International Journal of
Intelligent Systems and Applications in Engineering

ISSN:2147-6799 www.ijisae.org Original Research Paper

Binary tree-seed algorithms with S-shaped and V-shaped transfer functions

Mehmet Akif Sahman*[1], Ahmet Cevahir Cinar*[2]

Submitted :14/06/2019 Accepted: 27/06/2019

Abstract: Tree-seed algorithm (TSA) is a nature-inspired metaheuristic optimization algorithm. TSA was originally designed and introduced for solving continuous optimization problems. In this study, TSA was modified with transfer functions so as to solve binary optimization problems. Continuous search space was mapped to binary search space with transfer functions. Four S-shaped and four V-shaped transfer functions were used for discretization. Uncapacitated facility location problem (UFLP) is a pure binary optimization problem. In order to measure the performance, 15 different sized (small, medium, large and extra-large) UFLPs were solved with eight different binary TSAs in this study. Experimental results has shown that S-shaped transfer functions are better than V-shaped transfer functions on these problem sets.

Keywords: *transfer functions, logistic functions, s-shaped transfer functions, v-shaped transfer functions, tree-seed algorithm, binary optimization*

1. Introduction

Optimization problems are divided into two main groups according to the type of decision variables. These are continuous and discrete optimization problems. Discrete optimization problems are divided into two main groups as binary discrete and permutation coded discrete. Binary optimization problems have two different decision variables which are 0 and 1. Tree-seed algorithm (TSA) was proposed by Kiran [1] for solving continuous (real-valued) optimization problems. In literature, binary versions of TSA [2, 3] were proposed. In these studies, similarity measures and logic gates were used for creating binary variables. Due to there are many studies in literature, only the studies performed with particle swarm optimization (PSO), differential search algorithm (DSA) and TSA were mentioned. Kennedy and Eberhart [4] were proposed binary particle swarm optimization (BPSO) algorithm which uses the sigmoid function as a transfer function. Sevkli and Guner [5] were solved UFLP with BPSO In this study, the sigmoid function was used for mapping continuous search space to binary search space. In order to improve the performance, a local search mechanism was integrated to the BPSO. BPSO has outperformed the genetic algorithm (GA) and evolutionary simulated annealing (ESA). Sahman et al. [6] were solved UFLP by using BDSA. BDSA was tested on UFLP for four transfer methods (bijective, surjective, elitist1 and elitist2). Elitist2 transfer method achieved better solutions than the others. BDSA was compared other population based heuristic algorithms (CPSO and ABC_{bin}) and BDSA obtained better solutions especially for the big scale UFLP. than Nezamabadi-pour et al. [7] introduced a new binary PSO algorithm and called as NBPSO in 2008. NBPSO used a speed-based sigmoid function to convert real-valued variables to binary values. In addition, NBPSO was developed and methods such as Guaranteed Convergence BPSO (GCBPSO) and Improved NBPSO (INBPSO) were proposed. A different rate update equation was proposed for GCBPSO. INBPSO controlled the stagnation of the algorithm and a stagnation control parameter changed the sigmoid function. Guner and Sevkli [8] proposed discrete particle swarm optimization (DPSO) to solve the UFLP. The authors of this study were hybridized the proposed method with a local search mechanism to improve the results. Yuan et al. [9] proposed a newly developed dual PSO (IBPSO) method to solve the unit commitment (UC) problem by integrating lambda iteration method with BPSO. In order to confirm the success of the IBPSO method, other methods in the literature were compared on UC systems with a unit number of 10-100. Experimental results showed that IBPSO was better than other known methods in the literature in terms of lower production cost and shorter calculation time. Saha et al. [10] adapted PSO for binary optimization problems in their studies. The binarization process was performed by taking continuous values. The inertia weight, a special parameter of PSO, was set to decrease in the range of 0.9 - 0.4 depending on the number of iterations. Since it operated with continuous values, it was possible to produce values close to the targeted optimum values. In the study of Bansal and Deep [11], a new modified BPSO (MBPSO) algorithm was proposed for solving the 0-1 knapsack problem (KP) and multidimensional KP (MKP). Compared to the basic BPSO, this improved algorithm proposed a new probability function in the solution of KPs to preserve and make the diversity of the flock more exploratory. The sigmoid function was used to normalize the particle velocity. Beheshti et al. [12] proposed the memetic dual PSO approach. Binary hybrid topology PSO (BHTPSO) was integrated with quadratic interpolation and called as BHTPSO-QI. In this study, 0-1 MKPs were used for comparison. The results were compared with BPSO and binary gravitational search algorithm (BGSA). The success of the proposed method was demonstrated within the framework of convergence speed and solution accuracy. Cinar et

[1] *Department of Electrical and Electronics Engineering, Faculty of Eng., University of Selcuk, Konya, Turkey, RCID ID : 0000-0002-1718-3777*
[2] *Turkish State Meteorological Service, Konya Division, Konya, Turkey ORCID ID : 0000-0001-5596-6767*
* *Corresponding Author Email: asahman@selcuk.edu.tr*

al. [2] proposed XOR logical gate based binary TSA (XORTSA) in their studies. XORTSA used the XOR logic gate to create new binary individuals. XORTSA was compared with BPSO [4] on large scale (250, 500 and 1000 dimensions) binary optimization problems. The test set included five numerical benchmark functions. The results showed that XORTSA produced better solutions than BPSO. Cinar and Kiran [3] developed three different versions of TSA (LogicTSA, SimTSA and SimLogicTSA) in their studies. UFLP was used for performance measurement. Small (16 dimensions), medium (25 dimensions), large (50 dimensions) and very large (100 dimensions) UFLPs were effectively resolved. The experimental results showed that SimLogicTSA was better than SimTSA and LogicTSA. SimLogicTSA compared with artificial bee colony [13, 14], PSO [4, 9, 14] and differential evolution [15] variants. SimLogicTSA proved its success by producing competitive solutions.

TSA which used transfer functions for binarization was not proposed in the literature yet. In this study, four S-shaped and four V-shaped transfer functions which are primitive but also effective methods were used for mapping continuous search space to binary search space.

The remainder of the paper is structured as follows: Section 2 presents a brief introduction to TSA. The transfer functions are described in Section 3. Section 4 discusses the basic principles of the eight different binary versions of TSA. The UFLP is mentioned in Section 5. Section 6 is dedicated for the experimental results and discussions. Finally, Section 7 concludes the work and suggests some directions for future research.

2. Tree-Seed Algorithm

TSA is a nature-inspired, population-based, stochastic, metaheuristic optimization algorithm. TSA was proposed by Kiran [1] based on the relationship between trees and seeds for the solving of continuous optimization problems. Trees and seeds are possible solutions of an optimization problem. TSA is started with a random tree population and during the iterations new seeds are created for each tree. The number of seeds of each tree is determined randomly, but not less than one. Random seeds number should be between 10% and 25% of the total number of trees. Seeds are created with two different mechanisms. These mechanisms are selected with search tendency (ST) parameter. ST is a random number which is between 0 and 1. Two seed production equation (Eq.1 and Eq.2) are given as follows:

$$S_{k,j} = T_{i,j} + \alpha_{i,j}\ x\ (B_j - T_{r,j}) \tag{1}$$

$$S_{k,j} = T_{i,j} + \alpha_{i,j}\ x\ (T_{i,j} - T_{r,j}) \tag{2}$$

where, $S_{k,j}$ is *jth* dimension of the *kth* seed reproduced from *ith* tree , $T_{i,j}$ is the *jth* dimension of *ith* tree, B_j is *jth* dimension of achieved best tree location so far, $T_{r,j}$ is *jth* dimension of the *rth* tree which is selected randomly in population, $\alpha_{i,j}$ is scale factor that is randomly generated in the range of [-1,1]. A greedy selection is carried out between the seeds of each tree in the search process and the best seed is determined. If the solution quality of best seed is better than its own tree, the tree dries and the best seed substitutes that tree. TSA algorithm is executed until the termination criteria are met. The pseudocode of TSA is given in Fig.1. For the detailed information readers can look at these [1-3, 16-20] studies.

1. The initialization of algorithm.
2. Set the parameters (N, ST) of algorithm.
3. Generate trees randomly in search space.
4. Searching with seeds
5. Create new seeds via Eq.1 or Eq.2
6. Selection of best solution
7. Testing termination condition
8. Reporting the best solution

Fig 1. The pseudocode of TSA

In additionally, the flowchart of TSA is given in Fig.2.

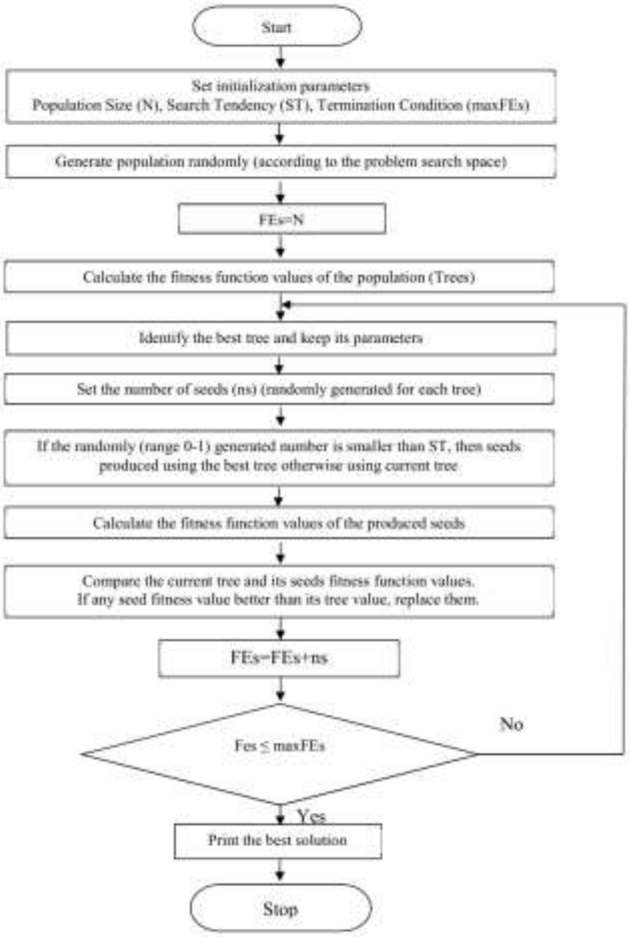

Fig 2. The flowchart of TSA

3. Transfer Functions

Transfer functions are used to map continuous search space to binary search space. Mirjalili and Lewis [21] proposed four S-shaped and four V-shaped transfer functions. The mathematical models of functions are given in Eq.3 to Eq.10.

$$S1(x) = \frac{1}{1 + e^{-2x}} \tag{3}$$

$$S2(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

$$S3(x) = \frac{1}{1 + e^{(-\frac{x}{2})}} \tag{5}$$

$$S4(x) = \frac{1}{1+e^{(-\frac{x}{3})}} \quad (6)$$

$$V1(x) = \left|erf\left(\frac{\sqrt{2}}{\pi}\right)\right| \quad (7)$$

$$V2(x) = |\tanh(x)| \quad (8)$$

$$V3(x) = \left|\frac{(x)}{\sqrt{1+x^2}}\right| \quad (9)$$

$$V4(x) = \left|\frac{2}{\pi} arc\ tan\left(\frac{\pi}{2}x\right)\right| \quad (10)$$

4. Binary Tree-Seed Algorithms

In this study, four S-shaped and four V-shaped transfer functions are carried out to TSA algorithm. These methods are named as TSA1 to TSA8. TSA1 to TSA4 is used S1 (Eq.3) to S4 (Eq.6) as a transfer function respectively. TSA5 to TSA8 is used V1 (Eq.7) to V4 (Eq.10) as a transfer function respectively.

After creating of trees or seeds, the continuous values are sent to transfer function. Then the returned value is compared to a random number which is between 0 and 1. If this returned value is smaller than this random number the binary value is set as 0 otherwise the binary value is set as 1. The four S-shaped transfer functions are shown in Fig 3.

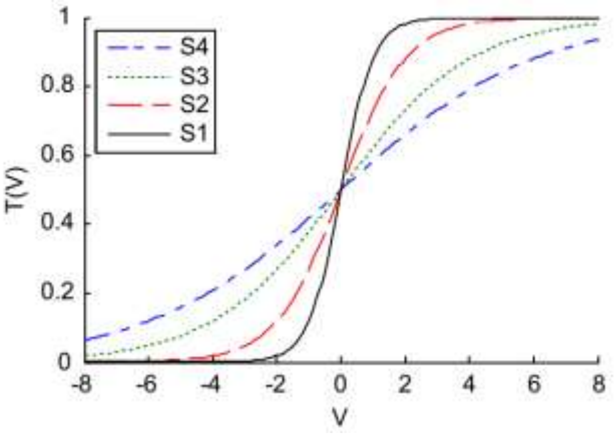

Fig 3. S-shaped transfer functions

The four V-shaped transfer functions are shown in Fig 4.

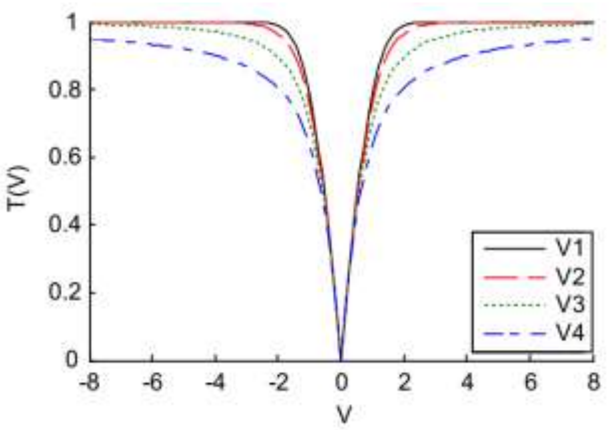

Fig 4. V-shaped transfer functions

As you have seen in Fig.3 and Fig.4, these functions are named according to their shapes.

5. Uncapacitated facility location problem (UFLP)

The UFLP is one of the widely studied and NP-Hard discrete location problem. The UFLP includes locating an undetermined number of facilities to minimize the sum of costs and the variable costs of serving the market demand from these facilities. The mathematical model of the problem is as follows:

$$\min f = \sum_{i\in n} f_i y_i + \sum_{i\in n}\sum_{j\in m} c_{ij}x_{ij} \quad (11)$$

$$\sum_{i\in} x_{ij} = 1, \qquad j = 1,\ldots,m \quad (12)$$

$$x_{ij} \le y_i \qquad i = 1,\ldots,n, j = 1,\ldots,m \quad (13)$$

$$x_{ij}, y_i = 0,1 \quad i = 1,\ldots,n, j = 1,\ldots,m \quad (14)$$

In UFLP, total cost depends on m customers and n facility in a specific location. This problem can be represented as a graph with $(m+n)$ nodes and $m*n$ edges [22]. The cost of opening facility j and the cost of serving to customer i from facility j are represented f_j and c_{ij} respectively. If the i-th facility is open, $y_i = 1$, otherwise $y_i = 0$. If the open facility i is in service to the j-th customer, x_{ij} is 1, otherwise, it will be 0. The main idea of the problem is to minimize the total cost under the condition of satisfying all customers demands. For the test, the problems which are given in the Table 1 are used.

Table 1. The test suite used for comparison of the methods.

Problem name	Problem size	Cost of 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
capA	100x1000	17,156,454.48
capB	100x1000	12,979,071.58
capC	100x1000	11,505,594.33

6. Experimental Results and Discussion

In experiments, population size is taken as 50 and ST is taken as 0.5. In this study, population size and ST analyses have not been studied because our aim is to determine the best transfer function for binary optimization. The continuous search space is limited with -10 and +10. The comparisons of algorithms have made with GAP values. GAP has been calculated as in Eq.15:

$$GAP = \frac{f(sol) - f(opt)}{f(opt)} x100 \quad (15)$$

where $f(opt)$ is the optimum solution of the problem, $f(sol)$ is the mean solution obtained by 30 independent runs of algorithm. The experimental results of low dimensional problems (cap71, cap72, cap73 and cap74) are presented in Table 2, Table 3, Table 4 and Table 5. All methods reached the cap71 problem optimal solution.

TSA1, TSA2, TSA3, TSA4, TSA5 and TSA6 reached the cap72 problem optimal solution. TSA7 solved cap72 problem with 0.01 GAP value and TSA8 solved cap72 problem with 0.04 GAP value. S-shaped transfer functions reached the cap73, cap74, cap101 and cap 102 problems with no GAP. On the other hand, V-shaped transfer functions did not reach the optimal solutions in cap73, cap74, cap101 and cap 102 problems.

As seen in Table 10, Table 11, and Table 12 only TSA1 solved related problem optimally. The cap134 problem solved by TSA1 and TSA2 optimally.

TSA1, TSA2 and TSA3 solved the cap103 problem optimally. The other methods did not reach the optimal solution of cap103 problem. V-shaped transfer functions solved the cap103 problem with about 5% GAP value.

As seen in Table 9, S-shaped transfer functions solved the cap104 problem optimally. V-shaped transfer functions moved away from the optimal solution for cap104 problem.

Table 2. Results of the cap71 problem over 30 independent runs

cap71	Best	Worst	Mean	Std.Dev.	GAP
TSA1	932615.75	932615.75	**932615.75**	0.00	0.00
TSA2	932615.75	932615.75	**932615.75**	0.00	0.00
TSA3	932615.75	932615.75	**932615.75**	0.00	0.00
TSA4	932615.75	932615.75	**932615.75**	0.00	0.00
TSA5	932615.75	932615.75	**932615.75**	0.00	0.00
TSA6	932615.75	932615.75	**932615.75**	0.00	0.00
TSA7	932615.75	932615.75	**932615.75**	0.00	0.00
TSA8	932615.75	932615.75	**932615.75**	0.00	0.00

Table 3. Results of the cap72 problem over 30 independent runs

cap72	Best	Worst	Mean	Std.Dev.	GAP
TSA1	977799.40	977799.40	**977799.40**	0.00	0.00
TSA2	977799.40	977799.40	**977799.40**	0.00	0.00
TSA3	977799.40	977799.40	**977799.40**	0.00	0.00
TSA4	977799.40	977799.40	**977799.40**	0.00	0.00
TSA5	977799.40	977799.40	**977799.40**	0.00	0.00
TSA6	977799.40	977799.40	**977799.40**	0.00	0.00
TSA7	977799.40	978876.30	977871.19	273.22	0.01
TSA8	977799.40	978876.30	978230.16	536.59	0.04

Table 4. Results of the cap73 problem over 30 independent runs

cap73	Best	Worst	Mean	Std.Dev.	GAP
TSA1	1010641.45	1010641.45	**1010641.45**	0.00	0.00
TSA2	1010641.45	1010641.45	**1010641.45**	0.00	0.00
TSA3	1010641.45	1010641.45	**1010641.45**	0.00	0.00
TSA4	1010641.45	1010641.45	**1010641.45**	0.00	0.00
TSA5	1010808.16	1014099.61	1012385.80	1401.73	0.17
TSA6	1010641.45	1014934.15	1012500.48	1527.58	0.18
TSA7	1010808.16	1020176.51	1014685.41	1893.28	0.40
TSA8	1011234.36	1020176.51	1016488.86	2085.56	0.58

Table 5. Results of the cap74 problem over 30 independent runs

cap74	Best	Worst	Mean	Std.Dev.	GAP
TSA1	1034976.98	1034976.98	**1034976.98**	0.00	0.00
TSA2	1034976.98	1034976.98	**1034976.98**	0.00	0.00
TSA3	1034976.98	1034976.98	**1034976.98**	0.00	0.00
TSA4	1034976.98	1034976.98	**1034976.98**	0.00	0.00
TSA5	1048308.16	1063066.66	1055812.93	4029.56	2.01
TSA6	1040641.45	1070132.69	1055870.12	6326.78	2.02
TSA7	1048480.20	1073603.65	1059535.17	6141.50	2.37
TSA8	1048308.16	1070211.60	1060483.60	5114.35	2.46

Table 6. Results of the cap101 problem over 30 independent runs

cap101	Best	Worst	Mean	Std.Dev.	GAP
TSA1	796648.44	796648.44	**796648.44**	0.00	0.00
TSA2	796648.44	796648.44	**796648.44**	0.00	0.00
TSA3	796648.44	796648.44	**796648.44**	0.00	0.00
TSA4	796648.44	796648.44	**796648.44**	0.00	0.00
TSA5	796648.44	800628.88	798300.24	887.57	0.21
TSA6	797582.29	802614.03	798698.11	1152.93	0.26
TSA7	797582.29	802282.89	799830.36	1241.32	0.40
TSA8	797601.59	804275.73	800745.17	1347.69	0.51

Table 7. Results of the cap102 problem over 30 independent runs

cap102	Best	Worst	Mean	Std.Dev.	GAP
TSA1	854704.20	854704.20	**854704.20**	0.00	0.00
TSA2	854704.20	854704.20	**854704.20**	0.00	0.00
TSA3	854704.20	854704.20	**854704.20**	0.00	0.00
TSA4	854704.20	854704.20	**854704.20**	0.00	0.00
TSA5	864914.25	879589.54	872453.54	3857.43	2.08
TSA6	859326.91	879743.93	872523.13	4419.29	2.08
TSA7	865056.75	880097.05	874325.96	3537.27	2.30
TSA8	864853.88	878932.10	874124.43	2799.36	2.27

Table 8. Results of the cap103 problem over 30 independent runs

cap103	Best	Worst	Mean	Std.Dev.	GAP
TSA1	893782.11	893782.11	**893782.11**	0.00	0.00
TSA2	893782.11	893782.11	**893782.11**	0.00	0.00
TSA3	893782.11	893782.11	**893782.11**	0.00	0.00
TSA4	893782.11	894008.14	893789.65	41.27	0.00
TSA5	925713.43	957983.51	944602.51	6821.34	5.69
TSA6	932796.73	955436.98	945306.71	6184.05	5.76
TSA7	915667.70	956131.88	942869.73	8834.85	5.49
TSA8	921898.35	945552.85	936681.65	6743.93	4.80

Table 9. Results of the cap104 problem over 30 independent runs

cap104	Best	Worst	Mean	Std.Dev.	GAP
TSA1	928941.75	928941.75	**928941.75**	0.00	0.00
TSA2	928941.75	928941.75	**928941.75**	0.00	0.00
TSA3	928941.75	928941.75	**928941.75**	0.00	0.00
TSA4	928941.75	928941.75	**928941.75**	0.00	0.00
TSA5	996666.01	1056124.63	1043416.64	13761.95	12.32
TSA6	995979.66	1065748.94	1041895.90	17296.95	12.16
TSA7	1021005.10	1055753.66	1040683.19	10248.81	12.03
TSA8	1015610.93	1047473.25	1028136.02	8113.09	10.68

Table 10. Results of the cap131 problem over 30 independent runs

cap131	Best	Worst	Mean	Std.Dev.	GAP
TSA1	793439.56	793439.56	**793439.56**	0.00	0.00
TSA2	793439.56	794373.41	793571.35	302.03	0.02
TSA3	793439.56	801288.50	797788.64	2122.31	0.55
TSA4	794373.41	810683.10	805127.51	3259.01	1.47
TSA5	891601.18	916713.93	907429.84	5575.12	14.37
TSA6	898159.61	913037.89	905916.90	3860.86	14.18
TSA7	887831.11	909439.28	900130.92	4910.04	13.45
TSA8	878337.90	894012.36	886426.26	4275.26	11.72

Table 11. Results of the cap132 problem over 30 independent runs

cap132	Best	Worst	Mean	Std.Dev.	GAP
TSA1	851495.33	851495.33	**851495.32**	0.00	0.00
TSA2	851495.33	852762.88	851631.49	295.76	0.02
TSA3	852747.03	858999.75	855949.93	1773.01	0.52
TSA4	861964.66	872784.43	866485.60	2949.02	1.76
TSA5	1057455.01	1095022.76	1082008.85	9430.04	27.07
TSA6	1048057.45	1094370.63	1077153.02	10274.12	26.50
TSA7	1047499.94	1080054.83	1067801.93	8670.13	25.40
TSA8	1012660.58	1053884.73	1037366.75	8516.89	21.83

Table 12. Results of the cap133 problem over 30 independent runs

cap133	Best	Worst	Mean	Std.Dev.	GAP
TSA1	893076.71	893732.98	**893098.59**	119.82	0.00
TSA2	893076.71	893782.11	893271.35	283.40	0.02
TSA3	893076.71	900974.43	896445.01	2080.61	0.38
TSA4	901289.05	917933.25	909487.05	5045.37	1.84
TSA5	1237709.98	1275605.56	1257552.68	10125.47	40.81
TSA6	1217527.25	1268329.84	1251393.24	12559.87	40.12
TSA7	1215038.00	1253129.93	1233365.04	10487.55	38.10
TSA8	1166035.63	1209443.45	1188147.70	12570.36	33.04

Table 13. Results of the cap134 problem over 30 independent runs

cap134	Best	Worst	Mean	Std.Dev.	GAP
TSA1	928941.75	928941.75	**928941.75**	0.00	0.00
TSA2	928941.75	928941.75	**928941.75**	0.00	0.00
TSA3	928941.75	948795.81	933979.28	4765.72	0.54
TSA4	937775.05	971803.00	953869.37	10400.48	2.68
TSA5	1453812.61	1537308.68	1504204.38	20261.49	61.93
TSA6	1462565.30	1545216.46	1505288.23	21950.06	62.04
TSA7	1417445.41	1516838.35	1474761.09	21495.80	58.76
TSA8	1351365.20	1445019.10	1409441.48	20233.36	51.73

The extra-large problems (100 dimensions) could not solved optimally with proposed methods. The best results have been obtained by TSA1 for capA, capB and capC problems.

Table 14. Results of the capA problem over 30 independent runs

capA	Best	Worst	Mean	Std.Dev.	GAP
TSA1	17901218.35	19317907.23	**18682847.39**	377856.53	8.90
TSA2	18527186.20	21441080.19	19900075.17	720750.34	15.99
TSA3	21130150.20	28886785.69	25257118.79	2021463.71	47.22
TSA4	26105915.24	38258724.31	34026661.93	2702142.27	98.33
TSA5	137319380.23	148511988.23	143633915.22	2342093.44	737.20
TSA6	139355744.27	145915742.36	142918245.56	1717088.79	733.03
TSA7	131817357.13	141143780.96	137183168.76	2040133.47	699.60
TSA8	120776715.78	127753872.35	125477429.26	1749344.46	631.37

Table 15. Results of the capB problem over 30 independent runs

capB	Best	Worst	Mean	Std.Dev.	GAP
TSA1	13497840.75	14247884.85	**13778283.10**	186079.34	6.16
TSA2	13514010.94	14776855.82	14192575.93	304092.17	9.35
TSA3	14767015.09	16593002.81	15618027.72	414538.36	20.33
TSA4	16352992.73	19959663.97	18510320.30	843438.32	42.62
TSA5	57241168.82	62345327.05	61135404.63	1040397.42	371.03
TSA6	58285705.92	61649079.94	60571877.66	809211.78	366.69
TSA7	55371439.59	59639236.85	57930517.62	1087008.94	346.34
TSA8	52199418.41	54823531.13	53403028.86	661638.01	311.45

Table 16. Results of the capC problem over 30 independent runs

capC	Best	Worst	Mean	Std.Dev.	GAP
TSA1	11850518.62	12361057.46	**12165873.92**	133437.59	5.74
TSA2	12040006.98	12725230.85	12411298.19	168395.95	7.87
TSA3	12799839.02	14112120.13	13499519.44	340487.55	17.33
TSA4	13624719.53	15866700.81	15066661.99	516194.42	30.95
TSA5	43237453.33	45472880.49	44791753.86	492646.45	289.30
TSA6	42267795.28	45075499.62	44247598.78	698780.34	284.57
TSA7	41455001.09	43601611.64	42753834.91	565520.98	271.59
TSA8	36844183.20	40464813.77	39284277.71	825845.31	241.44

The convergence graphs of the problems are shown in between Fig.5 to Fig.19. According to the convergence graphs, the transfer functions are changing similarly. As seen in the figures, S-shaped transfer functions rapidly convergence than V-shaped transfer functions.

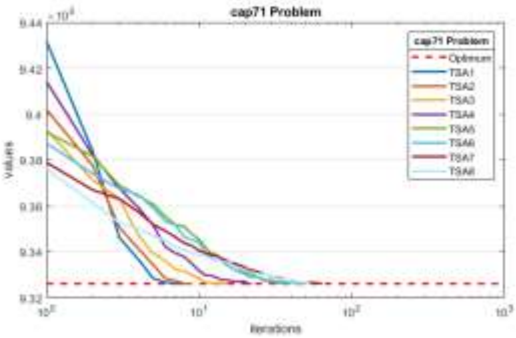

Fig 5. Convergence graph of the cap71 problem

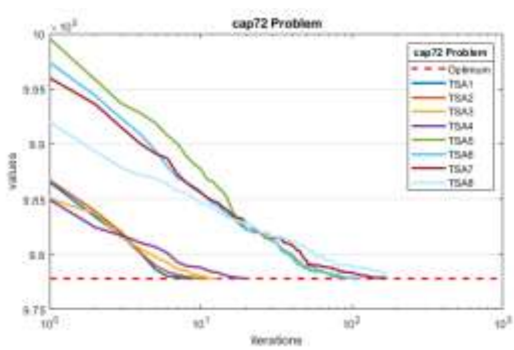

Fig 6. Convergence graph of the cap72 problem

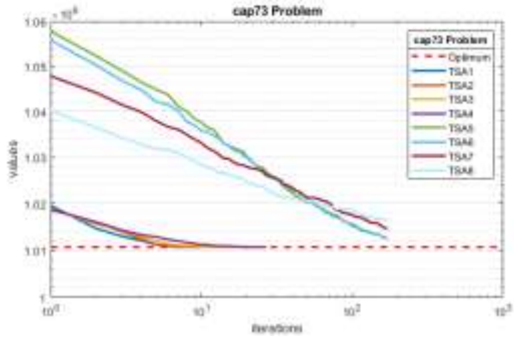

Fig 7. Convergence analyses of the cap73 problem

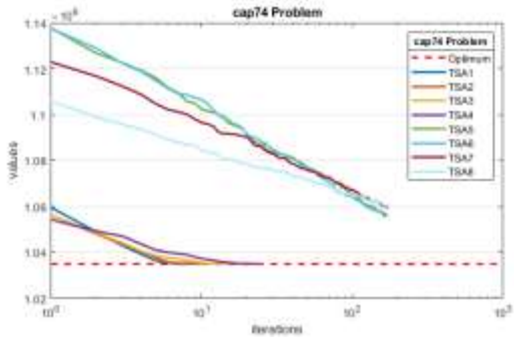

Fig 8. Convergence graph of the cap74 problem

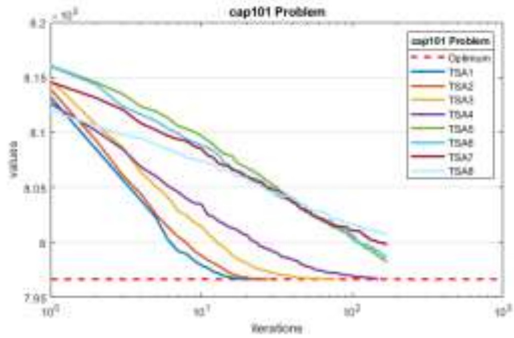

Fig 9. Convergence graph of the cap101 problem

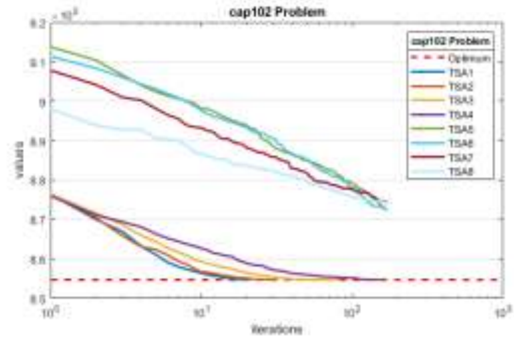

Fig 10. Convergence graph of the cap102 problem

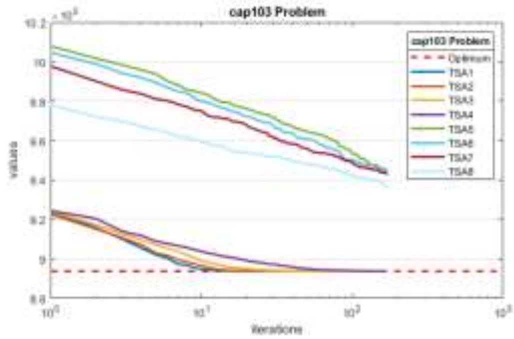

Fig 11. Convergence graph of the cap103 problem

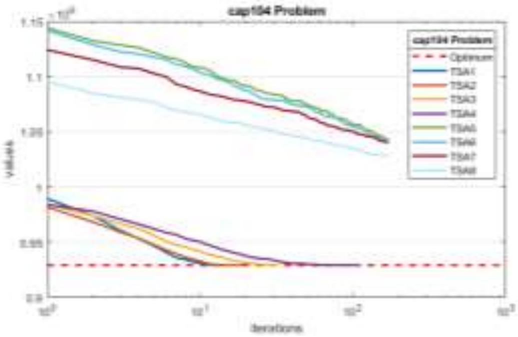

Fig 12. Convergence graph of the cap104 problem

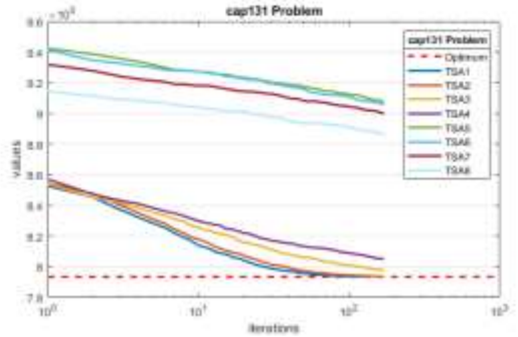

Fig 13. Convergence graph of the cap131 problem

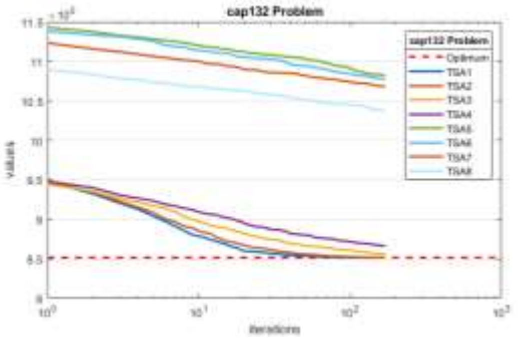

Fig 14. Convergence graph of the cap132 problem

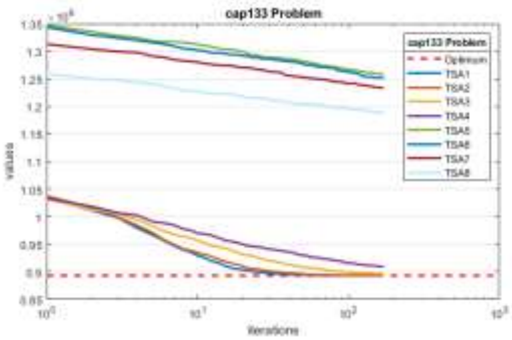

Fig 15. Convergence graph of the cap133 problem

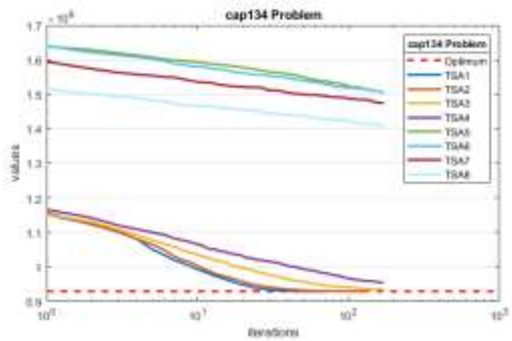

Fig 16. Convergence graph of the cap134 problem

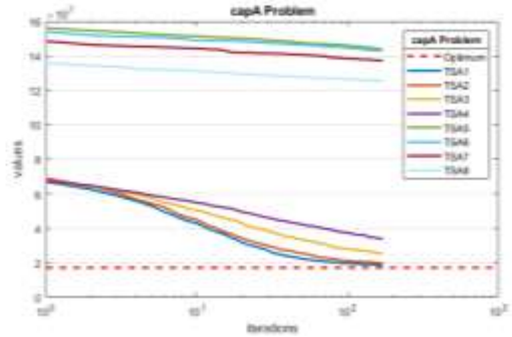

Fig 17. Convergence graph of the capA problem

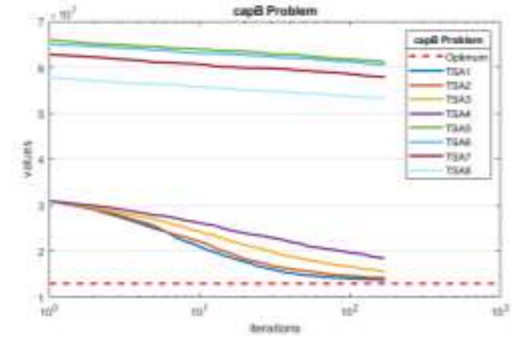

Fig 18. Convergence graph of the capB problem

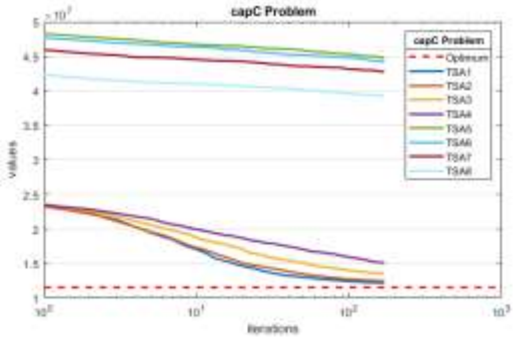

Fig 19. Convergence graph of the capC problem

7. Conclusion

In this study, four S-shaped and four V-shaped transfer function have been applied for UFLP by using TSA. These transfer functions were used in order to convert continuous search space (real coded values) to binary search space (binary coded values). In literature 15 benchmark UFLP which were studied widely have been tried to solve by these proposed transfer functions. According to experimental results, generally S-shaped transfer functions have been reached the optimal solutions for the small and medium scale UFLPs, however, the V-shaped transfer functions have not reached all optimal solutions for small and medium-sized UFLPs. For the big scale UFLP such as capA, capB and capC, TSA1 found the better results than the others. Results have showed that the V-shaped transfer functions are useless for the big scale UFLPs. As a result, S-shaped transfer functions have been more successful than V-shaped transfer functions in terms of reaching the optimal solutions.

8. References

[1] Kiran, M.S., *TSA: Tree-seed algorithm for continuous optimization.* Expert Systems with Applications, 2015. **42**(19): p. 6686-6698.

[2] Cinar, A.C., H. Iscan, and M.S. Kiran, *Tree-Seed algorithm for large-scale binary optimization.* KnE Social Sciences, 2018. **3**(1): p. 48-64.

[3] Cinar, A.C. and M.S. Kiran, *Similarity and logic gate-based tree-seed algorithms for binary optimization.* Computers & Industrial Engineering, 2018. **115**: p. 631-646.

[4] Kennedy, J. and R.C. Eberhart. *A discrete binary version of the particle swarm algorithm*. in *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*. 1997. IEEE.

[5] Sevkli, M. and A.R. Guner. *A continuous particle swarm optimization algorithm for uncapacitated facility location problem*. in *International Workshop on Ant Colony Optimization and Swarm Intelligence*. 2006. Springer.

[6] Sahman, M.A., A.A. Altun, and A.O. Dündar, *The binary differential search algorithm approach for solving uncapacitated facility location problems.* Journal of Computational and Theoretical Nanoscience, 2017. **14**(1): p. 670-684.

[7] Nezamabadi-pour, H., M. Rostami-Shahrbabaki, and M. Maghfoori-Farsangi, *Binary particle swarm optimization: challenges and new solutions.* CSI J Comput Sci Eng, 2008. **6**(1): p. 21-32.

[8] Guner, A.R. and M. Sevkli, *A discrete particle swarm optimization algorithm for uncapacitated facility location problem.* Journal of Artificial Evolution and Applications, 2008. **2008**.

[9] Yuan, X., et al., *An improved binary particle swarm optimization for unit commitment problem.* Expert Systems with applications, 2009. **36**(4): p. 8049-8055.

[10] Saha, S., A. Kole, and K. Dey. *A Modified Continuous Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem*. in *International Conference on Advances in Information Technology and Mobile Communication*. 2011. Springer.

[11] Bansal, J.C. and K. Deep, *A modified binary particle swarm optimization for knapsack problems.* Applied Mathematics and Computation, 2012. **218**(22): p. 11042-11061.

[12] Beheshti, Z., S.M. Shamsuddin, and S. Hasan, *Memetic binary particle swarm optimization for discrete optimization problems.* Information Sciences, 2015. **299**: p. 58-84.

[13] Kashan, M.H., N. Nahavandi, and A.H. Kashan, *DisABC: a new artificial bee colony algorithm for binary optimization.* Applied Soft Computing, 2012. **12**(1): p. 342-352.

[14] Kiran, M.S. and M. Gunduz, *XOR-based artificial bee colony algorithm for binary optimization.* Turkish Journal of Electrical Engineering & Computer Sciences, 2013. **21**(Sup. 2): p. 2307-2328.

[15] Kashan, M.H., A.H. Kashan, and N. Nahavandi, *A novel differential evolution algorithm for binary optimization.* Computational Optimization and Applications, 2013. **55**(2): p. 481-513.

[16] Babalik, A., A.C. Cinar, and M.S. Kiran, *A modification of tree-seed algorithm using Deb's rules for constrained optimization.* Applied Soft Computing, 2018. **63**(Supplement C): p. 289-305.

[17] Cinar, A. and M. Kiran. *A Parallel Version of Tree-Seed Algorithm (TSA) within CUDA Platform*. in *Selçuk International Scientific Conference On Applied Sciences*. 2016.

[18] Cinar, A.C., *Ağaç-tohum aalgoritması için cuda tabanlı bir paralel programlama yaklaşımı*, in *Fen Bilimleri Enstitüsü*. 2016, Selcuk University: Konya, Turkey. p. 102.

[19] Cinar, A.C. and M.S. Kiran. *Boundary conditions in Tree-Seed Algorithm: Analysis of the success of search space limitation techniques in Tree-Seed Algorithm*. in *2017 International Conference on Computer Science and Engineering (UBMK)*. 2017. IEEE.

[20] Cinar, A.C. and M.S. Kiran, *A parallel implementation of Tree-Seed Algorithm on CUDA-supported graphical processing unit.* Journal of the Faculty of Engineering and Architecture of Gazi University, 2018. **33**(4): p. 1397-1409.

[21] Mirjalili, S. and A. Lewis, *S-shaped versus V-shaped transfer functions for binary particle swarm optimization.* Swarm and Evolutionary Computation, 2013. **9**: p. 1-14.

[22] Sun, M., *Solving the uncapacitated facility location problem using tabu search.* Computers & Operations Research, 2006. **33**(9): p. 2563-2589.