^2}}$	Controlled monotone growth: Strictly increasing with horizontal asymptote at 1; avoids early saturation while keeping a smooth derivative, yielding stable adjustments.
V4	$T(x) = \frac{2}{\pi} \arctan\left(\frac{\pi}{2} x \right)$	Intermediate curvature: Compromise between the speed of V2 and the smoothness of V1; the monotone bounded slope provides a good tradeoff for general-purpose use.

Table 3. Summary of binarization rules (STD, COM, PS, ELIT, ELITR). Transfer function $T(x) \in [0, 1]$; random variate $\text{rand} \sim \mathcal{U}(0, 1)$.

Rule (Acronym)	Equation	Advantages
Standard (STD)	(9)	Simple and unbiased; flip probability controlled directly by $T(x)$; stable baseline and easy to tune.
Complement (COM)	(10)	Adds diversity by flipping current bit under control of $T(\cdot)$; helps to escape local minima.
Static Probability (PS)	(11)	Noise-robust thresholds; preserves bit in a middle band; α tunes conservativeness.
Elitist (ELIT)	(12)	Bias towards current best; faster convergence and fewer late random oscillations.
Elitist Roulette (ELITR)	(13)	Uses elite set via fitness-proportional sampling; balances exploitation and diversity.

Table 4. Binarization rules. Transfer function $T(x) \in [0, 1]$; $\text{rand} \sim \mathcal{U}(0, 1)$.**Binarization Functions.**

- (a) Standard (STD). Equation (9)
- (b) Complement (COM). Equation (10)
- (c) Probability (PS). Equation (11)
- (d) Elitist (ELIT). Equation (12)
- (e) Elitist Roulette (ELITR). Equation (13)

$$b_j^{\text{new}} = \begin{cases} 1, & \text{if } \text{rand} \leq T(d_w^j), \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

$$b_j^{\text{new}} = \begin{cases} \text{comp}(b_w^j), & \text{if } \text{rand} \leq T(d_w^j), \\ 0, & \text{otherwise,} \end{cases} \quad \text{with } \text{comp}(0) = 1, \text{comp}(1) = 0. \quad (10)$$

$$b_j^{\text{new}} = \begin{cases} 0, & \text{if } T(d_w^j) \leq \alpha, \\ b_w^j, & \text{if } \alpha < T(d_w^j) \leq \frac{1+\alpha}{2}, \\ 1, & \text{if } T(d_w^j) > \frac{1+\alpha}{2}. \end{cases} \quad (11)$$

$$b_j^{\text{new}} = \begin{cases} b_{\text{best}}^j, & \text{if } \text{rand} < T(d_w^j), \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

$$b_j^{\text{new}} = \begin{cases} b_s^j, & \text{if } \text{rand} \leq T(d_w^j), \\ 0, & \text{otherwise,} \end{cases} \quad \text{Pr}(s) = \frac{f(s)}{\sum_{u \in \Omega_\sigma} f(u)}, \quad s \in \Omega_\sigma. \quad (13)$$

where:

b_j^{new}	Resulting bit at position j
b_w^j	Current bit of individual w at position j
b_{best}^j	Bit j of the best-known individual
d_w^j	Continuous value (input to T) associated with (w, j)
$T(\cdot)$	Transfer function $[\cdot] \rightarrow [0, 1]$ returning a probability
$\text{rand} \sim \mathcal{U}(0, 1)$	Uniform random number in $[0, 1]$
$\alpha \in [0, 1]$	Threshold used in PS
Ω_σ	Elite set of size σ , with $s \in \Omega_\sigma$ the selected elite index, b_s^j bit j of elite s , and $f(s)$ its fitness

5. Binary AOA

In this section, we present the Binary Arithmetic Optimization Algorithm (BAOA), an adaptation of the AOA to the binary domain in order to solve the SCP. This design keeps the operation, structure, and parameters of the original algorithm intact, on which basis we incorporate a two-step binarization scheme consisting of transfer functions and binarization rules to transform continuous positions into $\{0, 1\}$ vectors. In addition, a greedy repair operator guarantees the feasibility of the solutions and eliminates redundancies without altering the logic of the base method. We detail the execution flow by iterations, the alternation between exploration and exploitation, the notation used, and the inputs and outputs of the procedure. This structure allows us to use the search dynamics of the

arithmetic algorithm and makes it operational in discrete spaces, preserving its essence and producing viable solutions for the SCP.

The algorithm description is as follows: the BAOA operates in iterations in which the arithmetic core (kept intact) updates the continuous population, determining the degree of exploration or exploitation through its adaptive control. The parameters are the same as in the original algorithm; see Table 1. Then, the binarization procedure transforms these positions into $\{0, 1\}$ candidates using the Algorithm 2, while the repair operator ensures feasibility and eliminates redundancies using Algorithm 3.

Algorithm 2 Two-Step Binarization Scheme with Specific Rule.

```

1: Input: Continuous vector  $\mathbf{d}_w = [d_w^1, d_w^2, \dots, d_w^n]$ 
2: Output: Binary vector  $\mathbf{b}_{\text{new}} = [b_{\text{new}}^1, b_{\text{new}}^2, \dots, b_{\text{new}}^n]$ 
3: for  $j = 1$  to  $n$  do
4:   Compute  $T(d_w^j)$  using the selected transfer function
5:   Apply the binarization rule specified
6:   Assign the result to  $b_{\text{new}}^j$ 
7: end for
8: return  $\mathbf{b}_{\text{new}}$ 

```

Where:

j Index of the dimension (loop variable), $j = 1, \dots, n$
 w Index of the individual (solution) in the population
 $\mathbf{d}_w$ Continuous solution vector $\mathbf{d}_w = [d_w^1, d_w^2, \dots, d_w^n] \in \mathbb{R}^n$
 $T(d_w^j)$ Transfer function that converts the continuous value d_w^j into a probability
 $\mathbf{b}_{\text{new}}$ Resulting binary vector $\mathbf{b}_{\text{new}} = [b_{\text{new}}^1, \dots, b_{\text{new}}^n] \in \{0, 1\}^n$

Algorithm 3 Greedy Repair Operator for the SCP.

```

1: Input: Binary vector  $\mathbf{b}$ , matrix  $A$ , cost vector  $\mathbf{c}$ 
2: for each row  $i$  not covered by  $\mathbf{b}$  do
3:   Select  $j^*$  with  $a_{ij} = 1$  and minimal  $c_j$ 
4:   Set  $b_{j^*} \leftarrow 1$ 
5: end for
6: for each  $j$  with  $b_j = 1$  do
7:   Set  $b_j \leftarrow 0$ 
8:   if any row becomes uncovered then
9:     Restore  $b_j \leftarrow 1$ 
10:  end if
11: end for
12: return repaired vector  $\mathbf{b}$ 

```

Finally, the cost is evaluated and the best found solution is updated. The inputs to the procedure are the cost and coverage matrix, the population size, the stopping criterion, and the parameters of the original arithmetic algorithm along with the choice of transfer function and binarization rule (outer layers); the output is the best feasible binary solution and its cost. The complete process is presented in Algorithm 4.

Algorithm 4 Binary Arithmetic Optimization Algorithm with Greedy Repair.

```

1: Initialize AOA parameters and positions of all solutions randomly ( $i = 1, \dots, N$ )
2: Apply Two-Step Binarization Scheme to initial positions, using Algorithm 2
3: Apply Greedy Repair Operator to all solutions, using Algorithm 3
4: while  $C_{\text{iter}} < M_{\text{iter}}$  do
5:   Evaluate fitness for all solutions
6:   Identify the best solution found so far
7:   Update MOA
8:   Update MOP
9:   for  $i = 1$  to  $N$  do
10:    for  $j = 1$  to  $d$  do
11:      Generate  $r_1, r_2, r_3 \in [0, 1]$ 
12:      if  $r_1 > \text{MOA}$  then
13:        if  $r_2 > 0.5$  then
14:          Apply Division operator ( $\div$ )
15:          Update position using Equation (6), Rule 1
16:        else
17:          Apply Multiplication operator ( $\times$ )
18:          Update position using Equation (6), Rule 2
19:        end if
20:      else if
21:        if  $r_3 > 0.5$  then
22:          Apply Subtraction operator ( $-$ )
23:          Update position using Equation (8), Rule 1
24:        else
25:          Apply Addition operator ( $+$ )
26:          Update position using Equation (8), Rule 2
27:        end if
28:      end if
29:    end for
30:    Apply Two-Step Binarization Scheme to  $x_i$ , using Algorithm 2
31:    Apply Greedy Repair Operator to  $b_i$ , using Algorithm 3
32:  end for
33:   $C_{\text{iter}} \leftarrow C_{\text{iter}} + 1$ 
34: end while
35: return Best repaired binary solution

```

Computational Complexity Analysis

The computational complexity of the BAOA for the SCP is estimated as $O(T \cdot P \cdot m \cdot n)$, where T is the maximum number of iterations, P is the population size, and $m \times n$ are the dimensions of the coverage matrix. This estimate encompasses the full procedure: the arithmetic updates over n variables drive the search, binarization is applied to each individual with a per-iteration cost on the order of $O(P \cdot n)$, and feasibility is evaluated with greedy repair on the problem matrix with a per-iteration cost on the order of $O(P \cdot m \cdot n)$. The latter term becomes dominant in combinatorial covering settings, meaning that the binarization cost is absorbed into the final order. Presenting the complexity in this way aligns with metaheuristic studies that explicitly analyze runtime using Big-O notation and detail the dependence on population size, problem dimensionality, and iteration count [14].

6. Experiments Results

In this section, we discuss both the experimental method and the obtained experimental results.

The experiments were executed on a MacBook Pro (15-inch, 2019) with the following specifications:

- Processor: 2.3 GHz 8-core Intel Core i9
- Graphics: Radeon Pro 560X 4 GB + Intel UHD Graphics 630 1536 MB
- Memory: 16 GB DDR4 at 2400 MHz
- Operating System: macOS Sequoia 15.5

6.1. Experimental Methodology

The experimental analysis of the BAOA was conducted using SCP instances from the OR-Library, which are widely used for their diversity in terms of size and complexity. Table 5 presents the considered set of instances, detailing the number of instances, dimensions of the problem (m rows and n columns), cost range, density of the coverage matrix, and availability of optimal solutions. These instances allow us to evaluate the algorithm's performance in heterogeneous scenarios, from cases with known optimal solutions (sets 4, 5, 6, A, B, C, D, NRE, and NRF) to more challenging problems for which only the best historical solution is available (NRG, NRH, and unicast).

For each instance and experimental configuration, 30 independent runs were performed; this number is considered sufficient to draw reliable statistical conclusions in metaheuristic studies. The BAOA parameters were established according to a previous limited study, with the results shown in Table 5. The evaluation metrics considered here were the value of the best found solution, the RPD (Relative Percentage Deviation) from the known optimum or best historical solution, the total execution time, and the Coefficient of Variation (CV) as a measure of stability [15].

The stopping criterion was identical across all instances, and was defined by the number of evaluations. This approach ensures fairness compared to alternatives such as elapsed time, which are highly dependent on machine performance.

Table 5. Description of the OR-Library SCP benchmark sets, including the number of instances, problem size, cost range, density, and optimal solution status.

Instance Family	Number of Instances	m	n	Cost Range	Density (%)	Optimal Solution
4	10	200	1000	[1, 100]	2.00	known
5	10	200	2000	[1, 100]	2.00	known
6	5	200	1000	[1, 100]	5.00	known
A	5	300	3000	[1, 100]	2.00	known
B	5	300	3000	[1, 100]	5.00	known
C	5	400	4000	[1, 100]	2.00	known
D	5	400	4000	[1, 100]	5.00	known
NRE	5	500	5000	[1, 100]	10.00	known
NRF	5	500	5000	[1, 100]	20.00	known
NRG	5	1000	10,000	[1, 100]	2.00	unknown
NRH	5	1000	10,000	[1, 100]	5.00	unknown

Finally, the obtained results were analyzed using descriptive statistics and compared using nonparametric tests according to the methodology from [9,13], as shown in Figure 1.

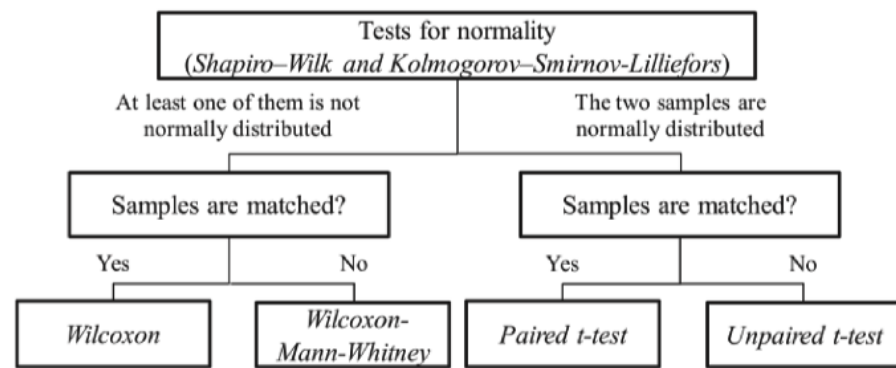


Figure 1. Statistical decision flow used for test selection.

6.2. Parameter Setting

The Table 6 shows the bounded execution of the BAOA on a sample of one instance per set. The execution parameters of the BAOA are the same as in the original version. The results were of good quality when considering RPD as a quality measure; thus, we used them in our experiments while only increasing the population size and number of iterations. The parameter settings are shown in Table 7.

Table 6. Comprehensive summary of BAOA results per SCP instance set (V3-ELIT) over 100 iterations, as used in the parameter-setting experiments.

Instance	m	n	Density (%)	Opt	Min	Max	Avg	Best RPD (%)	Total Time (min)
scp41	200	1000	2.0	429	433	463	444.75	0.93	7.62
scp51	200	2000	2.0	512	524	563	535.09	2.34	12.53
scp61	200	1000	5.0	138	141	151	144.25	2.17	5.61
scpa1	300	3000	2.0	253	267	278	269.47	5.53	25.52
scpb1	300	3000	5.0	279	284	306	294.35	1.79	17.88
scpc1	400	4000	2.0	146	160	152	150.58	9.59	48.12
scpd1	400	4000	5.0	60	60	65	62.47	0.00	30.87
scpnre1	500	5000	10.0	29	29	31	29.46	0.00	39.74
scpnrf1	500	5000	20.0	14	15	15	14.28	7.14	35.66
scpnrg1	1000	10000	2.0	63	64	69	66.64	1.59	326.13
scpnrh1	1000	10000	5.0	55	55	60	57.55	0.00	197.25

Table 7. BAOA parameter settings.

Parameter	Value
LB	1
UB	−1
M_Iter	100 for Setting Parameter/500 for experiments
P	Population size: 150 for Setting Parameter/200 for experiments
Number of executions	31
Min	0.2
Max	1.0.
α	$\alpha = 5$.
r_1, r_2, r_3, μ	A random number in $[0, 1]$
ε	10^{-8}

6.3. Statistical Indicators for Performance Evaluation

In this section, we present the indicators used to measure the quality of our solutions.

- (a) Relative Percentage Deviation (RPD). As a quality indicator, we use the average Relative Percentage Deviation (RPD), which measures the proximity of a candidate solution to the known optimum:

$$RPD = \frac{Z_{\text{alg}} - Z_{\text{ref}}}{Z_{\text{ref}}} \times 100 \quad (14)$$

where Z_{alg} is the objective function value returned by the algorithm under evaluation [16] and Z_{ref} is the best known or optimal value for the problem instance.

- (b) Coefficient of Variation (CV). The CV evaluates the stability of the algorithm over multiple independent runs [16]. It is defined as the ratio between the standard deviation and the mean of the results. A lower CV indicates greater consistency. The equation for CV is

$$CV = \frac{\sigma}{\mu} \times 100. \quad (15)$$

The CV, defined as the ratio between the standard deviation (σ) and the mean (μ), enables a relative assessment of stability across instances.

6.4. Performance Analysis of the BAOA

This section presents a comprehensive summary of our BAOA's performance in all benchmark instances. The statistical analysis is shown in Tables 8 and 9 (Min, Max, Avg, CV, RPD), while Figures 2–5 show the most effective combinations of instance-based binarization.

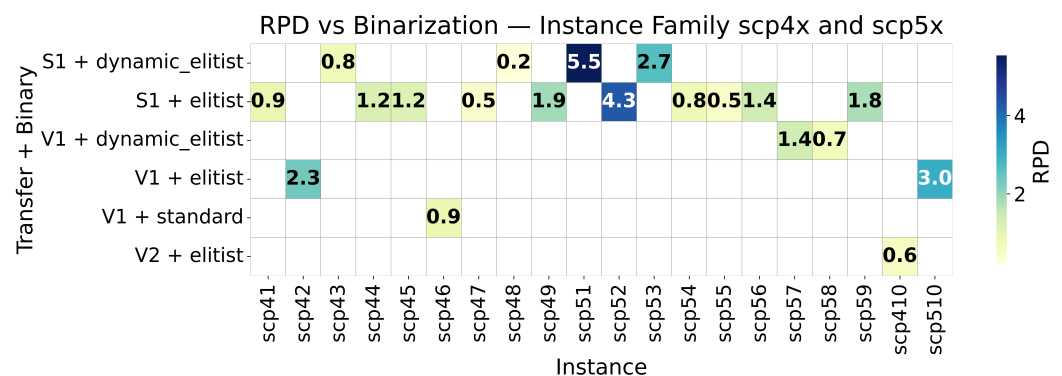


Figure 2. RPD heatmap for TF–binarization combinations; lighter cells indicate lower RPD.

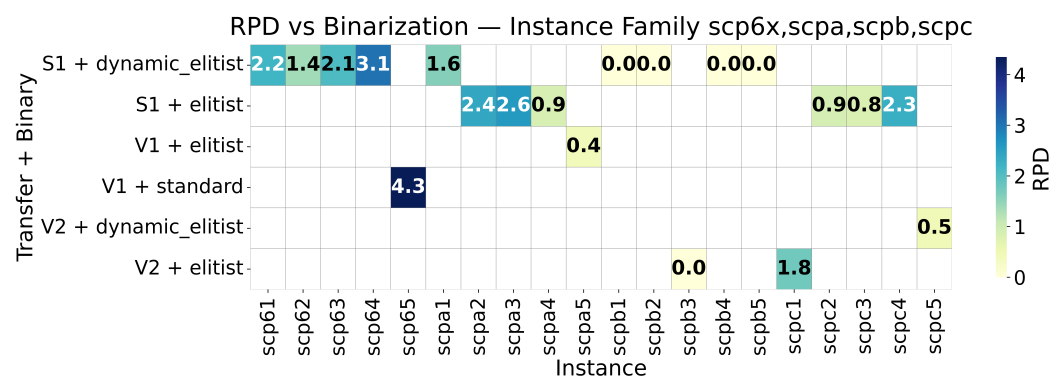


Figure 3. RPD heatmap for TF–binarization combinations; lighter cells indicate lower RPD.

Table 8. BAOA results on SCP (Group 1). Columns: Opt, Min (best), Max, Avg, CV, and RPD.

Inst	Opt	Min	Max	Avg	CV	RPD
scp41	429	433	463	444.75	2.44	0.93
scp42	512	524	563	535.09	1.81	2.34
scp43	516	520	567	528.25	2.16	0.78
scp44	494	500	543	519.82	2.59	1.21
scp45	512	518	563	537.29	3.40	1.17
scp46	560	565	615	585.16	2.93	0.89
scp47	430	432	472	447.28	3.15	0.47
scp48	492	493	533	510.22	3.09	0.20
scp49	641	653	705	673.99	1.71	1.87
scp410	514	517	556	536.92	2.37	0.58
scp51	253	267	278	269.47	0.96	5.53
scp52	302	315	332	322.94	1.29	4.30
scp53	226	232	246	237.68	2.29	2.65
scp54	242	244	265	255.46	2.53	0.83
scp55	211	212	232	221.48	3.41	0.47
scp56	213	216	234	225.21	1.95	1.41
scp57	293	297	322	309.34	2.46	1.37
scp58	288	290	316	302.25	2.39	0.69
scp59	279	284	306	294.35	2.48	1.79
scp510	265	273	291	280.16	2.11	3.02
scp61	138	141	151	144.25	1.65	2.17
scp62	146	148	160	152.58	1.87	1.37
scp63	145	148	159	150.68	1.68	2.07
scp64	131	135	144	138.36	2.14	3.05
scp65	161	168	177	174.20	1.11	4.35
scpa1	253	257	278	263.98	2.82	1.58
scpa2	252	258	277	264.16	1.21	2.38
scpa3	232	238	255	243.87	1.28	2.59
scpa4	234	236	257	241.28	2.22	0.85
scpa5	236	237	259	246.14	2.98	0.42
scpb1	69	69	75	71.77	2.65	0.00
scpb2	76	76	83	78.54	3.70	0.00
scpb3	80	80	87	82.64	2.38	0.00

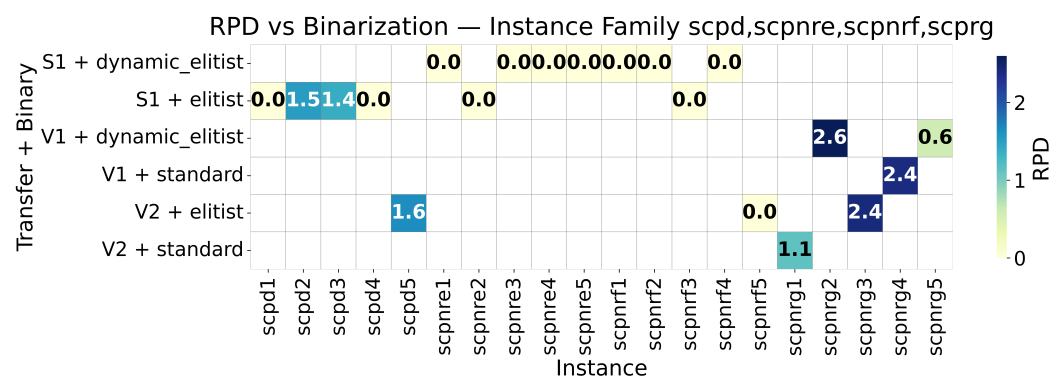
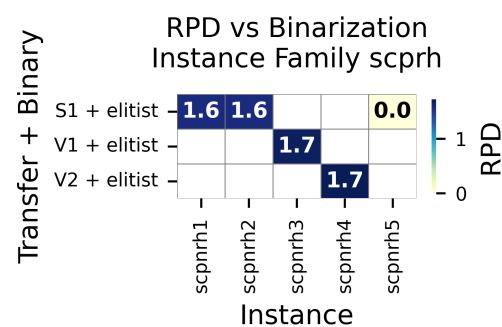
**Figure 4.** RPD heatmap for TF–binarization combinations; lighter cells indicate lower RPD.

Table 9. BAOA results on SCP (Group 2). Columns: Opt, Min (best), Max, Avg, CV, and RPD.

Inst	Opt	Min	Max	Avg	CV	RPD
scpb4	79	79	86	82.00	1.98	0.00
scpb5	72	72	78	74.59	3.05	0.00
scpc1	227	231	249	237.53	1.59	1.76
scpc2	219	221	240	228.10	2.20	0.91
scpc3	243	245	267	252.88	2.15	0.82
scpc4	219	224	240	230.24	1.73	2.28
scpc5	215	216	233	225.89	2.58	0.47
scpd1	60	60	65	62.47	2.20	0.00
scpd2	66	67	72	69.17	2.42	1.52
scpd3	72	73	79	76.12	1.78	1.39
scpd4	62	62	68	64.50	3.18	0.00
scpd5	61	62	67	63.73	1.74	1.64
scpnre1	29	29	31	29.46	1.70	0.00
scpnre2	30	30	32	31.33	2.44	0.00
scpnre3	27	27	29	28.13	1.47	0.00
scpnre4	28	28	30	28.99	2.39	0.00
scpnre5	28	28	30	28.71	2.62	0.00
scpnrf1	14	14	15	14.28	3.13	0.00
scpnrf2	15	15	16	15.49	3.23	0.00
scpnrf3	14	14	15	14.88	2.19	0.00
scpnrf4	14	14	15	14.57	3.40	0.00
scpnrf5	13	13	14	14.00	0.47	0.00
scpnrg1	176	178	193	185.24	1.93	1.14
scpnrg2	154	158	169	162.71	1.46	2.60
scpnrg3	166	170	182	176.42	1.77	2.41
scpnrg4	168	172	184	178.27	1.40	2.38
scpnrg5	168	169	184	177.78	1.75	0.60
scpnrh1	63	64	69	66.64	2.35	1.59
scpnrh2	63	64	69	66.64	2.00	1.59
scpnrh3	59	60	64	62.31	1.55	1.69
scpnrh4	58	59	63	61.52	1.65	1.72
scpnrh5	55	55	60	57.55	2.47	0.00

**Figure 5.** RPD heatmap for TF–binarization combinations; lighter cells indicate lower RPD.

We have used heatmaps to evaluate the results of different combinations between transfer functions and binarization rules; the results are as follows.

Figures 2–5 show heatmaps of the RPD values obtained from different combinations of transfer functions and binarization strategies for all SCP instance families; we have selected the best RPD results, in this case those below 5%. Lighter shades correspond to lower RPD values, i.e., solutions closer to the optimum.

The results reveal a clear trend, with the S1+elitist, V1+dynamic, and V2+elitist configurations achieving the lowest RPD values in most instances. In general, combinations with V transfer functions perform best. This indicates that elitist strategies are especially effective when combined with transfer functions that properly balance exploration and exploitation.

The results presented in Tables 10 and 11 show the average RPD, 95% confidence intervals [17], and Coefficient of Variation (CV) obtained by the V3-elitist combination across various SCP instances. Overall, the performance of this configuration shows stable convergence, with RPD values close to the known optima for each instance. The confidence intervals are narrow, indicating consistency across runs, and the CV is low in most instances.

Table 10. Average RPD results, 95% confidence intervals, and Coefficient of Variation (CV) (Group 1).

Instance	RPD Mean	RPD 95% CI	CV
scp41	2.6107	[2.2333, 2.9881]	0.1164
scp410	2.5292	[2.0167, 3.0416]	0.1632
scp42	6.3281	[5.8252, 6.8310]	0.0640
scp43	2.9845	[2.5154, 3.4536]	0.1266
scp44	4.4534	[3.4963, 5.4105]	0.1731
scp45	5.6250	[4.7439, 6.5061]	0.1262
scp46	2.4643	[2.1000, 2.8286]	0.1191
scp47	2.8837	[2.1035, 3.6639]	0.2179
scp48	2.4390	[2.0821, 2.7959]	0.1179
scp49	5.9594	[5.4219, 6.4970]	0.0726
scp51	7.1146	[6.5135, 7.7157]	0.0680
scp510	5.3585	[4.9665, 5.7505]	0.0589
scp52	8.2781	[7.7746, 8.7817]	0.0490
scp53	3.8938	[3.4341, 4.3535]	0.0951
scp54	4.2975	[3.8386, 4.7564]	0.0860
scp55	4.2654	[3.5447, 4.9861]	0.1361
scp56	4.6948	[3.0984, 6.2913]	0.2739
scp57	5.1877	[4.9982, 5.3772]	0.0294
scp58	4.0278	[2.6441, 5.4114]	0.2767
scp59	4.1577	[3.4828, 4.8326]	0.1307
scp61	3.6232	[2.5212, 4.7252]	0.2449
scp62	5.3425	[4.4108, 6.2741]	0.1404
scp63	2.7586	[2.7586, 2.7586]	0.0000
scp64	5.4962	[4.2604, 6.7320]	0.1811
scp65	10.0621	[9.4169, 10.7074]	0.0516
scpa1	4.5059	[3.8475, 5.1644]	0.1177
scpa2	4.8413	[4.4290, 5.2535]	0.0686
scpa3	5.5172	[4.6378, 6.3967]	0.1284
scpa4	4.8718	[3.4381, 6.3055]	0.2370
scpa5	4.4915	[3.8916, 5.0914]	0.1076
scpb1	2.3188	[1.3332, 3.3045]	0.3423
scpb2	0.7895	[−0.1054, 1.6843]	0.9129

In Table 12, considering the density of the OR-Library instances (low/medium in the SCP series (e.g., scp41, scp51, scp61) and high in the SCPA–SCPD series), the data show an increase in average execution time as density grows, from low values in scp61 (2.345 s) and scp51 (5.374 s) to the maximum observed in scp61 (64.522 s). In contrast, the stagnation indicators display heterogeneous variation across families and do not follow a monotonic pattern with respect to density.

Table 11. Average RPD results, 95% confidence intervals, and Coefficient of Variation (CV) (Group 2).

Instance	RPD Mean	RPD 95% CI	CV
scpb3	1.2500	[1.2500, 1.2500]	0.0000
scpb4	3.5443	[2.2293, 4.8593]	0.2988
scpb5	0.8333	[−0.1112, 1.7779]	0.9129
scpc1	6.2555	[5.7979, 6.7132]	0.0589
scpc2	6.5753	[5.7155, 7.4352]	0.1053
scpc3	3.3745	[2.7083, 4.0407]	0.1590
scpc4	7.1233	[6.8127, 7.4338]	0.0351
scpc5	4.0930	[3.6098, 4.5762]	0.0951
scpd1	6.0000	[4.1490, 7.8510]	0.2485
scpd2	1.5152	[1.5152, 1.5152]	0.0000
scpd3	6.6667	[4.7775, 8.5558]	0.2282
scpd4	0.6452	[−0.4518, 1.7421]	1.3693
scpd5	4.2623	[3.1474, 5.3772]	0.2107
scpnre1	0.0000	[0.0000, 0.0000]	nan
scpnre2	4.0000	[−0.5339, 8.5339]	0.9129
scpnre3	3.7037	[3.7037, 3.7037]	0.0000
scpnre4	0.7143	[−1.2689, 2.6975]	2.2361
scpnre5	0.0000	[0.0000, 0.0000]	nan
scpnrf1	0.0000	[0.0000, 0.0000]	nan
scpnrf2	0.0000	[0.0000, 0.0000]	nan
scpnrf3	1.4286	[−2.5378, 5.3949]	2.2361
scpnrf4	0.0000	[0.0000, 0.0000]	nan
scpnrf5	7.6923	[7.6923, 7.6923]	0.0000
scpnrg1	7.2727	[6.3529, 8.1926]	0.1019
scpnrg2	6.1039	[5.6623, 6.5455]	0.0583
scpnrg3	7.8313	[6.2446, 9.4181]	0.1632
scpnrg4	7.8571	[7.0475, 8.6668]	0.0830
scpnrg5	7.2619	[6.4523, 8.0715]	0.0898
scpnrh2	6.3492	[4.9556, 7.7428]	0.1768
scpnrh3	5.0847	[5.0847, 5.0847]	0.0000
scpnrh4	5.8621	[4.6895, 7.0346]	0.1611
scpnrh5	2.5455	[1.3089, 3.7820]	0.3912

Table 12. Average time: Average seconds per iteration. No-progress iteration: The iteration at which fitness shows no improvement. Stagnation ratio: No-progress iteration/total iterations.

Instance	Avg. Time (s)	No-Progress Iteration	Stagnation Ratio
scp41	3.302	160	32%
scp51	5.3740	150	30%
scp61	2.345	61	12.2%
scpa1	10.819	45	9%
scpb1	10.129	27	5.4%
scpc1	27.113	201	40.2%
scpd1	64.522	14	2.8%

The boxplots in Figure 6 show the distribution of the final RPD values in groups of ten instances. In most cases, the runs exhibit low variability and consistently converge toward near-optimal solutions, although some instances show greater dispersion, reflecting sensitivity to the problem structure. The Figures 7 and 8 show the performance metrics in terms of iteration time and convergence characteristics.

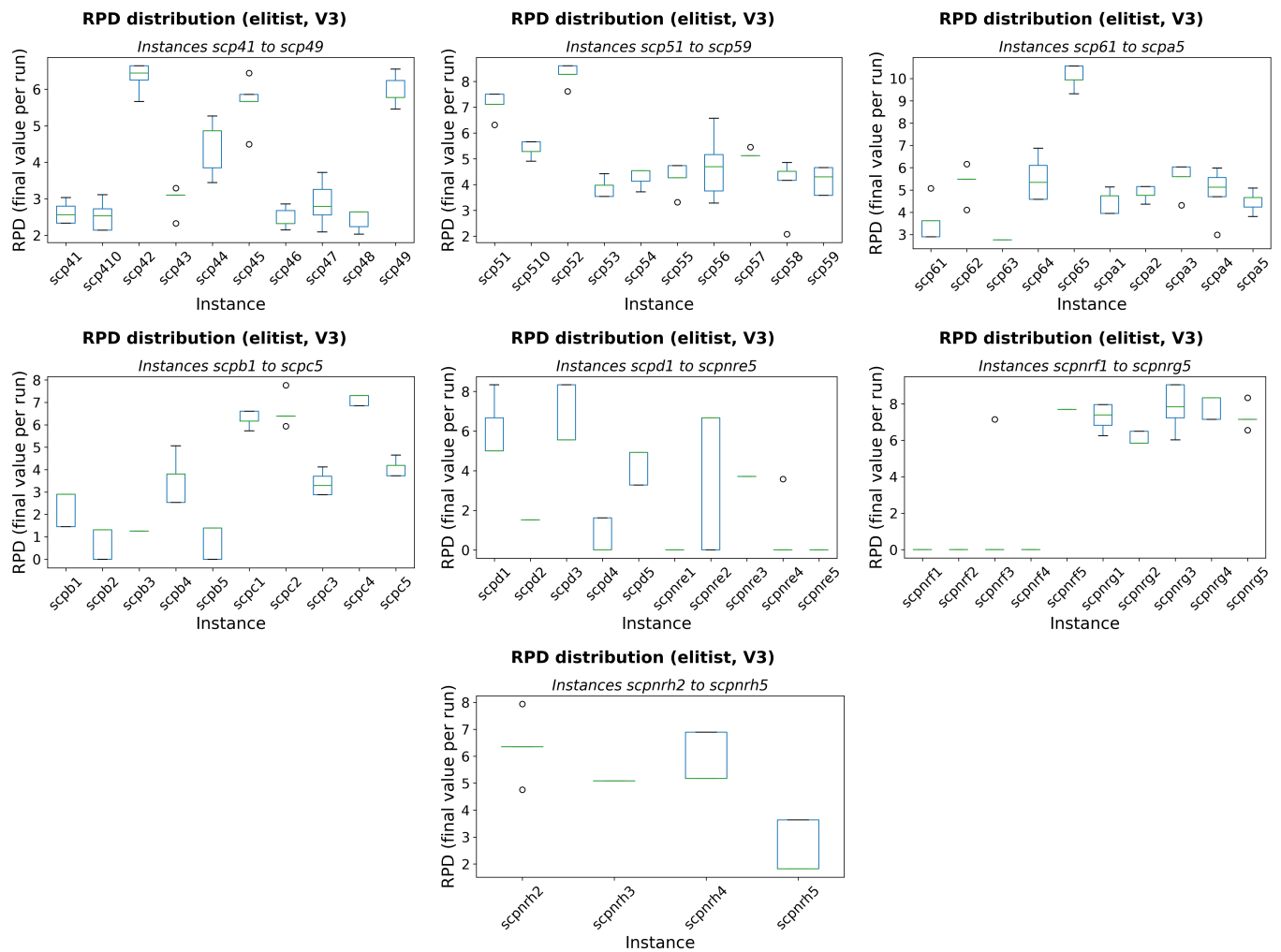


Figure 6. Boxplots RPD distribution for the *elitist* rule with the V3 transfer function (*scprnh2*–*scprnh5*). The blue box shows the interquartile range, the black line the median, the green whiskers the $1.5 \times \text{IQR}$ limits, and the white circles the outliers.

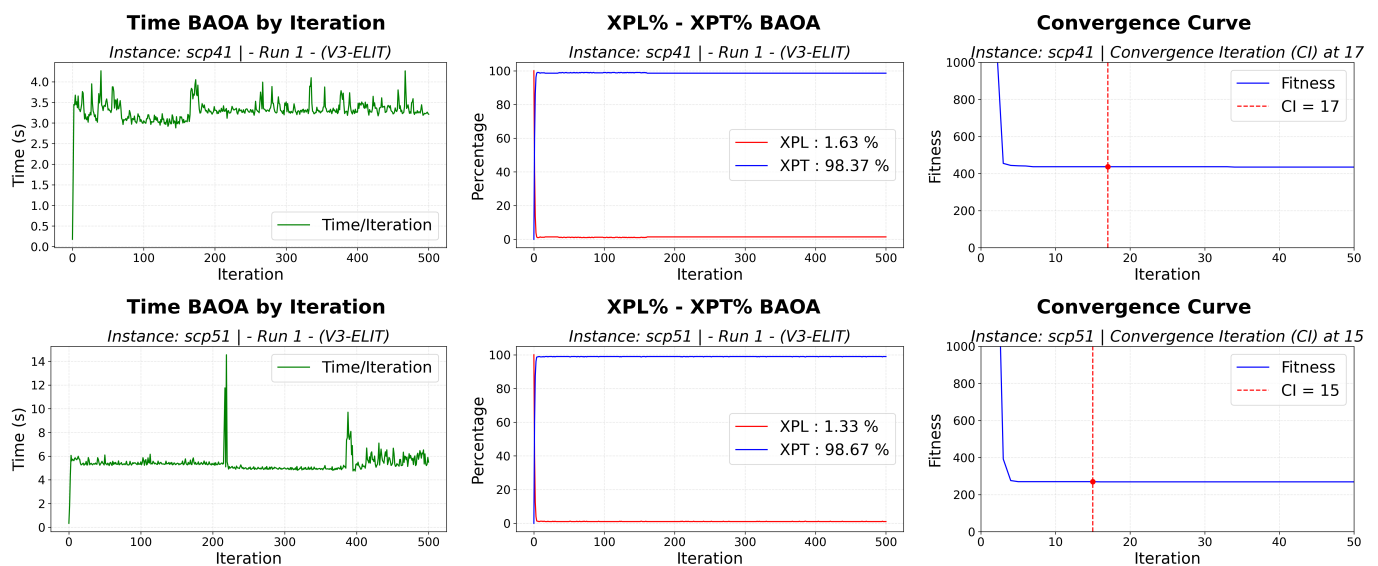


Figure 7. Cont.

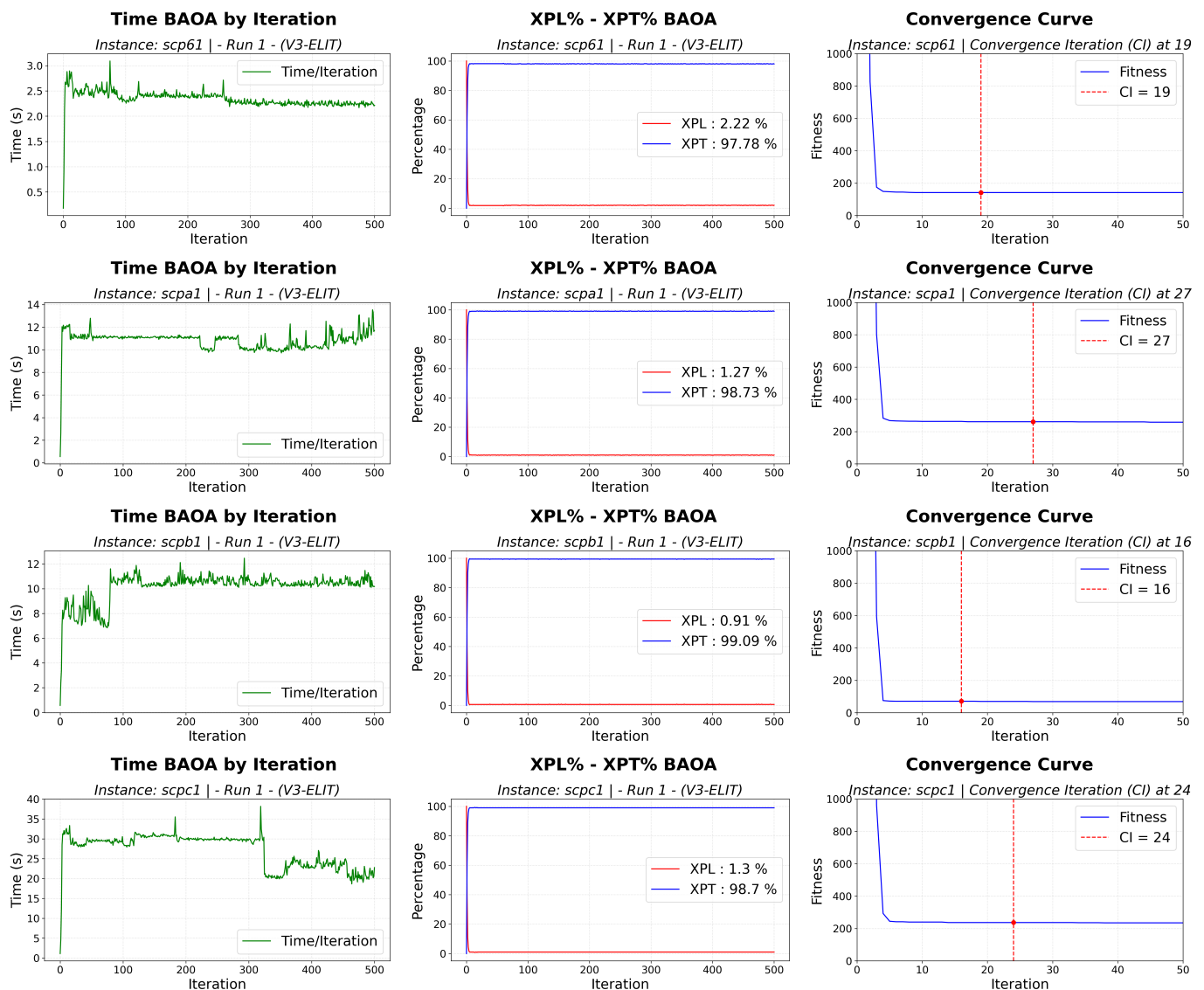


Figure 7. BAOA on (V3-ELIT): Time per iteration, XPL–XPT balance (averages), and convergence curve indicating the point from which the fitness no longer improves.

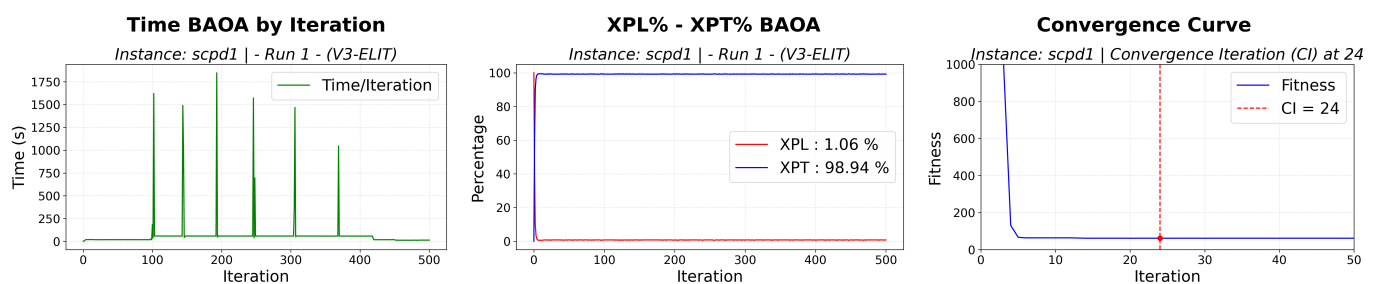


Figure 8. BAOA on (V3-ELIT): Time per iteration, XPL–XPT balance (averages), and convergence curve indicating the point from which the fitness no longer improves.

6.5. Benchmarking the BAOA with Competing Approaches

We compared our BAOA against recent competitive metaheuristics selected for their strong results in combinatorial optimization, namely, SCA, PSA, GWO, and BGO [18]. SCA provides a simple design with an effective exploration–exploitation balance [19], PSA is physics-inspired and performs well on discrete tasks [20], GWO is widely used in binary settings with robust convergence, and BGO is a recent variant tailored to binary spaces

that attains high-quality solutions [21]. The evaluation consistently compared the best cost achieved by each method on standard SCP instances, ensuring fairness and reproducibility. The experimental data were taken from [18]; the data are reported in Tables 13 and 14.

Figure 9 presents a comparative performance analysis of the evaluated metaheuristics. It should be noted that this table does not represent a statistical significance test but rather a descriptive comparison of results across problem instances. Three indicators are reported for each algorithm.

The results show that the BAOA exhibits the most competitive overall performance among the evaluated metaheuristics. In terms of solution quality, the BAOA achieves the lowest average minimum RPD (1.51), indicating solutions consistently closer to the optimal values. Regarding consistency, the BAOA also obtains the highest number of best instances (21), outperforming SCA (15), PSA (6), GWO (2), and BGO (1). Although its average rank (2.36) is not the lowest, the BAOA maintains a strong relative position across all instances. These findings highlight the BAOA's ability to combine high-quality solutions with robustness and reliability, positioning it as a competitive alternative to other state-of-the-art approaches.

Table 13. Comparative performance analysis of the evaluated metaheuristics.

MH	Avg Min RPD	Best Instances	Avg Rank
BAOA	1.51	21	2.36
SCA	2.25	15	1.67
PSA	1.83	6	1.84
GWO	2.13	2	1.80
BGO	2.15	1	1.93

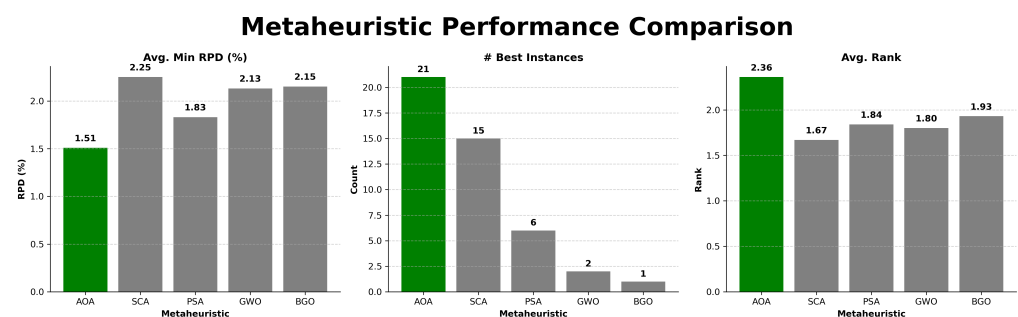


Figure 9. Barchart comparison of the metaheuristics based on three performance indicators.

6.6. Statistical Analysis

In this study, we used the RPD for the statistical analysis instead of absolute cost values. This indicator allows for fair comparisons between problem instances with different cost scales and provides a more consistent and impartial evaluation. Furthermore, the use of the RPD is well established in the optimization and metaheuristic literature, and is suitable for normality assessments such as the Shapiro–Wilk test [22].

We first assessed normality (Shapiro–Wilk and KS–Lilliefors) [23,24] on the paired differences between the BAOA and each metaheuristic. The hypotheses were as follows: H_0 , “the data are normal”; and H_1 , “the data are not normal”. In most cases $p < 0.05$; thus, H_0 was rejected. The results of this test are shown in Table 15.

Table 14. Performance comparison between the BAOA and recent metaheuristics (SCA, PSA, GWO, BGO) on benchmark SCP instances. The table reports the best cost and the RPD (%) for each method; lower values are better for both metrics, highlighting the BAOA’s competitiveness across instances.

Inst	Opt	BAOA			SCA			PSA			GWO			BGO		
		Min	Avg	RPD	Min	Avg	RPD	Min	Avg	RPD	Min	Avg	RPD	Min	Avg	RPD
41	429	433	444.75	0.93	431	433.75	0.466	431	433.78	0.466	433	434.0	0.932	433	433.03	0.932
42	512	517	536.92	0.58	523	527.0	21.48	517	528.29	0.977	518	526.55	11.72	518	525.10	11.72
43	516	524	535.09	2.34	520	521.06	0.775	520	521.47	0.775	520	520.88	0.775	520	520.41	0.775
44	494	520	528.25	0.78	496	504.45	4.049	496	506.58	2.142	499	505.42	1.012	499	504.48	1.012
45	512	500	519.82	1.21	514	518.29	0.391	518	519.68	1.172	518	518.13	1.172	518	518.13	1.172
46	560	518	537.29	1.17	564	567.81	0.714	565	569.0	0.893	565	567.77	0.714	567	567.18	1.250
47	430	505	585.16	0.89	432	434.29	0.465	433	434.26	0.698	432	434.0	0.465	433	433.97	0.698
48	492	432	447.28	0.47	493	494.06	0.203	493	493.84	0.203	492	493.84	0.203	492	493.84	0.203
49	641	493	510.22	0.20	655	663.77	21.84	656	667.52	23.40	654	662.77	20.28	653	662.10	18.72
410	514	653	673.99	1.87	517	522.68	0.584	517	523.42	0.973	517	523.42	0.973	517	524.06	0.584
51	253	267	269.47	5.53	267	267.77	5.53	257	267.03	1.58	267	267.48	5.53	267	267.48	5.53
52	302	315	322.94	4.30	315	319.50	4.30	313	319.12	3.64	315	319.09	4.30	315	319.09	4.30
53	226	232	237.68	2.65	230	232.03	1.77	229	231.84	1.33	232	232.00	2.65	232	232	2.65
54	242	244	255.46	0.83	244	248.32	0.83	244	247.90	0.83	244	248.10	0.83	244	248.09	0.82
55	211	212	221.48	0.47	212	214.45	0.47	212	213.42	0.47	212	213.06	0.47	212	213.06	0.47
56	213	216	225.21	1.41	216	223.35	1.41	216	223.35	1.41	216	221.90	1.41	216	221.90	1.40
57	293	297	309.34	1.37	296	302.19	1.02	297	301.81	1.37	299	301.29	2.05	299	301.29	2.04
58	288	290	302.25	0.69	290	297.52	0.69	290	297.61	0.69	290	297.39	0.69	290	297.38	0.69
59	279	284	294.35	1.79	284	288.13	1.79	284	288.26	1.79	284	286.23	1.79	284	286.22	1.79
510	265	273	280.16	3.02	272	274.42	2.64	272	273.87	2.64	273	274.03	3.02	273	274.03	3.01
61	138	141	143.39	2.17	141	144.10	2.17	141	143.03	2.17	141	142.23	2.17	141	142.23	2.17
62	146	148	150.48	1.37	148	151.06	1.37	148	150.10	1.37	148	150.42	1.37	148	150.42	1.37
63	145	148	149.23	2.07	148	150.03	2.07	147	149.23	1.38	148	148.61	2.07	148	148.61	2.07
64	131	134	135.39	2.29	135	135.39	3.05	135	135.13	3.05	134	135.23	2.29	134	135.23	2.29
65	161	172	174.87	6.83	165	174.81	2.48	172	174.52	6.83	171	174.48	6.21	171	174.48	6.21
a1	253	257	263.98	1.58	257	257.54	1.58	257	257.67	1.58	257	257.67	1.58	257	257.06	1.58
a2	252	258	264.16	2.38	258	262.25	2.38	258	263.12	2.38	258	261.38	2.38	258	261.12	2.38
a3	232	238	243.87	2.59	235	241	1.29	236	241.77	1.72	237	240.83	2.15	235	240.32	1.29
a2	234	236	241.28	0.85	236	237.48	0.85	236	237.06	0.85	236	236.74	0.85	236	236.61	0.85
a5	236	237	246.14	0.42	237	239.32	0.42	237	238.67	0.42	237	238.77	0.42	237	238.45	0.42
b1	69	76	78.54	0.1	69	70.48	0	69	70.64	0	69	70.25	0	69	70.22	0
b2	76	80	82.64	0.05	76	76.58	0	76	77.16	0	76	76.35	0	76	76.19	0
b3	80	80	82.0	0.0	80	81.22	0	80	81.25	0	80	81.12	0	81	81.16	1.25
b4	79	79	74.59	0.0	79	81.09	0	79	81.80	0	79	80.58	0	79	80.51	0
b5	72	72	74.59	0.0	72	72.38	0	72	72.54	0	72	72.29	0	72	72.54	0
c1	227	231	237.53	1.76	232	234.09	2.20	231	234.35	1.76	232	233.51	2.20	232	233.41	2.20
c2	219	221	228.10	0.91	221	224.51	0.91	221	225.00	0.91	221	224.16	0.91	221	223.74	0.91
c3	243	245	252.88	0.82	245	249.77	0.82	247	252.25	1.64	245	248.03	0.82	245	247.77	0.82
c4	219	224	230.24	2.28	224	226.96	2.28	224	228.83	2.28	221	226.58	0.91	222	225.51	1.36
c5	215	216	225.89	0.47	217	219.61	0.93	216	219.51	0.46	216	218.83	0.46	216	219.06	0.46
d1	60	60	62.470	0.000	60	619.355	0.000	60	618.065	0.000	60	62.129	0.000	60	621.613	1.6667
d2	66	67	69.170	1.520	67	682.258	1.5152	67	680.968	1.5152	67	68.129	1.5152	67	677.742	1.5152
d3	72	73	76.120	1.390	73	758.065	1.3889	74	762.903	1.3889	74	756.774	2.7778	74	758.387	2.7778
d4	62	62	64.500	0.000	62	630.968	0.000	62	636.774	0.000	62	632.581	0.000	62	628.387	0.000
d5	61	62	63.730	1.640	63	631.613	3.2787	63	632.903	1.6393	63	632.903	3.2787	63	630.323	3.2787

We then proceeded to the second stage, as the normality assumption was rejected in most cases according to the Shapiro–Wilk and Kolmogorov–Smirnov–Lilliefors tests. Therefore, following the methodological flow presented in Figure 1 and considering that the samples were paired across the same problem instances, we applied the Wilcoxon signed-rank [25] test as the appropriate non-parametric alternative to evaluate the significance of performance differences between our BAOA and the competing algorithms. The hypotheses were as follows: H_0 , “there are no statistically significant differences between the compared metaheuristics” (that is, both exhibit similar performance in terms of the minimum RPD value); and H_1 , “there are statistically significant differences between at least one pair of metaheuristics” (indicating differences in performance in terms of the minimum RPD value).

Table 15. Normality tests (Shapiro–Wilk and Kolmogorov–Smirnov–Lilliefors) applied to the differences between the BAOA and other algorithms (RPD); $p < 0.05$ indicates rejection of normality.

Set	Test	SCA	PSA	GWO	BGO
scp4x	Shapiro Lillie	$W = 0.704, p = 0.011$ $\text{stat} = 0.342, p = 0.052$	$W = 0.995, p = 0.994$ $\text{stat} = 0.155, p = 0.960$	$W = 0.674, p = 0.005$ $\text{stat} = 0.430, p = 0.002$	$W = 0.674, p = 0.005$ $\text{stat} = 0.430, p = 0.002$
scp5x	Shapiro Lillie	$W = 0.552, p = 0.0001$ $\text{stat} = 0.473, p = 0.001$	$W = 0.806, p = 0.091$ $\text{stat} = 0.267, p = 0.308$	$W = 1.000, p = 1.000$ –	$W = 0.552, p = 0.0001$ $\text{stat} = 0.473, p = 0.001$
scp6x	Shapiro Lillie	$W = 0.682, p = 0.006$ $\text{stat} = 0.436, p = 0.001$	$W = 0.882, p = 0.320$ $\text{stat} = 0.311, p = 0.129$	$W = 0.552, p = 0.0001$ $\text{stat} = 0.473, p = 0.001$	$W = 0.552, p = 0.0001$ $\text{stat} = 0.473, p = 0.001$
scpa	Shapiro Lillie	$W = 0.552, p = 0.0001$ $\text{stat} = 0.473, p = 0.001$	$W = 0.552, p = 0.0001$ $\text{stat} = 0.473, p = 0.001$	$W = 0.552, p = 0.0001$ $\text{stat} = 0.473, p = 0.001$	$W = 0.552, p = 0.0001$ $\text{stat} = 0.473, p = 0.001$
scpb	Shapiro Lillie	$W = 0.771, p = 0.046$ $\text{stat} = 0.349, p = 0.044$	$W = 0.771, p = 0.046$ $\text{stat} = 0.349, p = 0.044$	$W = 0.771, p = 0.046$ $\text{stat} = 0.349, p = 0.044$	$W = 0.620, p = 0.001$ $\text{stat} = 0.448, p = 0.001$
scpc	Shapiro Lillie	$W = 0.698, p = 0.009$ $\text{stat} = 0.367, p = 0.025$	$W = 0.562, p = 0.0002$ $\text{stat} = 0.470, p = 0.001$	$W = 0.767, p = 0.043$ $\text{stat} = 0.402, p = 0.008$	$W = 0.828, p = 0.134$ $\text{stat} = 0.370, p = 0.024$
scpd	Shapiro Lillie	$W = 0.555, p = 0.0001$ $\text{stat} = 0.472, p = 0.001$	$W = 0.745, p = 0.027$ $\text{stat} = 0.344, p = 0.049$	$W = 0.731, p = 0.020$ $\text{stat} = 0.366, p = 0.027$	$W = 0.758, p = 0.035$ $\text{stat} = 0.299, p = 0.174$

No statistically significant differences were found between any pair of algorithms, as all p -values were greater than 0.05 the results are show in Table 16. Therefore, the null hypothesis is accepted in all comparisons, indicating statistically comparable performance. These results suggest that additional evaluation criteria such as convergence speed or robustness could provide further insights into the algorithms' relative effectiveness.

Table 16. Wilcoxon signed-rank test p -values for the BAOA compared with other algorithms (RPD); a value of **ns** indicates $p \geq 0.05$ (no significant difference).

Set	SCA	PSA	GWO	BGO
scp4x	0.812 (ns)	0.812 (ns)	0.812 (ns)	0.812 (ns)
scp5x	0.317 (ns)	0.109 (ns)	NA	0.317 (ns)
scp6x	0.655 (ns)	0.655 (ns)	0.317 (ns)	0.317 (ns)
scpa	0.317 (ns)	0.317 (ns)	0.317 (ns)	0.317 (ns)
scpb	0.180 (ns)	0.180 (ns)	0.180 (ns)	1.000 (ns)
scpc	0.180 (ns)	0.655 (ns)	0.593 (ns)	0.593 (ns)
scpd	1.000 (ns)	0.109 (ns)	0.285 (ns)	0.144 (ns)

6.7. Conclusions from the Statistical Tests

The Shapiro–Wilk test showed that most RPD distributions were non-normal, justifying the use of non-parametric methods. The Wilcoxon signed-rank test revealed no statistically significant differences between the metaheuristics, as all p -values were above 0.05. Therefore, the null hypothesis could not be rejected. These results suggest that the algorithms have comparable performance based on minimum RPD; thus, further analysis using additional performance metrics is recommended.

7. Analysis of Computational Overhead and Methodological Limitations

We chose V3-ELIT to develop this section due to its ability to balance exploration and exploitation. The V3 (hyperbolic V-shaped) transfer function introduces smooth and controlled transitions in the binarization, while the ELIT rule preserves the best solutions, which in our experiments translated into low RPDs and practically zero variability (mean = best value; deviation around 0) in most instances. On this basis, Figure 10 shows the time per iteration of BAOA V3-ELIT across several SCP instances, revealing two clear

phases: a rapid initial overhead, and a subsequent stabilization phase in which times remain nearly constant. The largest instances, such as *scpc1* (400×4000) and *scpd1* (400×4000), reach the highest costs, with 30.0 s and 19.0 s per iteration, respectively, while *scpa1* (300×3000) and *scpb1* (300×3000) fall in an intermediate range, with 15.0 s and 11.0 s. In contrast, smaller instances such as *scp61* (200×1000) and *scp51* (200×2000) require only 4.5 s and 3.3 s per iteration. This pattern confirms that the computational cost increases with problem size; however, once past the initial phase, the time remains stable, reflecting temporal consistency of the algorithm across instances of different scales.

Across the hardest SCP cases, BAOA V3-ELIT exhibits a short transient followed by stable timing and a strongly exploitative regime. As shown in Figure 11, the iteration time settles ~ 18 – 19 s after the early iterations. In Figure 12, exploration quickly collapses to about 2.12% while exploitation stabilizes near 97.88%, evidencing minimal search volatility and a persistent bias toward exploitation once the algorithm stabilizes.

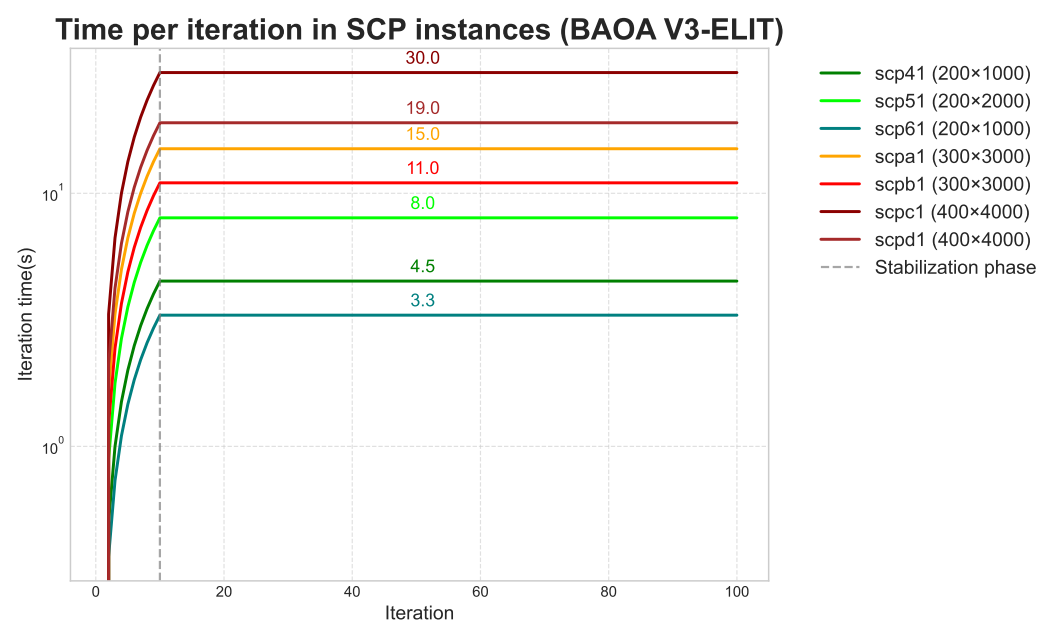


Figure 10. Summary of iteration times of AOA V3-ELIT across different SCP instances. Two stages can be observed: an initial overhead, and a stabilization phase with nearly constant time.

Table 17 presents the theoretical computational complexity of the BAOA algorithm expressed in terms of population size (P), number of iterations (T), and dimensions of the coverage matrix (m rows and n columns) for the Set Covering Problem (SCP). As discussed previously, this complexity is represented as $O(P \cdot T \cdot m \cdot n)$, which reflects how the expected execution time increases with the size of the problem instances.

Table 17. Computational complexity of the BAOA per SCP instance set. The general complexity is $O(T \cdot P \cdot m \cdot n)$, with $T = 100$ iterations and $P = 200$ individuals.

Instance Set	m	n	Big-O	Complexity (Numerical)
scp41	200	1000	$O(mn)$	4.0×10^9
scp51	200	2000	$O(mn)$	8.0×10^9
scp61	200	1000	$O(mn)$	4.0×10^9
scpa1	300	3000	$O(mn)$	1.8×10^{10}
scpb1	300	3000	$O(mn)$	1.8×10^{10}
scpc1	400	4000	$O(mn)$	3.2×10^{10}
scpd1	400	4000	$O(mn)$	3.2×10^{10}

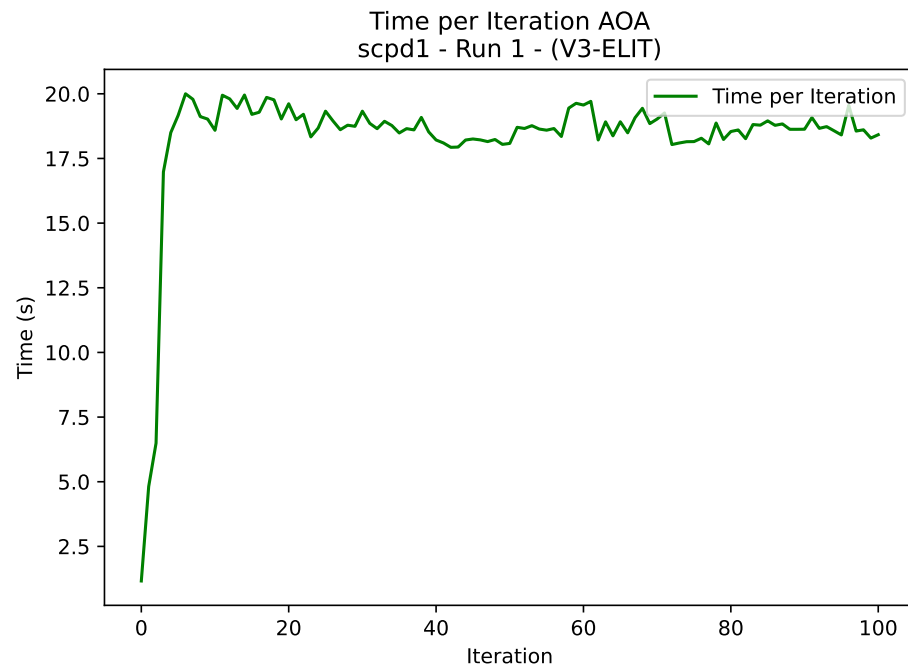


Figure 11. Iteration time per iteration for instance SCPd1 under the AOA V3-ELIT configuration. The curve shows two phases: an initial overhead where computation time increases sharply, followed by a stabilization phase where the iteration time remains nearly constant around 18–19 s.

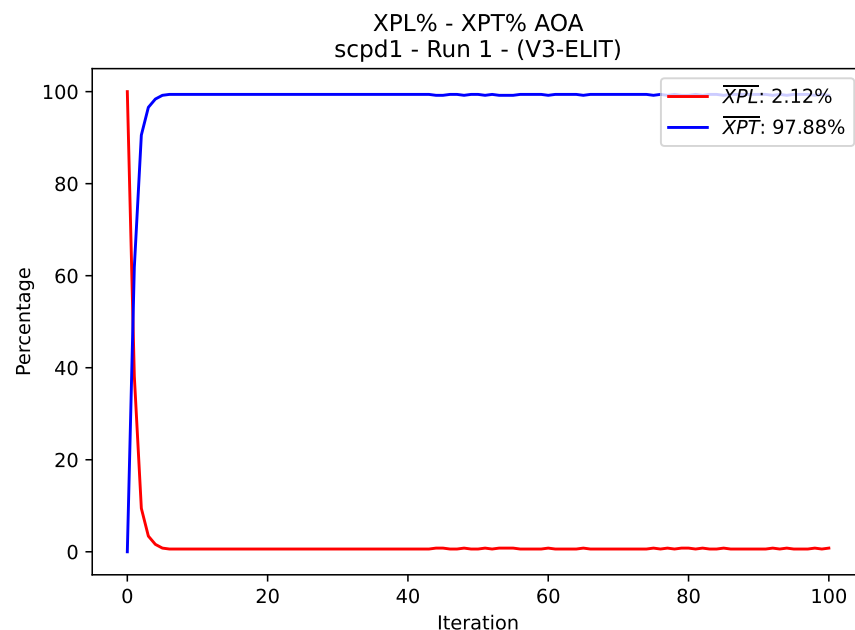


Figure 12. Exploration (XPL%) and exploitation (XPT%) dynamics for BAOA V3-ELIT in the scp1 instance. The curves show a sharp decline in exploration during the first iterations, stabilizing below 2.12%, while exploitation dominates above 97.88%.

8. Novelty and Contributions

In this paper, we have presented the first binary adaptation of the AOA to solve an important industry problem. In addition, we have introduced a new index that is complementary to the RPD for measuring stability across runs to quantify solution stability; Figure 2 shows this information. The joint analysis of RPD versus CV shows that 35 instances fall into the high quality zone and 27 are distributed across moderate zones, with

only three instances falling into the low quality zone. These results reinforce the robustness of our BAOA in terms of solution quality and stability (see Figure 13).

Following the statistical methodology (normality $\rightarrow$ nonparametric test for paired samples), the applied tests show that the proposed version is competitive with state-of-the-art techniques (SCA, PSA, GWO, BGO), with no statistically significant unfavorable differences.

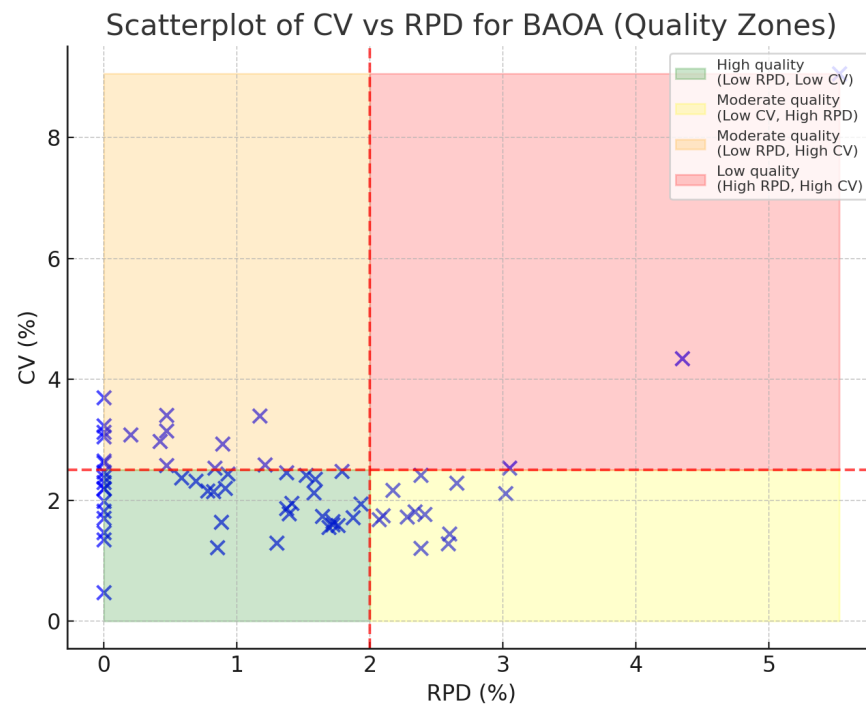


Figure 13. Summary of the high-quality zone from the RPD vs. CV analysis.

In terms of computational efficiency, the results show that our BAOA achieves high-quality solutions with reduced execution times. Table 6 reports total times in the range of 3 s to 19 s per iteration, depending on the instance (e.g., ~ 3.3 s on scp51 and ~ 18 – 19 s on scp1). Furthermore, the dynamic analysis of Figures 8 and 9 reveals stable behavior; after an initial overload phase, the time per iteration remains virtually constant throughout the execution.

9. Conclusions and Future Work

In this work, we have presented a new version for solving the SCP based on the Arithmetic Optimization Algorithm metaheuristic and coverage problems in general. Our design preserves the core mechanisms of the original algorithm while introducing a two-step binarization scheme (transfer functions and binarization rules) and a conventional greedy repair operator to ensure feasibility in discrete spaces. We propose the Coefficient of Variation (CV) as an index of stability across runs, enabling comparisons between configurations with similar averages, which complements the RPD index and allows for a precise evaluation of the solution's performance across quality zones formed by graphically plotting RPD versus CV. Our BAOA exhibits competitive performance across 45 benchmark instances from the OR-Library when compared to state-of-the-art metaheuristics SCA, PSA, GWO, and BGO, achieving a minimum average RPD (1.51%) and a high number of optimal solutions (21 out of 25 in the subset where the optima are known). After using multiple statistical analyses (the Shapiro-Wilk and KS-Lilliefors tests as well as the unpaired Wilcoxon signed-rank test), no statistically significant differences were revealed, with all results yielding $p > 0.05$. Therefore, there are no significant differences between our BAOA and the aforementioned

metaheuristics. While the results of these tests and descriptive comparisons were good, additional improvements can still be applied. Future research could explore alternative combinations of transfer functions and binarization rules, principle-based hybridization, adaptive parameter control, lightweight intensification strategies, and population diversity mechanisms to better balance exploration and exploitation. These guidelines could further improve the quality and efficiency of the solution.

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

Funding: Felipe Cisternas-Caneo is supported by National Agency for Research and Development (ANID)/Scholarship Program/DOCTORADO NACIONAL/2023-21230203.

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

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

Abbreviations

The following abbreviations are used in this manuscript:

AOA	Arithmetic Optimization Algorithm
BAOA	Binary Arithmetic Optimization Algorithm
SCP	Set Covering Problem
MOA	Mathematical Optimizer Accelerated
RPD	Relative Percentage Deviation
SCA	Sine Cosine Algorithm
PSA	Pendulum Search Algorithm
GWO	Grey Wolf Optimizer
BGO	Binary Growth Optimizer
ACO	Ant Colony Optimization
MOP	Mathematical Optimizer Probability

References

1. Karp, R.M. Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*; Miller, R.E., Thatcher, J.W., Eds.; Springer: Berlin/Heidelberg, Germany, 1972; pp. 85–103. [\[CrossRef\]](#)
2. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W. H. Freeman: New York, NY, USA, 1979. [\[CrossRef\]](#)
3. Suh, W.H.; Oh, S.; Ahn, C.W. Metaheuristic-based time series clustering for anomaly detection in manufacturing industry. *Appl. Intell.* **2023**, *53*, 21723–21742. [\[CrossRef\]](#)
4. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [\[CrossRef\]](#)
5. 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. [\[CrossRef\]](#)
6. Wen, X.; Chung, S.H.; Ma, H.L.; Khan, W.A. Airline crew scheduling with sustainability enhancement by data analytics under circular economy. *Ann. Oper. Res.* **2023**, *342*, 959–985. [\[CrossRef\]](#)
7. Beasley, J.E. OR-library: Distributing test problems by electronic mail. *J. Oper. Res. Soc.* **1990**, *41*, 1069–1072. [\[CrossRef\]](#)
8. Blum, C.; Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv. (CSUR)* **2003**, *35*, 268–308. [\[CrossRef\]](#)
9. Lanza-Gutiérrez, J.M.; Caballé, N.C.; Crawford, B.; Soto, R.; Gómez-Púlido, J.A.; Paredes, F. Exploring Further Advantages in an Alternative Formulation for the Set Covering Problem. *Math. Probl. Eng.* **2020**, *2020*, 5473501. [\[CrossRef\]](#)

10. Carrabs, F.; Cerulli, R.; Mansini, R.; Moreschini, L.; Serra, D. Solving the Set Covering Problem with Conflicts on Sets: A new parallel GRASP. *Comput. Oper. Res.* **2024**, *166*, 106620. [[CrossRef](#)]
11. Al-Himyari, B.; Al-khafaji, H. Exploration-Exploitation Tradeoffs in Metaheuristics: A Review. *Asian J. Appl. Sci.* **2024**, *12*, 1–27. [[CrossRef](#)]
12. Saremi, S.; Mirjalili, S.; Lewis, A. How important is a transfer function in discrete heuristic algorithms. *Neural Comput. Appl.* **2014**, *26*, 625–640. [[CrossRef](#)]
13. Lanza-Gutiérrez, 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](#)]
14. Arora, S.; Barak, B. *Computational Complexity: A Modern Approach*; Cambridge University Press: New York, NY, USA, 2009. [[CrossRef](#)]
15. Ospina, R.; Marmolejo-Ramos, F.; Medina, M. Performance of Some Estimators of Relative Variability. *Front. Appl. Math. Stat.* **2019**, *5*, 43. [[CrossRef](#)]
16. Cook, W.J.; Applegate, D.; Bixby, R.E.; Bixby, R.E.; Chvátal, V. *The Traveling Salesman Problem: A Computational Study*; Princeton University Press: Princeton, NJ, USA, 2007. [[CrossRef](#)]
17. Hazra, A. Using the confidence interval confidently. *J. Thorac. Dis.* **2017**, *9*, 4125–4130. [[CrossRef](#)] [[PubMed](#)]
18. Leiva, D.; Ramos-Tapia, B.; Crawford, B.; Soto, R.; Cisternas-Caneo, F. A Novel Approach to Combinatorial Problems: Binary Growth Optimizer Algorithm. *Biomimetics* **2024**, *9*, 283. [[CrossRef](#)]
19. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [[CrossRef](#)]
20. Ab. Aziz, N.A.; Ab. Aziz, K.A. Pendulum Search Algorithm: An Optimization Algorithm Based on Simple Harmonic Motion and Its Application for a Vaccine Distribution Problem. *Algorithms* **2022**, *15*, 214. [[CrossRef](#)]
21. Martín-Santamaría, R.; López-Ibáñez, M.; Stützle, T.; Colmenar, J.M. On the automatic generation of metaheuristic algorithms for combinatorial optimization problems. *Eur. J. Oper. Res.* **2024**, *318*, 740–751. [[CrossRef](#)]
22. Yap, B.W.; Sim, C.H. Comparisons of various types of normality tests. *J. Stat. Comput. Simul.* **2011**, *81*, 2141–2155. [[CrossRef](#)]
23. Shapiro, S.S.; Wilk, M.B. An analysis of variance test for normality (complete samples). *Biometrika* **1965**, *52*, 591–611. [[CrossRef](#)]
24. Lilliefors, H.W. On the Kolmogorov-Smirnov test for normality with mean and variance unknown. *J. Am. Stat. Assoc.* **1967**, *62*, 399–402. [[CrossRef](#)]
25. Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biom. Bull.* **1945**, *1*, 80–83. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.