Conference Paper

Tree-Seed algorithm for large-scale binary optimization

Ahmet Cevahir CINAR, Hazim ISCAN, and Mustafa Servet KIRAN

Dept. of Computer Engineering, Faculty of Engineering, Konya, Turkey

Abstract

Population-based swarm or evolutionary computation algorithms in optimization are attracted the interest of the researchers due their simple structure, optimization performance, easy-adaptation. Binary optimization problems can be also solved by using these algorithms. This paper focuses on solving large scale binary optimization problems by using Tree-Seed Algorithm (TSA) proposed for solving continuous optimization problems by imitating relationship between the trees and their seeds in nature. The basic TSA is modified by using xor logic gate for solving binary optimization problems in this study. In order to investigate the performance of the proposed algorithm, the numeric benchmark problems with the different dimensions are considered and obtained results show that the proposed algorithm produces effective and comparable solutions in terms of solution quality.

Keywords: binary optimization, tree-seed algorithm, xor-gate, large-scale optimization

Corresponding Author:
Ahmet Cevahir CINAR
ahmetcevahircinar@gmail.com

Received: 14 November 2017
Accepted: 25 December 2017
Published: 8 January 2018

Publishing services provided by Knowledge E

Selection and Peer-review under the responsibility of the IAIT Conference Committee.

1. Introduction

Many important optimization problems can be modeled as binary optimization problem and some of them try to solve high dimensional optimization problems. The solution of large-scale binary optimization problems is an interesting and important research area. There are two main solution approaches in the literature: exact methods and approximate methods. The exact methods guarantee the optimal solution but large-scale optimization problems cannot be solved in a reasonable time by these methods. Therefore, heuristic/metaheuristic optimization or search methods offer optimal or near optimal solutions in a reasonable time. In last decades, a lot of population-based optimization algorithm were proposed to solve different kind of optimization problems. TSA is one of them and it mimics relationship between trees and seeds in nature [1] for continuous optimization. A method proposed for solving continuous optimization problems should be adapted to binary optimization before applying it to solve a binary optimization problem. There are some binarization approaches in literature. Banitalebi et al. [2] have reviewed the literature and have

made the following grouping. Transfer function [3-6], Angle modulation [7], Quantum-inspired bits [8], Genetic operators [9], Binary operators [10], Measure of dissimilarity [11, 12] are some modification techniques which detailed in the references.

Kashan et al. [12] introduce a binary version of ABC, called DisABC, which is used Jaccard's [13] coefficient of similarity. Also similarly DisDE [11] has been proposed by Kashan et al. DisDE algorithm uses a measure of dissimilarity between binary structures. Binary Particle Swarm Optimization (BPSO) algorithm is proposed by Kennedy and Eberhart [6]. BPSO uses sigmoid function to transform continuous values to binary values. Binary Artificial Bee Colony (binABC) is designed with XOR logic gate by Kıran and Gündüz [14]. The binABC works on binary space and use XOR logic gate for creating new individuals. XOR logic gate enhances population diversity because of its design. By inspiring [14], we used the xor logic operator in TSA for solving binary optimization problems in this study.

Table 1: Mathematical Benchmark Functions Properties.

Name	Search Range	Type
Sphere	[-100,100]	Unimodal Separable
Rosenbrock	[-10,10]	Unimodal Non-Separable
Rastrigin	[-5.12,5.12]	Multimodal Separable
Griwank	[-600,600]	Multimodal Non-Separable
Ackley	[-32,32]	Multimodal Non-Separable

2. Tree-Seed Algorithm

Tree-Seed Algorithm (TSA) proposed by Kıran [1] mimics relationship between trees and seeds in nature. Trees and seeds correspond to solutions in an optimization problem. TSA is a population-based algorithm. Firstly trees are created by using (1).

$$T_{i,j} = Low_j + r_{i,j}\left(High_j - Low_j\right) \tag{1}$$

where $T_{i,j}$ is jth dimension of ith tree, $r_{i,j}$ is uniformly random number in range of [0, 1], Low_j is lower bound of jth dimension, $High_j$ is higher bound of jth dimension. Then every iteration seeds are created by using (2) or (3).

$$S_{i,j} = T_{i,j} + \alpha_{i,j} \times \left(Best_j - T_{r,j}\right) \tag{2}$$

$$S_{i,j} = T_{i,j} + \alpha_{i,j} \times \left(T_{i,j} - T_{r,j}\right) \tag{3}$$

Table 2: Results of XORTSA on Sphere Function with 250, 500 and 1000 Dimensions.

Sphere Function, Dimension:5 x 50bit = 250bit					
XORTSA	ST=0.1	ST=0.2	ST=0.3	ST=0.4	ST=0.5
Pop=10	8.19E-08	2.24E-04	5.21E-02	6.35E-01	1.17E+00
Pop=20	8.58E-10	9.10E-05	3.84E-02	3.37E-01	8.08E-01
Pop=30	4.78E-09	2.47E-04	2.04E-02	2.22E-01	5.90E-01
Pop=40	2.56E-08	2.25E-04	3.14E-02	2.36E-01	9.38E-01
Pop=50	2.02E-07	7.90E-04	2.50E-02	3.35E-01	6.24E-01
Sphere Function, Dimension: 10 x 50bit = 500bit					
XORTSA	ST=0.1	ST=0.2	ST=0.3	ST=0.4	ST=0.5
Pop=10	1.38E-02	7.46E+00	6.33E+01	2.20E+02	2.76E+02
Pop=20	1.81E-02	1.02E+01	8.65E+01	2.67E+02	3.56E+02
Pop=30	4.27E-02	1.25E+01	1.04E+02	2.85E+02	3.70E+02
Pop=40	1.46E-01	1.84E+01	1.14E+02	2.86E+02	4.67E+02
Pop=50	2.48E-01	2.13E+01	1.53E+02	3.10E+02	4.99E+02
Sphere Function, Dimension: 20 x 50bit = 1000bit					
XORTSA	ST=0.1	ST=0.2	ST=0.3	ST=0.4	ST=0.5
Pop=10	4.15E+01	1.22E+03	3.07E+03	4.94E+03	5.90E+03
Pop=20	9.66E+01	1.66E+03	4.15E+03	5.99E+03	7.26E+03
Pop=30	1.49E+02	2.01E+03	4.32E+03	6.81E+03	7.68E+03
Pop=40	2.27E+02	2.34E+03	4.88E+03	6.70E+03	8.36E+03
Pop=50	3.15E+02	2.50E+03	5.02E+03	7.07E+03	8.50E+03

where, $S_{i,j}$ is jth dimension of ith seed, $T_{i,j}$ is jth dimension of ith tree, $\alpha_{i,j}$ is uniformly random number in range of [-1,1] using for scaling, $Best_j$ is jth dimension of best tree which is obtained so far, $T_{r,j}$ is jth dimension of rth tree and r and i must be different. Which equation is selected for creating a seed is an issue in TSA and it is solved by a control parameter, whose name is ST (Search Tendency). If (2) is selected, exploitation capability is increased; otherwise exploration skill enhances. ST is determined at initialization as a number between 0 and 1. Then at every iteration, a uniformly random number generated and is compared ST. If this random value is smaller than ST then (2) is used for seed creation otherwise (3) is used. The number of seeds which will be

Table 3: Results of XORTSA on Rosenbrock Function with 250, 500 and 1000 Dimensions.

Rosenbrock Function, Dimension:5x50bit=250bit					
XORTSA	ST=0.1	ST=0.2	ST=0.3	ST=0.4	ST=0.5
Pop=10	2.42E+00	2.42E+00	2.21E+00	3.05E+00	3.96E+00
Pop=20	2.23E+00	1.54E+00	1.54E+00	2.98E+00	4.10E+00
Pop=30	2.25E+00	1.58E+00	1.85E+00	2.90E+00	3.87E+00
Pop=40	1.85E+00	1.89E+00	2.03E+00	2.77E+00	3.66E+00
Pop=50	1.69E+00	1.55E+00	2.07E+00	2.87E+00	4.90E+00
Rosenbrock Function, Dimension:10x50bit=500bit					
XORTSA	ST=0.1	ST=0.2	ST=0.3	ST=0.4	ST=0.5
Pop=10	8.08E+00	2.34E+01	1.30E+02	3.47E+02	4.34E+02
Pop=20	7.86E+00	2.30E+01	1.34E+02	3.50E+02	5.88E+02
Pop=30	8.27E+00	3.06E+01	1.62E+02	3.57E+02	6.59E+02
Pop=40	8.56E+00	4.07E+01	1.91E+02	4.26E+02	7.58E+02
Pop=50	9.07E+00	5.29E+01	2.58E+02	5.00E+02	8.48E+02
Rosenbrock Function, Dimension:20x50bit=1000bit					
XORTSA	ST=0.1	ST=0.2	ST=0.3	ST=0.4	ST=0.5
Pop=10	1.33E+02	3.33E+03	1.84E+04	3.27E+04	4.78E+04
Pop=20	2.00E+02	5.38E+03	2.15E+04	4.34E+04	5.27E+04
Pop=30	3.16E+02	6.38E+03	2.69E+04	5.10E+04	6.22E+04
Pop=40	4.84E+02	8.10E+03	3.28E+04	5.42E+04	7.44E+04
Pop=50	6.26E+02	9.59E+03	3.29E+04	6.25E+04	7.77E+04

created for each tree is changeable in TSA. This is controlled properly using population size and using (4).

$$NSd = 0.1 \times N \quad NSu = 0.25 \times N \tag{4}$$

where, NSd, are NSu are the lower and upper bounds for the number of seeds created for each tree, respectively. TSA works on continuous solution space and in this work TSA has been modified by using xor logic operator to work on binary space.

Table 4: Results of XORTSA on Rastrigin Function with 250, 500 and 1000 Dimensions.

Rastrigin Function, Dimension:5x50bit=250bit					
XORTSA	ST=0.1	ST=0.2	ST=0.3	ST=0.4	ST=0.5
Pop=10	2.65E-03	7.57E-02	2.22E-01	8.28E-01	1.56E+00
Pop=20	3.32E-02	1.41E-02	1.74E-01	1.23E+00	2.55E+00
Pop=30	3.32E-02	3.40E-03	4.06E-01	1.46E+00	2.25E+00
Pop=40	1.15E-01	5.52E-02	7.71E-01	2.03E+00	3.00E+00
Pop=50	8.39E-02	1.05E-01	7.43E-01	2.53E+00	3.21E+00
Rastrigin Function, Dimension:10x50bit=500bit					
XORTSA	ST=0.1	ST=0.2	ST=0.3	ST=0.4	ST=0.5
Pop=10	3.58E+00	1.14E+01	2.12E+01	2.77E+01	2.93E+01
Pop=20	4.20E+00	1.26E+01	2.20E+01	2.97E+01	3.39E+01
Pop=30	5.28E+00	1.41E+01	2.43E+01	3.34E+01	3.45E+01
Pop=40	5.82E+00	1.52E+01	2.46E+01	3.01E+01	3.82E+01
Pop=50	6.63E+00	1.61E+01	2.46E+01	3.21E+01	3.75E+01
Rastrigin Function, Dimension:20x50bit=1000bit					
XORTSA	ST=0.1	ST=0.2	ST=0.3	ST=0.4	ST=0.5
Pop=10	4.03E+01	8.19E+01	1.03E+02	1.18E+02	1.27E+02
Pop=20	4.32E+01	8.71E+01	1.16E+02	1.36E+02	1.40E+02
Pop=30	4.85E+01	9.23E+01	1.17E+02	1.40E+02	1.49E+02
Pop=40	5.04E+01	9.53E+01	1.19E+02	1.37E+02	1.49E+02
Pop=50	5.30E+01	9.40E+01	1.19E+02	1.41E+02	1.51E+02

3. XOR-Based Binary Tree-Seed Algorithm

The element of {0,1} set can be assigned to the binary decision variables in binary solution space. In the initialization of binary decision variables, 0 or 1 values can be assigned by equal probability. This approach is fairly easy to understand. A random number is created between 0 and 1. If this random number smaller than 0.5, binary value is 0, otherwise binary value is 1. XOR gate is used in the seed production phase after the trees are initialized with the binary values. Seed creation equations of basic TSA are changed as follows:

$$S_{kj} = \begin{cases} T_{ij} \oplus \left(B_j \oplus T_{rj}\right) & if\ (rand_{ij} < ST) \\ T_{ij} & otherwise \end{cases} \quad (5)$$

Table 5: Results of XORTSA on Griewank Function with 250, 500 and 1000 Dimensions.

Griewank Function, Dimension:5x50bit=250bit					
XORTSA	ST=0.1	ST=0.2	ST=0.3	ST=0.4	ST=0.5
Pop=10	2.23E-02	4.16E-02	1.41E-01	2.90E-01	4.35E-01
Pop=20	1.59E-02	4.73E-02	1.89E-01	2.46E-01	3.96E-01
Pop=30	2.24E-02	6.73E-02	1.56E-01	2.99E-01	3.78E-01
Pop=40	2.15E-02	7.24E-02	1.62E-01	2.73E-01	3.51E-01
Pop=50	2.31E-02	7.37E-02	1.76E-01	3.30E-01	3.90E-01
Griewank Function, Dimension:10x50bit=500bit					
XORTSA	ST=0.1	ST=0.2	ST=0.3	ST=0.4	ST=0.5
Pop=10	1.40E-01	9.58E-01	1.64E+00	2.84E+00	3.70E+00
Pop=20	1.63E-01	9.75E-01	1.68E+00	3.17E+00	4.36E+00
Pop=30	2.67E-01	1.05E+00	2.02E+00	3.53E+00	4.25E+00
Pop=40	3.45E-01	1.14E+00	2.08E+00	4.04E+00	5.08E+00
Pop=50	4.22E-01	1.21E+00	2.28E+00	4.26E+00	5.27E+00
Griewank Function, Dimension:20x50bit=1000bit					
XORTSA	ST=0.1	ST=0.2	ST=0.3	ST=0.4	ST=0.5
Pop=10	1.45E+00	1.20E+01	2.84E+01	4.47E+01	5.25E+01
Pop=20	1.84E+00	1.59E+01	3.75E+01	5.35E+01	6.36E+01
Pop=30	2.39E+00	2.01E+01	4.10E+01	5.67E+01	7.17E+01
Pop=40	2.94E+00	2.13E+01	4.59E+01	6.17E+01	7.54E+01
Pop=50	3.71E+00	2.39E+01	4.61E+01	6.40E+01	7.58E+01

S_{kj} is the jth dimension of kth seed produced for ith tree, T_{ij} is the jth dimension of ith tree, B_j is the jth dimension of best tree obtained so far, and T_{rj} is the jth dimension of neighbor tree randomly selected from the population. The main difference between TSA and XORTSA is the seed production equations. TSA uses (2) or (3), XORTSA uses (5) for seed creation because while output of 2 or 3 can be continuous values, the output of 5 is absolutely binary value.

Table 6: Results of XORTSA on Ackley Function with 250, 500 and 1000 Dimensions.

Ackley Function, Dimension:5x50bit=250bit					
XORTSA	ST=0.1	ST=0.2	ST=0.3	ST=0.4	ST=0.5
Pop=10	3.69E-05	9.97E-03	1.14E-01	1.00E+00	1.17E+00
Pop=20	1.10E-05	3.74E-03	9.02E-02	4.51E-01	9.27E-01
Pop=30	2.97E-05	7.35E-03	1.22E-01	4.37E-01	1.14E+00
Pop=40	6.76E-05	8.09E-03	1.18E-01	5.45E-01	1.08E+00
Pop=50	3.50E-04	1.87E-02	1.61E-01	5.41E-01	1.10E+00
Ackley Function, Dimension:10x50bit=500bit					
XORTSA	ST=0.1	ST=0.2	ST=0.3	ST=0.4	ST=0.5
Pop=10	8.92E-02	2.08E+00	4.73E+00	6.65E+00	7.27E+00
Pop=20	7.93E-02	2.43E+00	5.13E+00	6.78E+00	8.12E+00
Pop=30	1.38E-01	2.79E+00	5.22E+00	7.29E+00	7.74E+00
Pop=40	2.98E-01	3.07E+00	5.67E+00	7.40E+00	8.54E+00
Pop=50	5.13E-01	3.56E+00	5.72E+00	8.52E+00	8.87E+00
Ackley Function, Dimension:20x50bit=1000bit					
XORTSA	ST=0.1	ST=0.2	ST=0.3	ST=0.4	ST=0.5
Pop=10	3.35E+00	9.40E+00	1.29E+01	1.43E+01	1.44E+01
Pop=20	4.25E+00	1.04E+01	1.34E+01	1.46E+01	1.55E+01
Pop=30	4.98E+00	1.13E+01	1.39E+01	1.53E+01	1.57E+01
Pop=40	5.61E+00	1.17E+01	1.40E+01	1.55E+01	1.61E+01
Pop=50	6.32E+00	1.20E+01	1.45E+01	1.58E+01	1.63E+01

4. Binary to Continuos Transformation

Before evaluating the objective function with binary values, the binary values must be converted to continuous values with the aid of a conversion function. This transformation operation is carried out as follows:

1. Input: binary string (X),
2. Input: lower and upper bounds of solutions space (low, up),
3. Input: dimensionality (D),
4. Input: the bit size (bitSize),
5. Calculate maximum value (maxVal) calculated as 2bitSize-1,

Table 7: Comparison of XORTSA and BPSO on Sphere Function.

	BPSO		XORTSA	
Sphere Function, Dimension:5x50bit=250bit				
Pop	Mean	Std	Mean	Std
10	3.64E-26	0.00E+00	8.19E-08	4.44E-07
20	2.81E-22	1.51E-21	8.58E-10	2.34E-09
30	1.90E-11	1.04E-10	4.78E-09	2.08E-08
40	1.31E+00	7.13E+00	2.56E-08	5.52E-08
50	1.15E-02	3.93E-02	2.02E-07	5.18E-07
Sphere Function, Dimension:10x50bit=500bit				
Pop	Mean	Std	Mean	Std
10	5.62E-24	7.27E-26	1.38E-02	2.32E-02
20	2.14E-11	2.93E-21	1.81E-02	1.73E-02
30	3.27E-01	2.17E-15	4.27E-02	2.40E-02
40	5.55E+00	1.02E-11	1.46E-01	1.17E-01
50	2.00E+00	1.47E-07	2.48E-01	1.71E-01
Sphere Function, Dimension: 20x50bit=1000bit				
Pop	Mean	Std	Mean	Std
10	3.05E-10	9.53E-16	4.15E+01	2.06E+01
20	1.31E+00	2.22E-09	9.66E+01	3.53E+01
30	5.48E+00	8.30E-06	1.49E+02	2.86E+01
40	5.14E+01	4.46E-03	2.27E+02	5.72E+01
50	1.10E+02	9.66E-02	3.15E+02	8.88E+01

6. Divide X into x with D dimensional,

7. Do the following operations for all x

a. Calculate decimal value (decVal) for all x

b. Calculate continuous values (conVal) by conVal=low+(up-low)*(decVal/maxVal)

8. Output: continuous valued vector of x (conVal).

In our work bit size is taken as 50 due to floating point precision. Below you see an example how this transformation operation is done.

1. X= 1101 100111,

2. low=-100, up=100,

Table 8: Comparison of XORTSA and BPSO on Rosenbrock Function.

	BPSO		XORTSA	
Rosenbrock Function, Dimension:5x50bit=250bit				
Pop	Mean	Std	Mean	Std
10	1.53E+02	2.33E+02	2.42E+00	8.94E-01
20	2.08E+02	2.38E+02	2.23E+00	8.87E-01
30	1.65E+02	1.91E+02	2.25E+00	7.56E-01
40	2.35E+02	2.75E+02	1.85E+00	7.18E-01
50	1.34E+02	2.21E+02	1.69E+00	7.16E-01
Rosenbrock Function, Dimension:10x50bit=500bit				
Pop	Mean	Std	Mean	Std
10	2.96E+02	3.03E+02	8.08E+00	1.05E+00
20	2.21E+02	2.77E+02	7.86E+00	8.70E-01
30	2.86E+02	2.88E+02	8.27E+00	8.62E-01
40	1.85E+02	2.42E+02	8.56E+00	6.77E-01
50	4.36E+02	6.74E+02	9.07E+00	8.85E-01
Rosenbrock Function, Dimension: 20x50bit=1000bit				
Pop	Mean	Std	Mean	Std
10	3.73E+02	3.44E+02	1.33E+02	5.02E+01
20	2.67E+02	2.38E+02	2.00E+02	5.31E+01
30	3.22E+02	3.22E+02	3.16E+02	1.02E+02
40	6.98E+02	8.94E+02	4.84E+02	1.39E+02
50	6.21E+02	4.14E+02	6.26E+02	2.21E+02

3. D=1,
4. bitSize=10;
5. Dim=1*10=10;
6. maxVal=2^{10}-1=1023;
7. decVal=871,
8. conVal=-100+(100-(-100))*(871/1023)
9. conVal=70,2834799608993.

Table 9: Comparison of XORTSA and BPSO on Rastrigin Function.

	BPSO		XORTSA	
Rastrigin Function, Dimension:5x50bit=250bit				
Pop	Mean	Std	Mean	Std
10	1.00E+01	4.10E+00	2.65E-03	1.45E-02
20	1.22E+01	6.20E+00	3.32E-02	1.82E-01
30	1.25E+01	6.15E+00	3.32E-02	1.82E-01
40	1.16E+01	4.72E+00	1.15E-01	3.09E-01
50	1.04E+01	5.23E+00	8.39E-02	2.60E-01
Rastrigin Function, Dimension:10x50bit=500bit				
Pop	Mean	Std	Mean	Std
10	2.30E+01	6.46E+00	3.58E+00	1.59E+00
20	2.46E+01	8.12E+00	4.20E+00	1.74E+00
30	2.28E+01	6.50E+00	5.28E+00	1.51E+00
40	2.33E+01	6.62E+00	5.82E+00	1.64E+00
50	2.40E+01	8.90E+00	6.63E+00	1.74E+00
Rastrigin Function, Dimension: 20x50bit=1000bit				
Pop	Mean	Std	Mean	Std
10	4.51E+01	7.87E+00	4.03E+01	5.36E+00
20	4.88E+01	1.27E+01	4.32E+01	5.07E+00
30	4.97E+01	1.31E+01	4.85E+01	5.37E+00
40	5.07E+01	1.18E+01	5.04E+01	4.30E+00
50	5.10E+01	9.67E+00	5.30E+01	4.71E+00

5. Numerical Benchmark Problems

For investigating the performance of XORTSA, five numerical benchmark functions, namely Sphere, Rosenbrock, Rastrigin, Griewank and Ackley are considered in the experiments. The description of these mathematical benchmark functions properties are given in Table 1.

Table 10: Comparison of XORTSA and BPSO on Griewank Function.

	BPSO		XORTSA	
Griewank Function, Dimension:5x50bit=250bit				
Pop	Mean	Std	Mean	Std
10	5.40E-01	4.94E-01	2.23E-02	1.25E-02
20	4.20E-01	2.62E-01	1.59E-02	1.06E-02
30	4.84E-01	3.45E-01	2.24E-02	1.15E-02
40	4.81E-01	3.78E-01	2.15E-02	1.19E-02
50	7.13E-01	1.12E+00	2.31E-02	1.12E-02
Griewank Function, Dimension:10x50bit=500bit				
Pop	Mean	Std	Mean	Std
10	3.29E-01	2.54E-01	1.40E-01	6.80E-02
20	3.55E-01	2.45E-01	1.63E-01	6.64E-02
30	3.73E-01	2.42E-01	2.67E-01	8.03E-02
40	3.23E-01	2.98E-01	3.45E-01	8.10E-02
50	4.91E-01	5.93E-01	4.22E-01	1.02E-01
Griewank Function, Dimension: 20x50bit=1000bit				
Pop	Mean	Std	Mean	Std
10	3.58E-01	2.58E-01	1.45E+00	2.16E-01
20	3.78E-01	2.49E-01	1.84E+00	2.89E-01
30	4.67E-01	3.30E-01	2.39E+00	3.58E-01
40	1.09E+00	1.36E+00	2.94E+00	4.82E-01
50	1.96E+00	2.50E+00	3.71E+00	7.01E-01

5.1. Sphere Function

Sphere function is a unimodal and separable funtion. It has a global minimum f(x) = 0 where x = [0,...,0]^D. This function's search range is [-100, 100].

$$f(x) = \sum_{i=1}^{D} x_i^2 \tag{6}$$

Table 11: Comparison of XORTSA and BPSO on Ackley Function.

	BPSO		XORTSA	
Ackley Function, Dimension:5x50bit=250bit				
Pop	Mean	Std	Mean	Std
10	1.74E+00	1.28E+00	3.69E-05	8.53E-05
20	1.83E+00	1.25E+00	1.10E-05	1.56E-05
30	1.90E+00	1.28E+00	2.97E-05	2.79E-05
40	2.03E+00	1.17E+00	6.76E-05	8.78E-05
50	1.80E+00	1.26E+00	3.50E-04	2.71E-04
Ackley Function, Dimension:10x50bit=500bit				
Pop	Mean	Std	Mean	Std
10	2.44E+00	3.41E-01	8.92E-02	1.07E-01
20	2.20E+00	6.96E-01	7.93E-02	3.87E-02
30	2.41E+00	3.66E-01	1.38E-01	7.25E-02
40	2.34E+00	6.39E-01	2.98E-01	1.89E-01
50	2.68E+00	1.13E+00	5.13E-01	2.02E-01
Ackley Function, Dimension:20x50bit=1000bit				
Pop	Mean	Std	Mean	Std
10	2.41E+00	3.08E-01	3.35E+00	7.81E-01
20	2.45E+00	2.98E-01	4.25E+00	5.01E-01
30	2.45E+00	4.16E-01	4.98E+00	5.76E-01
40	2.69E+00	6.20E-01	5.61E+00	4.72E-01
50	3.72E+00	1.58E+00	6.32E+00	4.45E-01

5.2. Rosenbrock Function

Rosenbrock function is unimodal and non-separable funtion. It has a global minimum f(x) = 0 where x = $[1,\ldots,1]^D$. This function's search range is [-10, 10].

$$f(x) = \sum_{i=1}^{D-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2 \right] \tag{7}$$

5.3. Rastrigin Function

Rastrigin function is multimodal and separable funtion. It has a global minimum f(x) = 0 where x = [0,...,0]^D. This function's search range is [-5.12, 5.12].

$$f(x) = \sum_{i=1}^{D} \left[{x_i}^2 - 10\cos\left(2\pi x_i\right) + 10 \right] \tag{8}$$

5.4. Griewank Function

Griewank function is multimodal and non-separable. It has a global minimum f(x) = 0 where x = [0,...,0]^D. This function's search range is [-600, 600]. This function is relatively easy to solve in high dimensional cases [15].

$$f(x) = \frac{1}{400} \sum_{i=1}^{D} {x_i}^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{9}$$

5.5. Ackley Function

Ackley function is multimodal and non-separable. It has a global minimum f(x) = 0 where x = [0,...,0]^D. This function's search range is [-32, 32].

$$f(x) = -20\exp\left\{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} {x_i}^2}\right\} - \exp\left\{\frac{1}{D}\sum_{i=1}^{D}\cos\left(2\pi x_i\right)\right\} + 20 + e \tag{10}$$

6. Experimental Results

In this work, different type of benchmark functions are used for analyzing BPSO and XORTSA. Dimensionality is taken as 250, 500 and 1000. Function's search ranges are fixed as seen in Table 1. Population size is taken as 10, 20, 30, 40 and 50. Maximum function evaluation number is taken as 100000. ST is taken as 0.1, 0.2, 0.3, 0.4 and 0.5. All results are mean of 30 independent runs with random seeds.

The following Table 2, Table 3, Table 4, Table 5 and Table 6 show the results of the XORTSA for 5 benchmark functions in 250, 500, 1000 dimensions and 10, 20, 30, 40, 50 populations.

The following Table VII, Table VIII, Table IX, Table X and Table XI show the comparison of XORTSA and BPSO results for the 5 functions in 250, 500, 1000 dimensions and 10, 20, 30, 40, 50 populations.

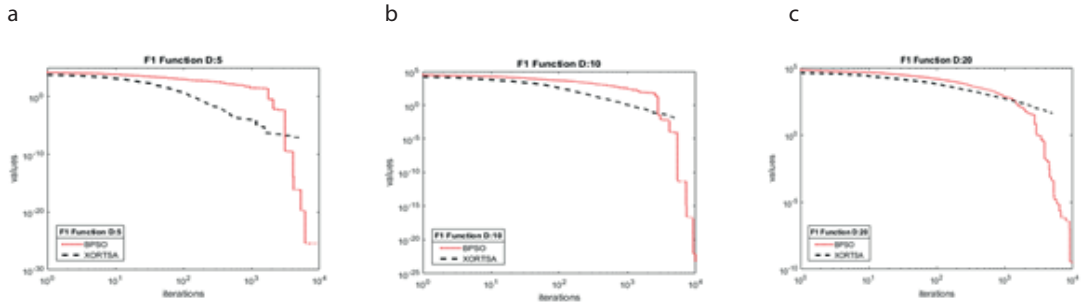

Figure 1: (a) Convergence graphs on Sphere function for 250 dimension; (b) Convergence graphs on Sphere function for 500 dimension; (c) Convergence graphs on Sphere function for 1000 dimension.

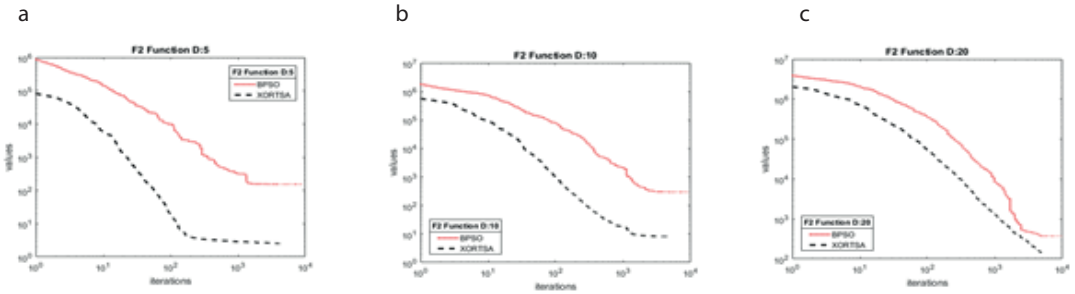

Figure 2: (a) Convergence graphs on Rosenbrock function for 250 dimension; (b) Convergence graphs on Rosenbrock function for 500 dimension; (c) Convergence graphs on Rosenbrock function for 1000 dimension.

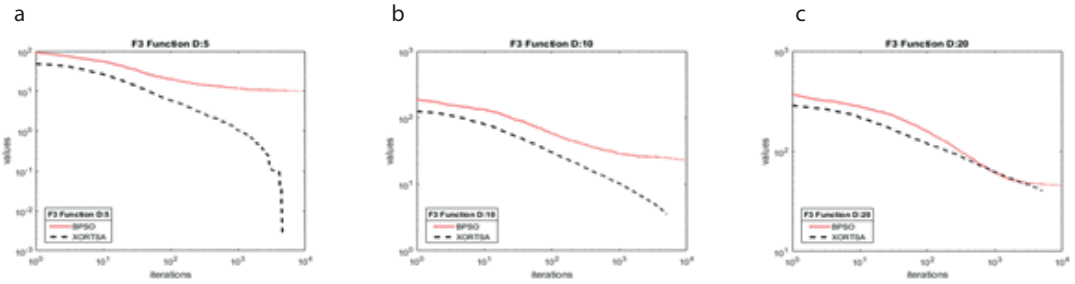

Figure 3: (a) Convergence graphs on Rastrigin function for 250 dimension; (b) Convergence graphs on Rastrigin function for 500 dimension; (c) Convergence graphs on Rastrigin function for 1000 dimension.

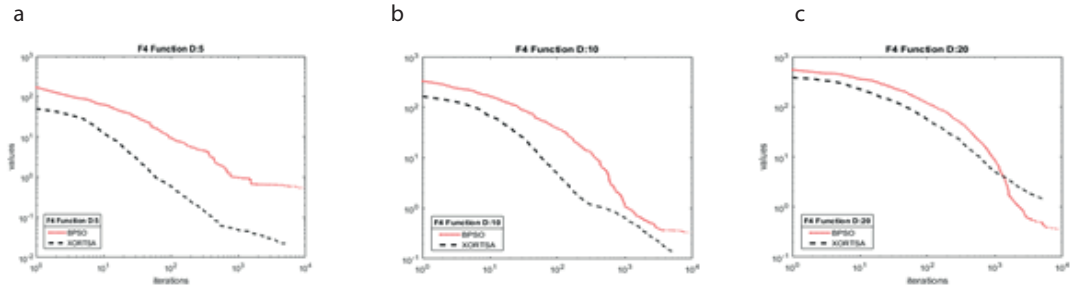

Figure 4: (a) Convergence graphs on Griewank function for 250 dimension; (b) Convergence graphs on Griewank function for 500 dimension; (c) Convergence graphs on Griewank function for 1000 dimension.

The comparisons of the convergences of XORTSA and BPSO are given in Figure 1-5 for different dimensional functions.

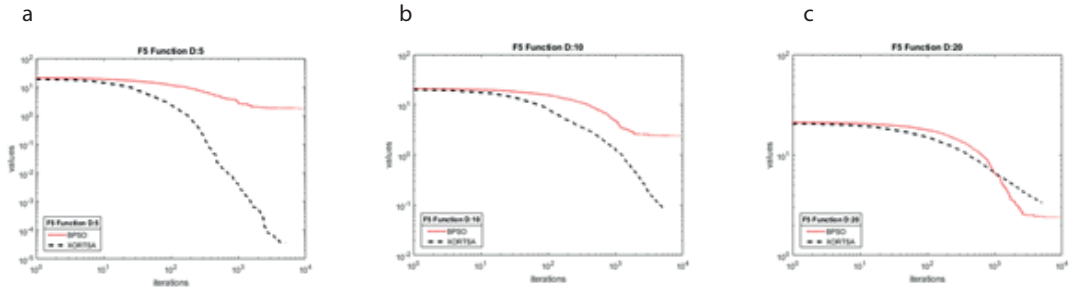

Figure 5: (a) Convergence graphs on Ackley function for 250 dimension; (b) Convergence graphs on Ackley function for 500 dimension; (c) Convergence graphs on Ackley function for 1000 dimension.

In the Sphere function, BPSO is better than XORTSA on 1000 dimensional cases and lower population sizes on other cases. When we consider population size is higher than 30 and dimensions for functions are 5 or 10, the XORTSA is better than BPSO.

According to the comparison results, XORTSA is better than BPSO on all sizes and populations on Rosenbrock and Rastrigin functions.

In Griewank and Ackley functions, on 250 and 500 dimensional cases, XORTSA produces better results than BPSO, while BPSO produces better results on 1000 dimensional case.

According to the convergence graphs given in Figures 1-15, the convergence performance of the XORTSA shows acceptable and comparable convergence characteristics when compared with the BPSO algorithm.

Generally speaking, XORTSA shows better performance on solving multimodal non-separable functions and solving these functions are harder than the others.

7. Conclusion

In this paper, a new method for solving binary optimization problems is presented. The binary optimization problem is solved by using the tree-seed algorithm with the XOR logic operator. The proposed method (XORTSA) has been compared with BPSO. To compare XORTSA and BPSO, five benchmark functions are used in different features. According to the convergence graphs shown in Figure 1-15, the convergence speed of XORTSA is better than BPSO on almost all cases. According to the experimental results, the proposed method (XORTSA) produced better results than BPSO on some functions. Experimental results show that XORTSA can be applied to solve large-scale binary optimization problems.

ACKNOWLEDGMENT

This study has been supported by Scientific Research Projects Coordinatorship of Selcuk University.

References

[1] M. S. Kiran, "TSA: Tree-seed algorithm for continuous optimization," Expert Systems with Applications, vol. 42, pp. 6686-6698, 2015.

[2] A. Banitalebi, M. I. A. Aziz, and Z. A. Aziz, "A self-adaptive binary differential evolution algorithm for large scale binary optimization problems," Information Sciences, vol. 367, pp. 487-511, 2016.

[3] Z. Beheshti, S. M. Shamsuddin, and S. Hasan, "Memetic binary particle swarm optimization for discrete optimization problems," Information Sciences, vol. 299, pp. 58-84, 2015.

[4] H. Nezamabadi-pour, M. Rostami-Shahrbabaki, and M. Maghfoori-Farsangi, "Binary particle swarm optimization: challenges and new solutions," CSI J Comput Sci Eng, vol. 6, pp. 21-32, 2008.

[5] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, 1998, pp. 69-73.

[6] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on, 1997, pp. 4104-4108.

[7] G. Pampara, A. P. Engelbrecht, and N. Franken, "Binary differential evolution," in Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, 2006, pp. 1873-1879.

[8] H. Nezamabadi-pour, "A quantum-inspired gravitational search algorithm for binary encoded optimization problems," Engineering Applications of Artificial Intelligence, vol. 40, pp. 62-75, 2015.

[9] C. Ozturk, E. Hancer, and D. Karaboga, "A novel binary artificial bee colony algorithm based on genetic operators," Information Sciences, vol. 297, pp. 154-170, 2015.

[10] C. Deng, B. Zhao, Y. Yang, H. Peng, and Q. Wei, "Novel binary encoding differential evolution algorithm," Advances in Swarm Intelligence, pp. 416-423, 2011.

[11] M. H. Kashan, A. H. Kashan, and N. Nahavandi, "A novel differential evolution algorithm for binary optimization," Computational Optimization and Applications, vol. 55, p. 481, 2013.

[12] M. H. Kashan, N. Nahavandi, and A. H. Kashan, "DisABC: a new artificial bee colony algorithm for binary optimization," Applied Soft Computing, vol. 12, pp. 342-352, 2012.

[13] P. Jaccard, "Étude comparative de la distribution florale dans une portion des Alpes et des Jura," Bull Soc Vaudoise Sci Nat, vol. 37, pp. 547-579, 1901.

[14] M. S. Kiran and M. GÜNDÜZ„ "XOR-based artificial bee colony algorithm for binary optimization," Turkish Journal of Electrical Engineering & Computer Sciences, vol. 21, pp. 2307-2328, 2013.

[15] M. Locatelli, "A Note on the Griewank Test Function," Journal of Global Optimization, vol. 25, pp. 169-174, February 01 2003Van der Geer J, Hanraads JAJ, Lupton RA. The art of writing a scientific article. J Sci Commun 2000;163:51-59.