International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V72I4P132

Volume 72 Issue 4, 315-323, April 2024
© 2024 Seventh Sense Research Group®

Original Article

Three Binary Versions of Tree-Seed Algorithm for Binary Optimization

Yahye Abukar Ahmed[1], Mustafa Servet Kiran[2*], Mohamed Omar Abdullahi[1], Abdukadir Dahir Jimale[1], Abdulaziz Yasin Nageye[1], Ali Abdi Jama[1]

[1]*Faculty of Computing, SIMAD University, Mogadishu, Somalia.*
[2]*Department of Computer Engineering, Faculty of Engineering, Selcuk University, Konya- Turkey*

[*]*Corresponding Author : mskiran@selcuk.edu.tr*

Received: 25 October 2023 Revised: 09 February 2024 Accepted: 01 April 2024 Published: 24 April 2024

Abstract *- This study explores the adaptation of the Tree-Seed Algorithm (TSA), a population-based optimization method, for solving binary optimization problems. Initially designed for problems featuring continuous solution spaces, TSA is adjusted to address binary-structured optimization challenges. Three distinct approaches, namely the sigmoid function, modulo function, and xor logic gate, are employed to address TSA for binary optimization problem-solving. The efficacy of these methods is evaluated through experimentation on Uncapacitated Facility Location Problems (UFLPs), representative pure binary problems from existing literature. A comprehensive analysis is conducted using a selection of well-known small, medium, and large-sized UFLP instances to assess the performances and the impact of TSA's control parameters. Comparative analysis of obtained results reveals promising outcomes achieved by the proposed algorithm.*

Keywords *- Sigmoid function, UFLP, Modulo function, Binary optimization, Tree-seed algorithm, x-or logic gate.*

1. Introduction

Recently, numerous optimization algorithms, such as populations, have emerged to address optimization problems with diverse characteristics. Among these algorithms, the Tree-Seed Algorithm (TSA) has attracted attention for its effectiveness in resolving such optimization challenges. TSA is characterized as an algorithm that operates through iterative processes and is suggested for addressing unconstrained continuous optimization challenges. In the basic variant of TSA, the key control parameters are analysed, and its efficacy was compared with the firefly algorithm [4], artificial bee colony (ABC) [2] [3], particle swarm optimization (PSO) [1], and bat algorithm [5]. In another study, the TSA was applied to address the pressure vessel design problem, which is a constrained optimization problem. However, despite TSA's demonstrated ability to handle constrained or unconstrained optimization problems, a notable research gap exists in its adaptation for solving binary optimization problems. Such problems possess a binary-structured solution space, where decision variables are constrained to binary values of 0 or 1. While several population-based heuristic methods have been modified for binary optimization — employing approaches like the sigmoid function in binary PSO [6], modulo operations in PSO [7] and ABC [2] [3], and XOR-based logic gates in ABC [2] [3] — the exploration of TSA variants for binary optimization remains relatively unexplored.

This gap in the literature highlights the need for novel approaches to adapt TSA specifically for binary optimization problems, addressing challenges such as solution representation, exploration-exploitation balance, and convergence dynamics unique to this problem domain. By developing and evaluating modified versions of TSA tailored for binary optimization, authors aim to fill this research gap and contribute to the advancement of evolutionary computation techniques in solving binary optimization problems. This study aims to bridge this gap by proposing three modified versions of TSA tailored specifically for binary optimization problems. Our focus is on addressing the Uncapacitated Facility Location Problem (UFLP), a classical binary optimization problem with a benchmark dataset available in the literature, thereby facilitating the performance evaluation of our proposed TSA variants. The paper is structured as follows: Section, I providing an overview of the research objectives; a brief review of the literature on binary optimization and the mathematical model of the problem. Section II demonstrates the Basic and Modified Versions of TSA. Section III elaborates on the proposed binary version of the Tree Seed Algorithm (TSA). Section IV details the experimental setup such as parameter settings and evaluation criteria. Section V provides a comprehensive discussion of the results obtained from the experiments.

1.1. Related Work

This section illustrates the literature and the related work of the study. The population-based swarm intelligence methods have been developed with inspiration from natural phenomena. Tree-Seed Algorithm is also invented by the trees and their seeds in nature, and the tree collection is named stand instead of population or swarm, but the working of the stand is similar to the working of the swarm. The aim of the stand is to maintain and sustain the existence of the trees by producing seeds from the trees. Swarm intelligence refers to the field of study focused on collective behaviours emerging from the local interactions among many individuals who coordinate their activities through decentralized systems [8-10].

In the literature, the most prominent examples of swarm intelligence algorithms include particle swarm optimization (PSO), Ant Colony Optimization (ACO), and the Artificial Bee Colony (ABC) algorithm. PSO, a stochastic, population-based technique conceived by Dr. Eberhart and Dr. Kennedy in 1995, draws inspiration from natural phenomena such as bird flocking or fish schooling [1]. Ant colony optimization, introduced by M. Dorigo and collaborators in the early 1990s, represents a pioneering nature-inspired metaheuristic approach for addressing Combinatorial Optimization (CO) challenges [11].

ACO achieves finding an optimal solution for the search problems through the inspiring source of ants' foraging behavior. ABC has been suggested to solve continuous optimization problems by inspiring waggle dance and food search behaviors of honey bees [2], [3]. One of the recently introduced algorithms is the Tree-Seed Algorithm (TSA), which draws inspiration from natural processes such as the production of trees and seeds. This algorithm was developed in 2015. TSA addresses the interaction between trees and their seeds as they grow in a specific area of land.

According to nature way, trees grow and spread on the ground over their seeds. These seeds are cultivated over time and generate new trees. The trees and seeds' location is regarded as an n-dimensional solution space corresponding with the possible solution of an optimization problem. The search starts with sowing the trees into the ground. During the iterations, the number of seeds of each tree is produced. TSA was successfully applied to solve constrained and unconstrained optimization problems and multilevel thresholding problems.

Many swarm intelligence methodologies have emerged in recent years as solutions for tackling Operational Research (OR) dilemmas, including the Uncapacitated Facility Location Problem (UFLP), which stands out as one of the most intensively researched discrete location predicaments. Concerned with the optimal placement of an undetermined number of facilities to minimize transportation costs [7], [12], [13].

1.2. Mathematical Model of UFLP

The Uncapacitated Facility Location Problem (UFLP) entails a scenario whereby a set of customer locations, identified as m, needs to be served by a selection of potential facilities n. Each facility incurs a fixed cost fcj, and there exists a transportation cost cij associated with serving customer I from facility j. Notably, there are no capacity constraints imposed on any candidate facility, and each customer's entire demand must be assigned to a single facility [14]. The objective of this problem is to determine the optimal number of facilities to be established and to identify those facilities in order to minimize the total cost. The mathematical representation of this problem can be articulated as follows:

$$Z = min\left(\sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}.x_{i,j} + \sum_{j=1}^{n} fc_j.y_j\right) \tag{1}$$

subject to:

$$\sum_{j=1}^{n} x_{i,j} = 1 \qquad \forall i\ in\ m \tag{2}$$

$$0 \le x_{i,j} \le 1 \tag{3}$$

In the provided context, where i = 1...m and j = 1, ...,n, x_{ij} represents the quantity supplied from facility I to customer j, and y_j indicates whether facility j is established ($y_j = 1$) or not ($y_j = 0$). Equation 2 ensures that all customer demands are fulfilled by at least one operational facility, while Equation 3 maintains the integrity of the solution. After implementing the UFL problem into specific algorithms, researchers use the ORLIB dataset, which is one of the most widely used UFLP benchmark instances, to evaluate the performance and accuracy of their algorithms. ORLIB instances are part of OR-Library introduced by J.E Beasley in 1996 for a variety of Operations Research (OR) problems [15]. This test suite comprises three sets of benchmark problems m×n, categorized by their dimensions, where m represents the number of customers and n denotes the number of facilities. Specifically, there are four small-sized problems labeled as Cap71–74 and four medium-sized problems labeled as Cap101–104. The remaining four problems, denoted as Cap131–134, are categorized as large-sized in this investigation. This problem set is selected to investigate the performance of binary versions of TSAs because Cap problems are pure binary problems, and there are no floating decision variables in the problem model. The problem is also a well-known problem, and the effectiveness of many swarms or evolutionary computation methods are analyzed in solving these problems.

2. The Foundational Version of the Tree-Seed Algorithm

In the fundamental version of TSA, trees and their seeds symbolize potential solutions to the optimization problem. After control parameters (population size, upper and lower bound for the various seeds and search tendency – ST) are set, the trees are scattered to the search space of the problem by using the equation given as follows:

$$T_{i,j} = L_{j,min} + r_{i,j}\left(H_{j,max} - L_{j,min}\right)\ ,\ \ j = 1,2,\dots,D\ and\ i = 1,2,\dots,N \tag{1}$$

if (rnd<ST)

$$S_{k,j} = T_{i,j} + \alpha_{i,j} \times (B_j - T_{r,j})$$

else

$$S_{k,j} = T_{i,j} + \alpha_{i,j} \times (T_{i,j} - T_{r,j})$$

Fig. 1 The solution update procedure of TSA using the ST parameter

Where, $T_{i,j}$ represents the jth dimension of ith tree, $L_{j,\min}$ is the lower bound of search space, $H_{j,\max}$ indicates the upper bound of search space, $r_{i,j}$ is a random number generated in a variety of situations, such as [0,1]. N represents the number of trees, and D represents the dimensionality of the problem. The trees dispersed to search space are called as stand. The performance of the trees within the stand is assessed by employing a specialized objective function tailored to the optimization problem, and the best tree within the stand is selected given as follows:

$$B = argmin\left\{f\left(\vec{T_i}\right)\right\} \qquad i = 1,2, \ldots ,N \tag{2}$$

Where B represents the best tree within the stand, the best tree is picked from the stand because there are two solution update mechanisms in TSA. The selection of this equation is performed by using the ST parameter of TSA. While creating seeds for each tree, either the best tree and a neighbor tree or parent tree and a neighbor tree is used in the TSA algorithm. The quantity of seeds falls within a predetermined range set for the algorithm in the initial study. In the previous study, this range is suggested as 10% to 25% of the number of trees within the stand. For instance, if the stand comprises 10 trees, each tree generates a minimum of 1 and a maximum of 3 seeds, with floating numbers rounded up. Following the explanations provided, the seeds are generated using the procedure outlined in Figure 1, and the operational diagram of TSA is depicted in Figure 2.

In Figure 1., $T_{r,j}$ represents the jth dimension of the rth tree randomly chosen from the stand, $S_{k,j}$ indicates the jth dimension of K_{th} seed generated for ith tree, $\alpha_{i,j}$ denotes the scaling factor randomly obtained from a variety of situations such as [0,1], ST is selected within the interval of [0,1], and rnd is randomly produced within the range of [0,1]. If ST is selected as 0, more global search capability for TSA is obtained. If it is selected as 1, the convergence speed of TSA is improved, but effectively searching the solution space is reduced in TSA. Therefore, this control parameter is adjusted and selected depending on the problem characteristics.

3. Proposed Binary Versions of TSA

The TSA algorithm is proposed to solve continuous optimization problems, as seen from the explanations and equations in Section 2. If an optimization problem whose solution space is binary-structured is tried to solve by TSA, it should be adapted to address this category of problems. In this research, three approaches are used to solve this type of problem. Two of them use a conversion (Sigmoid and mod function). In these methods, the trees are distributed across the continuous solution space, as depicted in Figure 1, through the seed production mechanism as works with continuous values. However, before the objective function is evaluated, the continuous values represented by the trees or seeds are converted to binary values. The third approach uses the exclusive or – XOR – logic operator to obtain candidate solutions (seeds) for the trees. This method works on binary structured solution space, and it does not use any conversion.

3.1. Sigmoid function-based binary TSA

The sigmoid function, as defined in Equation 3, is employed to derive probability values for conversion.

$$P_{i,j} = \frac{1}{1+e^{-T_{i,j}}} \tag{3}$$

When P_{ij} exceeds 0.5, a temporary array of decision variables is instantiated, with its jth dimension being initialized to 0 before the evaluation of the objective function. These procedures for each tree in the initialization and seeds during the iteration are performed to adapt TSA to binary optimization.

3.2. Modulo function-based binary TSA

Similar to the application of the sigmoid function in TSA, the conversion process, represented in Equation 4, utilizes modulo operation with base 2 to transform continuous solutions into their binary counterparts.

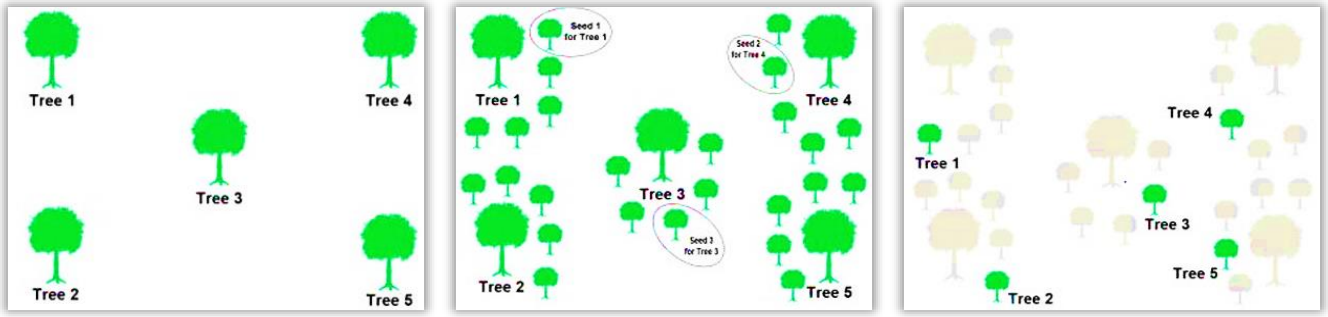

Fig. 2 The solution of the working diagram of TSA [16]

$$BS_{i,j} = mod(abs(\lfloor T_{i,j} \rfloor), 2) \quad (4)$$

In this context, BS_{ij} represents the binary solution acquired for T_{ij} through a rounding operation downwards, where abs denote the absolute function. The fitness of T_{ij} is determined by evaluating the objective function pertinent to the binary optimization problem, utilizing the BS_{ij} binary array of decision variables.

3.3. Xor-based binary TSA

Exclusive or –xor logic operator accepts two inputs, which are binary, and if they are the same, 0 output is produced. Otherwise, it produces 1 as output. To use this operation in TSA to solve binary problems, trees in the stand are scattered to binary solution space and seeds are produced for each tree given as follows:

$$S_{k,j} = \begin{cases} xor(B_j, T_{r,j}), \ ST < rnd \\ xor(T_{i,j}, T_{r,j}), \ otherwise \end{cases} \quad (5)$$

The difference between the xor-based approach from the other two approaches is that in xor-based TSA, the operation is conducted within the binary solution space, where trees and seeds symbolize binary values rather than continuous values.

4. Experimental Setup

To assess the performance and accuracy of the three distinct versions of TSA, the authors employed the uncapacitated facility location test suite consisting of 12 test problems sourced from the OR-Library [15]. Within the test suite, four problems (Cap71-74) are categorized as small-sized, while eight problems (Cap101-104 and Cap131-134) are medium-sized. The sizes and the costs of the optimal solutions are outlined in Table 1. To ensure consistency in comparison, a population size of 40 is adopted across all experiments.

The termination criterion for each experiment is defined as the maximum number of function evaluations (Max_FEs), set at 80,000. Different binary functions and varying search tendencies (STs) are employed as control parameters of TSA in each experimental study. These experiments are conducted 30 times to address the UFLP.

The results, including best, worst values, and mean, along with standard deviations created from these runs, are documented in Tables 2-10. The accuracy and robustness of the proposed TSA variation are evaluated based on the mean and standard deviations, respectively, enabling a comparative analysis.

Table 1. Outlines the description of the test suite

Problem name	Problem size	Cost of the optimal solution
Cap71	16x50	932,615.75
Cap72	16x50	977,799.40
Cap73	16x50	1,010,641.45
Cap74	16x50	1,034,976.98
Cap101	25x50	796,648.44
Cap102	25x50	854,704.20
Cap103	25x50	893,782.11
Cap104	25x50	928,941.75
Cap131	50x50	793,439.56
Cap132	50x50	851,495.33
Cap133	50x50	893,076.71
Cap134	50x50	928,941.75

Table 2. The test results of small-sized problems for ST=0.1.

1-70		Mean	Std.Dev.	Best	Worst
Cap 71	TSA-Mode	9.33E+05	4.36E+02	9.33E+05	9.34E+05
	TSA-XOR	9.37E+05	2.59E+03	9.33E+05	9.42E+05
	TSA-Sig	9.36E+05	4.74E-10	9.36E+05	9.36E+05
Cap 72	TSA-Mode	9.78E+05	4.13E+02	9.78E+05	9.79E+05
	TSA-XOR	9.84E+05	1.58E+03	9.80E+05	9.87E+05
	TSA-Sig	9.82E+05	4.74E-10	9.82E+05	9.82E+05
Cap 73	TSA-Mode	1.01E+06	4.74E-10	1.01E+06	1.01E+06
	TSA-XOR	1.01E+06	2.38E+03	1.01E+06	1.02E+06
	TSA-Sig	1.01E+06	4.74E-10	1.01E+06	1.01E+06
Cap 74	TSA-Mode	1.03E+06	3.55E-10	1.03E+06	1.03E+06
	TSA-XOR	1.05E+06	6.46E+03	1.03E+06	1.06E+06
	TSA-Sig	1.04E+06	4.74E-10	1.04E+06	1.04E+06

Table 3. The test results of small-sized problems for ST=0.2

2-70		Mean	Std.Dev.	Best	Worst
Cap 71	TSA-Mode	9.33E+05	5.37E+02	9.33E+05	9.34E+05
	TSA-XOR	9.38E+05	1.66E+03	9.34E+05	9.42E+05
	TSA-Sig	9.36E+05	4.74E-10	9.36E+05	9.36E+05
Cap 72	TSA-Mode	9.78E+05	1.97E+02	9.78E+05	9.79E+05
	TSA-XOR	9.82E+05	2.10E+03	9.79E+05	9.86E+05
	TSA-Sig	9.82E+05	4.74E-10	9.82E+05	9.82E+05
Cap 73	TSA-Mode	1.01E+06	4.74E-10	1.01E+06	1.01E+06
	TSA-XOR	1.01E+06	2.16E+03	1.01E+06	1.02E+06
	TSA-Sig	1.01E+06	4.74E-10	1.01E+06	1.01E+06
Cap 74	TSA-Mode	1.03E+06	3.55E-10	1.03E+06	1.03E+06
	TSA-XOR	1.05E+06	5.60E+03	1.03E+06	1.06E+06
	TSA-Sig	1.04E+06	4.74E-10	1.04E+06	1.04E+06

Table 4. The test results of small-sized problems for ST=0.3.

3-70		Mean	Std.Dev.	Best	Worst
Cap 71	TSA-Mode	9.33E+05	5.31E+02	9.33E+05	9.34E+05
	TSA-XOR	9.37E+05	2.08E+03	9.34E+05	9.40E+05
	TSA-Sig	9.36E+05	4.74E-10	9.36E+05	9.36E+05
Cap 72	TSA-Mode	9.78E+05	3.76E+02	9.78E+05	9.79E+05
	TSA-XOR	9.82E+05	1.63E+03	9.79E+05	9.85E+05
	TSA-Sig	9.82E+05	4.74E-10	9.82E+05	9.82E+05
Cap 73	TSA-Mode	1.01E+06	4.74E-10	1.01E+06	1.01E+06
	TSA-XOR	1.01E+06	2.25E+03	1.01E+06	1.02E+06
	TSA-Sig	1.01E+06	4.74E-10	1.01E+06	1.01E+06
Cap 74	TSA-Mode	1.03E+06	3.55E-10	1.03E+06	1.03E+06
	TSA-XOR	1.05E+06	4.99E+03	1.03E+06	1.06E+06
	TSA-Sig	1.04E+06	4.74E-10	1.04E+06	1.04E+06

Table 5. The test results of medium-sized problems for ST=0.1.

1-100		Mean	Std.Dev.	Best	Worst
Cap 101	TSA-Mode	8.02E+05	1.20E+03	7.99E+05	8.04E+05
	TSA-XOR	8.09E+05	2.24E+03	8.01E+05	8.13E+05
	TSA-Sig	8.07E+05	1.91E+03	8.03E+05	8.08E+05
Cap 102	TSA-Mode	8.58E+05	1.11E+03	8.55E+05	8.59E+05
	TSA-XOR	8.70E+05	3.54E+03	8.61E+05	8.76E+05
	TSA-Sig	8.61E+05	1.88E+03	8.58E+05	8.65E+05
Cap 103	TSA-Mode	8.95E+05	1.18E+03	8.94E+05	8.98E+05
	TSA-XOR	9.16E+05	4.66E+03	9.05E+05	9.23E+05
	TSA-Sig	8.99E+05	2.10E+03	8.96E+05	9.02E+05
Cap 104	TSA-Mode	9.31E+05	2.59E+03	9.29E+05	9.37E+05
	TSA-XOR	9.75E+05	8.64E+03	9.57E+05	9.91E+05
	TSA-Sig	9.42E+05	2.23E+03	9.38E+05	9.45E+05

Table 6. The test results of medium-sized problems for ST=0.2.

2-100		Mean	Std.Dev.	Best	Worst
Cap 101	TSA-Mode	8.01E+05	1.50E+03	7.98E+05	8.04E+05
	TSA-XOR	8.09E+05	2.07E+03	8.05E+05	8.12E+05
	TSA-Sig	8.06E+05	2.32E+03	8.03E+05	8.09E+05
Cap 102	TSA-Mode	8.58E+05	1.49E+03	8.55E+05	8.61E+05
	TSA-XOR	8.69E+05	4.51E+03	8.59E+05	8.74E+05
	TSA-Sig	8.61E+05	3.18E+03	8.56E+05	8.65E+05
Cap 103	TSA-Mode	8.96E+05	1.54E+03	8.94E+05	8.99E+05
	TSA-XOR	9.16E+05	5.76E+03	9.00E+05	9.25E+05
	TSA-Sig	9.00E+05	1.68E+03	8.98E+05	9.03E+05
Cap 104	TSA-Mode	9.31E+05	2.57E+03	9.29E+05	9.36E+05
	TSA-XOR	9.71E+05	1.14E+04	9.38E+05	9.88E+05
	TSA-Sig	9.41E+05	4.97E+03	9.33E+05	9.48E+05

Table 7. The test results of medium-sized problems for ST=0.3.

3-100		Mean	Std.Dev.	Best	Worst
Cap 101	TSA-Mode	8.02E+05	1.61E+03	7.98E+05	8.05E+05
	TSA-XOR	8.08E+05	2.25E+03	8.02E+05	8.11E+05
	TSA-Sig	8.06E+05	9.41E+02	8.05E+05	8.08E+05
Cap 102	TSA-Mode	8.58E+05	1.32E+03	8.55E+05	8.61E+05
	TSA-XOR	8.69E+05	3.13E+03	8.62E+05	8.75E+05
	TSA-Sig	8.63E+05	9.99E+02	8.61E+05	8.64E+05
Cap 103	TSA-Mode	8.96E+05	1.44E+03	8.94E+05	8.98E+05
	TSA-XOR	9.16E+05	4.83E+03	9.05E+05	9.24E+05
	TSA-Sig	9.01E+05	1.17E+03	8.99E+05	9.03E+05
Cap 104	TSA-Mode	9.31E+05	2.45E+03	9.29E+05	9.37E+05
	TSA-XOR	9.70E+05	6.53E+03	9.55E+05	9.82E+05
	TSA-Sig	9.40E+05	2.92E+03	9.35E+05	9.44E+05

Table 8. The test results of large-sized problems for ST=0.1.

1-130		Mean	Std.Dev.	Best	Worst
Cap 131	TSA-Mode	8.18E+05	3.49E+03	8.09E+05	8.23E+05
	TSA-XOR	8.47E+05	4.05E+03	8.38E+05	8.52E+05
	TSA-Sig	8.26E+05	3.29E+03	8.21E+05	8.32E+05
Cap 132	TSA-Mode	8.83E+05	5.47E+03	8.63E+05	8.90E+05
	TSA-XOR	9.38E+05	7.20E+03	9.18E+05	9.49E+05
	TSA-Sig	8.95E+05	6.20E+03	8.86E+05	9.02E+05
Cap 133	TSA-Mode	9.28E+05	8.42E+03	9.07E+05	9.42E+05
	TSA-XOR	1.02E+06	1.02E+04	1.00E+06	1.04E+06
	TSA-Sig	9.15E+05	9.31E+03	9.06E+05	9.31E+05
Cap 134	TSA-Mode	9.88E+05	9.89E+03	9.69E+05	1.00E+06
	TSA-XOR	1.14E+06	2.03E+04	1.09E+06	1.17E+06
	TSA-Sig	9.64E+05	1.27E+04	9.49E+05	9.81E+05

Table 9. The test results of large-sized problems for ST=0.2.

2-130		Mean	Std.Dev.	Best	Worst
Cap 131	TSA-Mode	8.19E+05	3.86E+03	8.10E+05	8.25E+05
	TSA-XOR	8.47E+05	4.61E+03	8.31E+05	8.53E+05
	TSA-Sig	8.28E+05	2.03E+03	8.25E+05	8.34E+05
Cap 132	TSA-Mode	8.80E+05	5.20E+03	8.63E+05	8.87E+05
	TSA-XOR	9.40E+05	6.01E+03	9.25E+05	9.49E+05
	TSA-Sig	8.97E+05	5.60E+03	8.88E+05	9.01E+05
Cap 133	TSA-Mode	9.26E+05	7.30E+03	9.11E+05	9.35E+05
	TSA-XOR	1.02E+06	1.10E+04	9.98E+05	1.04E+06
	TSA-Sig	9.23E+05	6.24E+03	9.12E+05	9.31E+05
Cap 134	TSA-Mode	9.87E+05	1.07E+04	9.46E+05	1.00E+06
	TSA-XOR	1.13E+06	1.67E+04	1.10E+06	1.16E+06
	TSA-Sig	9.65E+05	1.03E+04	9.52E+05	9.86E+05

Table 10. The test results of large-sized problems for ST=0.3.

3-130		Mean	Std.Dev.	Best	Worst
Cap 131	TSA-Mode	8.17E+05	3.54E+03	8.09E+05	8.22E+05
	TSA-XOR	8.44E+05	5.52E+03	8.23E+05	8.52E+05
	TSA-Sig	8.24E+05	2.84E+03	8.21E+05	8.31E+05
Cap 132	TSA-Mode	8.80E+05	4.77E+03	8.69E+05	8.88E+05
	TSA-XOR	9.33E+05	1.06E+04	9.09E+05	9.50E+05
	TSA-Sig	8.97E+05	3.26E+03	8.93E+05	9.02E+05
Cap 133	TSA-Mode	9.27E+05	7.74E+03	9.06E+05	9.40E+05
	TSA-XOR	1.02E+06	1.16E+04	9.93E+05	1.04E+06
	TSA-Sig	9.49E+05	1.27E+04	9.27E+05	9.66E+05
Cap 134	TSA-Mode	9.84E+05	9.54E+03	9.63E+05	1.00E+06
	TSA-XOR	1.13E+06	2.04E+04	1.09E+06	1.16E+06
	TSA-Sig	1.03E+06	9.69E+03	1.02E+06	1.04E+06

In the comparison tables (Tables 2-10), the minimum mean, best, worst values and standard deviation values are given in bold font. The results obtained by Cap71, Cap72, Cap73 and Cap74 problems are presented in Tables 2, 3 and 4 for ST=0.1, 0.2 and 0.3, respectively. As seen from Tables 2-4, when the mode function-based TSA (TSA-Mode) is compared with other binary versions, this version demonstrates significant improvement in both solution quality and robustness. In general, TSA-Mode reaches the optimal solutions to the corresponding problems. Based on the standard deviations, the robustness of the TSA-Sig algorithm is better than that of the TSA-XOR for all of the test problems. When ST values are set as 0.1, 0.2 and 0.3, the similar results are obtained. Hence, for this particular problem group (Cap70s), varying the ST values does not affect the effectiveness of the algorithm. Table 3 displays the test results for small-sized problems with ST=0.2, evaluating three techniques: TSA-Mode, TSA-XOR, and TSA-Sig.

It presents mean, standard deviation, best, and worst objective function values for each technique across problem instances (Cap71 to Cap74). The data reveals variations in performance among techniques, with TSA-Mode generally exhibiting slightly worse performance. Additionally, the best and worst objective function values illustrate the range of performance achieved by each technique, with TSA-Sig generally outperforming TSA-Mode and TSA-XOR in terms of mean and best values. The experimental results obtained for Cap101, Cap102, Cap103 and Cap104 problems are respectively given in Tables 5, 6 and 7 for ST=0.1, 0.2 and 0.3. As seen from Tables 5-7, in all the test problems, the TSA-Mode binary version is better for both solution quality and robustness than the others. However, TSA-Mode could not find the optimal solutions to any problem, although it is closer to the optimal values. The robustness of the TSA-Sig algorithm is similar to that of the TSA-XOR for all the test problems. TSA-XOR has the worst performance with regard to mean and standard deviation values when compared with all the binary versions. When ST values are set from 0.1 to 0.3 for solving UFLP, the obtained results are closer to each other.

The experimental results obtained by Cap131, Cap132, Cap133 and Cap134 problems are respectively given in Tables 8, 9 and 10 for ST=0.1, 0.2 and 0.3. As seen from Tables 8-10, the TSA-Mode binary version is generally better for both solution quality and robustness than the others. However, TSA-Sig is better than TSA-Mode for Cap 134 problem under ST=0.1 and 0.2. For all of the Cap 130 problems, any binary versions of TSA could not find the optimal solutions. The solution quality of the TSA-Sig algorithm is better than that of the TSA-XOR for all the test problems. In addition, TSA-XOR has the worst performance among other binary versions, according to Tables 8-10. When ST values are assigned to 0.1, 0.2 and 0.3 for solving UFLP, the obtained results are closer to each other. Table 8 presents test results for solving large-sized problems using three different techniques (TSA-Mode, TSA-XOR, and TSA-Sig) with a parameter value of ST=0.1.

The Mean objective function value of TSA-Mode is 8.18E+05, with a standard deviation of 3.49E+03. The best objective function value obtained is 8.09E+05, and the worst is 8.23E+05. In addition, the TSA-XOR Mean objective function value is 8.47E+05, with a standard deviation of 4.05E+03. The best objective function value obtained is 8.38E+05, and the worst is 8.52E+05. Finally, the TSA-Sig Mean objective function value is 8.26E+05, with a standard deviation of 3.29E+03. The best objective function value obtained is 8.21E+05, and the worst is 8.32E+05. Table 9 outlines the test results for large-sized p0072oblems with ST=0.2, comparing three techniques such as TSA-Mode, TSA-XOR, and TSA-Sig cross problem instances (Cap131 to Cap134). The result reveals slight variations in mean values across techniques, with TSA-Sig generally performing slightly better. Standard deviation indicates variability, while best and worst values show the range of performance. Based on the experimental results obtained by the proposed methods, promising results are obtained in solving uncapacitated facility location problems.

5. Results and Discussion

Results can be evaluated due to several parameters involved in the tackled optimization problem. According to the problem dimension parameter, which is represented by the number of facilities, as seen from Table 2, for Cap71, Cap72, Cap73, and Cap74 problems, all methods exhibit similar performance. However, as the number of facilities increases in Cap101, Cap102, Cap103, Cap104, Cap131, Cap132, and Cap133 problems, TSA-Sig outperforms TSA-XOR. In contrast, TSA-Mode outperforms both of them since it achieves the best cost with a relatively considerable difference from costs obtained by TSA-XOR and TSA- sig, as observed in Tables 5 and 8.

As for the Search Tendency parameter (ST) introduced by TSA, Tables 2, 3, 4, 5, 6 and 7 show that for different ST values (0.1,0.2 and 0.3), the best cost values of the three methods are nearly the same in Cap71, Cap72, Cap73, Cap74, Cap101, Cap102, Cap103 and Cap104 problems. Also, for Cap131 and Cap132, Tables 8, 9 and 10 reveal that there is no observed change in all method's performance due to the different values of ST. At the same time, for Cap 133 and Cap134 problems, TSA-XOR obtains better cost as ST increases, in contrast to the STA-Sig method, in which cost value decreases with ST increment.

Regarding the standard deviation value of all the methods, it has been noted that TSA-Mode and TSA-Sig are more robust than TSA-XOR for all cases since in Cap71, Cap72, and Cap73 problems, TSA-Sig was of the best robustness. While, the TSA-Mode method was more robust in Cap74, as it is seen in Tables 2, 3 and 4. However, for Cap101, Cap102, Cap103 and Cap104 problems, TSA-Mode was the best robust method as seen in Tables 5, 6 and 7.

As for Cap131, Cap132, Cap133 and Cap134 problems, close standard deviations were noted for TSA-Mode and TSA-Sig methods, though TSA-Mode achieved better standard deviations in Cap132, Cap133 and Cap134 problems, as seen in Tables 8, 9 and 10.In conclusion, based on the evaluations conducted above, it can be inferred that the TSA-Mode method surpasses TSA-XOR and TSA-Sig in optimizing the UFLP problem. The TSA-Mode method has demonstrated robustness and consistently achieved the lowest costs across all instances of the UFLP problem addressed. Based on these results, the binary versions of TSA produce promising and competitive results in solving UFLPs in terms of solution quality and robustness.

6. Conclusion

In this paper, the authors examined the adaptation of TSA for addressing binary optimization problems. Three variations of the TSA algorithm were proposed, and their efficacy was evaluated on uncapacitated facility location problems of varying sizes. The performance of these proposed methods was analyzed across different ST values. The experimental results revealed promising outcomes achieved by these methods. Overall, the findings suggest that TSA variants designed for binary optimization hold the potential for effectively addressing a wide range of binary optimization problems.

Acknowledgement

We extend our heartfelt thanks to SIMAD University, especially the Center for Research and Development (CRD), for sponsoring this research with grant number SU-PG-2024-0005. Their support has been invaluable in enabling the completion of this manuscript, contributing significantly to our field of study.

References

[1] J. Kennedy, and R. Eberhart, “Particle Swarm Optimization,” *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia, pp. 1942-1948, 1995. [CrossRef] [Google Scholar] [Publisher Link]

[2] D. Karaboga, and B. Basturk, “On the Performance of Artificial Bee Colony (ABC) Algorithm,” *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 687-697, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[3] Dervis Karaboga, and Bahriye Basturk, “A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm,” *Journal of Global Optimization*, vol. 39, pp. 459-471, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[4] Xin-She Yang, “Firefly Algorithm, Stochastic Test Functions and Design Optimization,” *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78-84, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[5] Xin-She Yang, “A New Metaheuristic Bat-Inspired Algorithm,” *Nature Inspired Cooperative Strategies for Optimization*, pp. 65-74, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[6] J. Kennedy, and R.C. Eberhart, “A Discrete Binary Version of the Particle Swarm Algorithm,” *1997 IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation*, Orlando, FL, USA, pp. 4104-4108, 1997. [CrossRef] [Google Scholar] [Publisher Link]

[7] Mehmet Sevkli, and Ali R. Guner, “A Continuous Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem,” *International Workshop on Ant Colony Optimization and Swarm Intelligence*, pp. 316-323, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[8] Nevena Lazic, Brendan J. Frey, and Parham Aarabi, “Solving the Uncapacitated Facility Location Problem Using Message Passing Algorithms,” *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 429-436, 2010. [Google Scholar] [Publisher Link]

[9] Hazem Ahmed, and Janice Glasgow, “*Swarm Intelligence: Concepts, Models and Applications*,” School of Computing, Queens University, Technical Report, pp. 1-51, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[10] Frederick Ducatelle, Gianni A. Di Caro, and Luca M. Gambardella, “Principles and Applications of Swarm Intelligence for Adaptive Routing in Telecommunications Networks,” *Swarm Intelligence*, vol. 4, no. 3, pp. 173-198, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[11] M. Dorigo, V. Maniezzo, and A. Colorni, “The Ant System: Optimization by a Colony of Cooperating Agents,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29-41, 1996. [CrossRef] [Google Scholar] [Publisher Link]

[12] Arnab Kole, Parichay Chakrabarti, and Somnath Bhattacharyya, “An Ant Colony Optimization Algorithm for Uncapacitated Facility Location Problem,” *Artificial Intelligence and Applications*, vol. 1, no. 1, pp. 55-61, 2014. [Google Scholar]

[13] Yusuke Watanabe, Mayumi Takaya, and Akihiro Yamamura, “Fitness Function in ABC Algorithm for Uncapacitated Facility Location Problem,” *3rd International Conference on Information and Communication Technology-EurAsia (ICT-EURASIA) and 9th International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS)*, Daejeon, Korea, pp. 129-138, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[14] Mauricio G.C. Resende, and Renato F. Werneck, “A Hybrid Multistart Heuristic for the Uncapacitated Facility Location Problem,” *European Journal of Operational Research*, vol. 174, no. 1, pp. 54-68, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[15] J.E. Beasley, “OR-Library: Distributing Test Problems by Electronic Mail,” *Journal of the Operational Research Society*, vol. 41, no. 11, pp. 1069-1072, 1990. [CrossRef] [Google Scholar] [Publisher Link]

[16] Ahamet Cevahir Çınar, “*A Cuda-based Parallel Programming Approach to Tree-Seed Algorithm*,” MSc Thesis, Selçuk University, pp. 1-111, 2016. [Google Scholar] [Publisher Link]