A New Binary Arithmetic Optimization Algorithm For Uncapacitated Facility Location Problem

Emine BAŞ (ebas@ktun.edu.tr)
Konya Technical University
Gülnur YILDIZDAN
Selcuk University

Research Article

Keywords: Binary optimization, Arithmetic optimization algorithm, Logic gate

Posted Date: September 27th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2088938/v1

A New Binary Arithmetic Optimization Algorithm For Uncapacitated Facility Location Problem

Emine BAŞ[1*], Gülnur YILDIZDAN[2]

[1] *Department of Software Engineering, Faculty of Engineering and Nature Sciences, Konya Technical University, 42075, Konya, Turkey*

[2]*Kulu Vocational School, Selcuk University, Kulu, Konya, Turkey*

ebas@ktun.edu.tr, gavsar@selcuk.edu.tr

ABSTRACT- Arithmetic Optimization Algorithm (AOA) is a heuristic method developed in recent years. The original version was developed for continuous optimization problems. Its success on different optimization problems has not yet been tested. In this paper, the binary form of AOA (BinAOA) has been proposed. In addition, the candidate solution production scene of BinAOA is developed with the xor logic gate and the BinAOAX method was proposed. Both methods have been tested for success on well-known uncapacitated facility location problems (UFLPs) in the literature. The UFL problem is a binary optimization problem whose optimum results are known. The results of BinAOA and BinAOAX methods were compared and discussed according to best, worst, mean, standard deviation, and gap values. The results of BinAOA and BinAOAX on UFLP are compared with binary heuristic methods used in the literature. As a second application, the performances of BinAOA and BinAOAX algorithms are also tested on classical benchmark functions. The binary forms of AOA, AOAX, Jaya, Tree Seed Algorithm (TSA), and Gray Wolf Optimization (GWO) algorithms were compared in different candidate generation scenarios. The results showed that the binary form of AOA is successful and can be preferred as an alternative binary heuristic method.

Keywords— *Binary optimization, Arithmetic optimization algorithm, Logic gate*

1. Introduction

Binary optimization is a different type of discrete optimization problem. In discrete optimization problems, decision variables are expressed with real values, in binary optimization they are expressed with {0,1}. In the search space, each solution is positioned as binary values. Some of the real-world problems can be easily solved by representing in the binary search space and these problems are named binary optimization problems. The Uncapacitated Facility Location Problem (UFLP) is one of these problems. In the literature, UFLP is often solved by heuristic algorithms (Aslan et al., 2019; Baş and Ülker, 2020a; Çınar and Kiran, 2018).
Abualigah et al. (2021) suggested a new meta-heuristic method called the Arithmetic Optimization Algorithm (AOA) (Abualigah et al., 2021). AOA realizes its exploration and exploitation capabilities with four main arithmetic operators. AOA has improved the ability to exploration with multiplication and division operators and exploitation with addition and subtraction operators. Its success comes from arithmetic operators. AOA was originally developed to solve continuous optimization problems. The success of AOA on discrete or binary optimization problems has not yet been made in the literature. In this paper, AOA has been updated to solve binary optimization problems in terms of structure. Binary AOA (BinAOA), which is a new binary optimization algorithm, has been proposed in the literature. The performance of BinAOA has been tested on UFLPs, a well-known binary optimization problem in the literature. In order to increase the success of BinAOA, logic gates, which are frequently used in binary optimization in the literature, have been added (Aslan et al., 2019; Baş and Ülker, 2020a; Çınar and Kiran, 2018). Logic gates are very suitable for binary optimization due to their structure. There are three types of logic gates in the literature (*and*, *or*, and *xor* logic gates). Since the input and output values of the logic gates are binary values, the candidate solution in binary optimization is successfully applied in the production strategy. Especially because the xor logic gate output values are 50% equal to each other, the xor logic gate is also preferred in this study. This proposed new version of BinAOA is named BinAOAX. As a second test application, the performances of BinAOA and BinAOAX algorithms are also tested on unimodal and multimodal

classical benchmark functions. The binary forms of AOA, AOAX, Jaya, Tree Seed Algorithm (TSA), and Gray Wolf Optimization (GWO) algorithms were compared in different candidate generation scenarios.
In this study, the materials and methods used in the paper are explained in Section 2, and the experimental results of BinAOA and BinAOAX are given and discussed in Section 3. In Section 4, the results are explained.

2. Material and Method

2.1. The Arithmetic Optimization Algorithm (AOA)

Abualigah et al. (2021) suggested a new meta-heuristic method called the Arithmetic Optimization Algorithm (AOA) (Abualigah et al., 2021). The basic structure of AOA consists of four main arithmetic operators used in mathematics (Multiplication (M), Division (D), Subtraction (S), and Addition (A)). These arithmetic operators formed the search mechanism of AOA in the search space. Addition and subtraction operators shaped the local search structure in AOA, while multiplication and division operators shaped the global search structure in AOA.

AOA chooses the exploration or exploitation phase at first. For this selection, the Math Optimizer Accelerated (MOA) function is calculated. Equation 1 shows the MOA function.

$$MOA(iter) = Min + iter \times \left(\frac{Max - Min}{Max\_iter}\right) \tag{1}$$

where *MOA(iter)* denotes the function value at the *t*th iteration, and *iter* denotes the current iteration, and (*Max_ iter*) denotes the maximum number of iterations. *Min* and *Max* denote the minimum and maximum values of the accelerated function, respectively.

2.1.1. Exploration phase

The exploration operators of AOA explore the search area randomly on several regions with Division (D) search strategy and Multiplication (M) search strategy and find a better solution. Equation 2 shows the exploration phase. This phase of searching (exploration search by executing D or M) is conditioned by the Math Optimizer accelerated (MOA) function for the condition of r_1 > MOA (r_1 is a random number). Which of Division (D) search strategy or the Multiplication (M) search strategy to be used is determined by the value of r_2.

$$x_{i,j}(iter+1) = \begin{cases} best(x_j) \div (MOP + \epsilon) \times \left((UB_j - LB_j) \times \mu + LB_j\right), & r_2 < 0.5 \\ best(x_j) \times MOP \times \left((UB_j - LB_j) \times \mu + LB_j\right), & otherwise \end{cases} \tag{2}$$

where r_2 is a random number. $best(x_j)$ is the *j*th position in the best-obtained solution so far. ϵ is a small integer number, UB_j denotes the upper bound value of the *j*th position and LB_j denotes the lower bound value of the *j*th position μ is a control parameter to adjust the search process. Math Optimizer Probability (MOP) is shown by Equation 3.

$$MOP(iter) = 1 - \left(\frac{iter^{1/\alpha}}{Max\_iter^{1/\alpha}}\right) \tag{3}$$

where *MOP(iter)* denotes the function value at the *t*th iteration. α is a sensitive parameter and defines the exploitation accuracy over the iterations (Abualigah et al., 2021).

2.1.2. Exploitation phase

The exploitation operators of AOA are carried out with the Addition (A) search strategy and Subtraction (S) search strategy. In AOA, AOA's exploitation operators search the search area in detail in several local regions. Equation 4 shows the exploitation phase. Which of the Subtraction (S) search strategy or the Addition (A) search strategy to be used is determined by the value of r_3.

$$x_{i,j}(iter+1) = \begin{cases} best(x_j) - MOP \times \left((UB_j - LB_j) \times \mu + LB_j\right), & r_3 < 0.5 \\ best(x_j) + MOP \times \left((UB_j - LB_j) \times \mu + LB_j\right), & otherwise \end{cases} \tag{4}$$

where r_3 is a random number. $best(x_j)$ is the jth position in the best-obtained solution so far. ϵ is a small integer number, UB_j denotes the upper bound value of the jth position and LB_j denotes to lower bound value of the jth position μ is a control parameter to adjust the search process. The Pseudo-code of the AOA has been explained in Algorithm 1.

Algorithm 1: Pseudo-code of the AOA

```
Assign parameter values (α, μ, etc.) of AOA and initialize.
Initialize the solutions'positions (n) randomly. (Solutions: i=1, ..., N)
while (iter < max_iter) do
  Calculate the objective function for the given solutions
  Find the best (best) solution
  Update the MOA and the MOP values with Eq. (1) and Eq. (3)
  for (i=1 to N) do
     for (j=1 to n) do
        Generate a random values between [0, 1] (r1, r2, and r3)
        if r1>MOA then
           if r2>0.5 then
              Update the ith solution positions using Eq. (2) ( Division (D) search strategy)
           else
              Update the ith solution positions using Eq. (2) (Multiplication (M) search strategy)
           end if
        else
           if r3>0.5 then
              Update the ith solution positions using Eq. (4) (Subtraction (S) search strategy)
           else
              Update the ith solution positions using Eq. (4) (Addition (A) search strategy)
           end if
        end if
     end for
  end for
  iter=iter+1
end while
The best solution
```

2.2. Binary Arithmetic Optimization Algorithm (BinAOA)

The original AOA method was originally applied for continuous optimization problems. There is not yet an application in the literature for different types of optimization problems. In this paper, the AOA algorithm has been updated again to solve binary optimization problems. In binary optimization, the search space is expressed in binary structures ({0,1}). Continuous values produced in continuous optimization must be converted into binary values in binary optimization. In the most basic case, this process is performed by Equation 5. They are transfer functions that are frequently preferred in the literature for converting continuous values to binary values. There are various transfer functions in the literatüre (Baş and Ülker, 2020a; Baş and Ülker, 2020b). The most used transfer functions are shown in Table 1. In this study, the success of eight different transfer functions was tested on BinAOA and BinAOAX. Thus, both the success of BinAOA and BinAOAX and the most successful transfer function were determined.

$$x_{i,j} = \begin{cases} 0, & if\ (x_{i,j} < 0.5) \\ 1, & otherwise \end{cases} \quad i=1, 2, \ldots, N; \quad j=1, 2, \ldots, n \tag{5}$$

where $x_{i,j}$ denotes the individual position at the jth dimension of the ith population.

Table 1. S-shaped and V-shaped transfer functions (Beskirli et al., 2018).

S-Shaped		V-Shaped	
Name	**Transfer Functions**	**Name**	**Transfer Functions**

S1	$T(x) = \frac{1}{1+e^{-2x}}$	**V1**	$T(x) = \left\lvert erf\left(\frac{\sqrt{\pi}}{2}x\right)\right\rvert$
S2	$T(x) = \frac{1}{1+e^{-x}}$	**V2**	$T(x) = \lvert tanh(x)\rvert$
S3	$T(x) = \frac{1}{1+e^{(-x/2)}}$	**V3**	$T(x) = \left\lvert (x)\big/\sqrt{1+x^2}\right\rvert$
S4	$T(x) = \frac{1}{1+e^{(-x/3)}}$	**V4**	$T(x) = \left\lvert\frac{2}{\pi}\right\rvert arc\ tan\left(\frac{\pi}{2}x\right)$

Logic gates were used to generate new candidates in BinAOA. The proposed new method is called BinAOAX. Logic gates are often used in the literature to obtain binary values. It is often preferred because of the 50% same value of the xor gate output values (Aslan et al., 2019; Baş and Ülker, 2020a; Baş and Ülker, 2020b; Çınar and Kiran, 2018). In BinAOAX, new candidate solutions are produced according to Equation 6.

$$x_{new,j} = \begin{cases} x_{ij} \oplus \left(x_j^r \oplus x_{best,j}\right), \text{if} \left(rand_{ij} < 0.5\right) \\ x_{ij}, \quad otherwise \end{cases} \tag{6}$$

where x_{newj} is the *j*th dimension of the new candidate solution produced for the *i*th solution, x_{ij} is the *j*th dimension of *i*th current solution, x_j^r is the *j*th dimension of random neighbor solution. The Pseudo-codes of the BinAOA and BinAOAX have been explained in Algorithm 2 and Algorithm 3, respectively.

Algorithm 2: Pseudo-code of the BinAOA

```
Assign parameter values (α, μ, etc.) of BAOA and initialize.
Create the binary solutions using Eq. (5).
while (iter < max_iter) do
  Calculate the fitness function for the UFL problem
  Find the best (best) solution
  Update the MOA and the MOP values with Eq. (1) and Eq. (3)
  for (i=1 to N) do
     for (j=1 to n) do
        Generate a random values between [0, 1] (r1, r2, and r3)
        if r1>MOA then
           if r2>0.5 then
              Update xnew using Eq. (2)
           end if
        else
           if r3>0.5 then
              Update xnew using Eq. (4)
           end if
        end if
     end for
  end for
Transfer function selection
Continuous solutions are transformed into new binary solutions (xnew) using the transfer function
Calculate fitness values of new candidate solutions (xnew) for the UFL problem
Compare the fitness values of new candidate solutions and existing solutions
  iter=iter+1
end while
The best solution
```

Algorithm 3: Pseudo-code of the BinAOAX

```
Assign parameter values (α, μ, etc.) of BAOA and initialize.
Create the binary solutions using Eq. (5).
while (iter < max_iter) do
```

```
    Calculate the fitness function for the UFL problem
    Find the best (best) solution
    Update the MOA and the MOP values with Eq. (1) and Eq. (3)
    for (i=1 to N) do
       for (j=1 to n) do
           Generate a random values between [0, 1] (r1, r2, and r3)
           if r1>MOA then
              if r2>0.5 then
                   Update xnewx using Eq. (2)
              end if
           else
              if r3>0.5 then
                 Update xnewx using Eq. (4)
              end if
           end if
       end for
   end for
  Transfer function selection
  Continuous solutions are transformed using the transfer function
  Create new binary candidate solutions (xnewx) for BinAOAX using Eq. (6)
  Calculate fitness values of new candidate solutions for the UFL problem
  Compare the fitness values of new candidate solutions and existing solutions
   iter=iter+1
  end while
  The best solution
```

2.3. The Uncapacitated Facility Location Problem (UFLP)

BinAOA's performance is studied on UFL Problems. Since UFLPs are very appropriate for binary optimizations and easily applicable for binary structures, we have preferred UFLPs as benchmark problems. In basic UFLP formulation, UFLP includes a set of *(I)* potential facilities. Each facility can be open or closed. In BAOA, if the facility is open '1', and if it is closed '0' values are selected. These facilities serve a set of *(J)* customers. The objective function of this problem is to minimize the sum of the shipment costs between *I* and *J* and the opening costs of the facilities (Çınar and Kiran, 2018). The mathematical structure of the UFLP is shown below.

$$f(UFLP) = min\{\sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i\} \quad (7)$$

where $I = \{1,2,...,k\}$ is the set of possible facility locations, $J = \{1,2,...,g\}$ is the set of customer demand points, f_i is the fixed cost of opening a facility in $i \in I$, c_{ij} is the shipment cost between ith facility location and jth customer point. The decision variable x_{ij} is the demand of customer *j* corresponded ith facility and y_i is the binary variable: $y_i = 1$ if a facility is located in $i \in I$, $y_i = 0$ otherwise.

Subject to:

$$\sum_{i\in I} x_{ij} = 1 \quad j \in J \quad (8)$$

$$x_{ij} \leq y_{i}, \; {}_{i\in I, j\in J} \quad (9)$$

$$x_{ij} \in \{0,1\}, i \in I, \; j \in J \quad (10)$$

$$y_i \in \{0,1\}, i \in I \quad (11)$$

While Equation 8 provides to satisfy all demands of customers, Equation 9 ensures that a customer can be served from a facility only if a facility is opened. Equations 10 and 11 define the decision variables in the binary structure (Çınar and Kiran, 2018).

3. Experimental Analysis

AOA method has been updated in this paper to solve binary optimization problems. As a result, Binary AOA (BinAOA) was proposed. BinAOA has been developed and a new candidate solution strategy has been added. The developed BinAOA is named BinAOAX. In the new candidate solution strategy added to BinAOA, the xor logic gate is used. New candidate solutions produced according to the xor logic gate are included in the system. BinAOA and BinAOAX methods have been tested on UFLP, a well-known binary optimization problem in the literature. UFLP has twelve low and high-dimensional data sets. These data sets are shown in Table 2. Twenty independent studies were conducted for all experiments. All experiments were run on a windows 7 operating system, 4gb ram, and 2.3Ghz processor environment. The Gap value determined how close the current result is to the optimum result. The Gap value is calculated by Equation 12. The best shows the value of the best fitness, the worst shows the value of the worst fitness, the mean the value shows value of the average fitness, and the std shows the standard deviation of the fitness value.

$$Gap = \frac{fitness(solution) - optimum}{optimum} \times 100 \tag{12}$$

Table 2. The data sets for UFL problems

Problem Name	Size	Optimum
Cap71	16 x 50	932615,7500
Cap72	16 x 50	977799,4000
Cap73	16 x 50	1010641,4500
Cap74	16 x 50	1034976,9800
Cap101	25 x 50	796648,4400
Cap102	25 x 50	854704,2000
Cap103	25 x 50	893782,1100
Cap104	25 x 50	928941,7500
Cap131	50 x 50	793439,5600
Cap132	50 x 50	851495,3300
Cap133	50 x 50	893076,7100
Cap134	50 x 50	928941,7500

3.1. Parameter setups for BinAOA

Different population sizes (pop={10, 20, 30, 40, 50, 60, 70, 80, 90}) have been tested in the BinAOA for transfer function=S1 and maximum evaluation=8.00E+04. BinAOA has been tested on twelve different UFLP. Test results are shown in Table 3. According to the results, as the population size increases, the optimum result is approached more. BinAOA produced more successful results, especially when the population size was 80. However, increasing the population size too much increases the time it takes for the method to work, and also prevents the success of the method from being understood adequately. The parameter settings used in this study are shown in Table 4. μ, α, Min, and Max parameter values were used as determined in the original paper (Abualigah et al., 2021). Thus, fair comparisons could be made in the literature.

Table 3. Different means of the population sizes for BinAOA for S=1

ID	BinAOA								
	Pop=10	*Pop=20*	*Pop=30*	*Pop=40*	*Pop=50*	*Pop=60*	*Pop=70*	*Pop=80*	*Pop=90*
71	**932615,8**	**932615,8**	**932615,8**	**932615,8**	**932615,8**	**932615,8**	**932615,8**	**932615,8**	**932615,8**
72	**977799,4**	**977799,4**	**977799,4**	**977799,4**	**977799,4**	**977799,4**	**977799,4**	**977799,4**	**977799,4**
73	1010897,17	**1010641,45**	**1010641,45**	**1010641,45**	**1010641,45**	**1010641,45**	**1010641,45**	**1010641,45**	**1010641,45**
74	1039607,11	1037393,92	1035471,895	1035524,995	1035524,995	1035250,985	1035250,985	**1034976,975**	**1034976,975**
101	798834,7338	797967,0625	797298,6363	797772,9175	797791,775	797194,035	**797025,8375**	**797025,8375**	**797025,8375**
102	859762,23	858837,41	858765,62	857614,4075	859036,91	857498,7025	858059,02	**856840,4375**	856891,4175
103	907512,3375	904841,535	904910,8375	902201,86	902643,0375	903786,7475	902868,5225	**899149,815**	900218,2525
104	961453,7625	957387,7825	953344,2263	952503,32	950687,6363	949898,045	949328,3838	950521,7613	**945358,3875**
131	840855,0213	836403,28	836349,3788	834511,2913	835634,4013	833349,7275	833452,185	**833043,98**	834321,1313
132	943130,1038	936985,3338	935604,7263	930438,7225	927859,435	928580,5113	925765,8038	928867,73	**924182,1575**
133	1031529,218	1017725,224	1016103,306	1016014,518	1003908,886	1012950,283	**1000362,451**	1005067,814	1004087,429
134	1168812,003	1137814,88	1135041,369	1133629,371	1116165,786	1130133,69	1125929,625	**1113058,54**	1113751,104

Table 4. Parameter setups for BinAOA and BinAOAX

Methods	Population size (N)	Maximum evaluation	μ	α	*Min*	*Max*	*r1, r2, and r3*
BinAOA	40	8,00E+04	0,5	5	0,2	1	[0,1]
BinAOAX	40	8,00E+04	0,5	5	0,2	1	[0,1]

3.2. The Comparisons of the BinAOA and BinAOAX on UFLPs

3.2.1. The results of BinAOA on S and V-shaped transfer functions

The performance of BinAOA has been successfully tested on eight different transfer functions. As parameter settings, population number and maximum evaluation number were selected as 40 and 8,00E+04, respectively. Experiment results for BinAOA are shown in Tables 5-12. The convergence graphs of the transfer functions S1, S2, S3, S4, V1, V2, V3, and V4 for twelve different UFLP datasets are shown in Figure 1. Eight different transfer functions are compared according to six different comparison criteria. These are the best, mean, the worst, standard deviation (Std), gap, and CPU time.

The success of BinAOA has been thoroughly tested in four S-shaped and four V-shaped transfer functions. It has been shown that V-shaped transfer functions give more successful results than S-shaped transfer functions. According to the Gap and Std results, the most successful transfer functions were V1 and V2, while the most unsuccessful transfer function was S4. V1 and V2 transfer functions reached optimum results in 7 out of 12 benchmark datasets (71, 72, 73, 74, 102, 104, and 134). According to Figure 1, V-shaped transfer functions converged to optimum results faster than S-shaped transfer functions.

Table 5. The results of BinAOA on UFL Problems for S1.

ID	BinAOA					
	Best	*Mean*	*Worst*	*Std*	*Gap*	*CPU Time*
71	932615,75	932615,8	932615,75	0	0	0,1292
72	977799,4	977799,4	977799,4	0	0	0,1461
73	1010641,45	1010641,45	1010641,45	0	0	0,1152
74	1034976,975	1035524,995	1037717,075	1124,5	0,0529	0,0294
101	797582,2875	797772,9175	798535,4375	391,2	0,1412	0,0386
102	854704,2	857614,4075	860547,375	1953,7	0,3405	0,1725
103	899428,5625	902201,86	904515,0625	1813,7	0,9420	0,1828
104	949595,9375	952503,32	956310,125	2444,0	2,5364	0,2102
131	829938,425	834511,2913	843833,325	4452,06	5,1764	0,3745
132	916028,4	930438,7225	939473,125	8108,6	9,2711	0,3278
133	983923,4125	1016014,518	1029223,625	12387,3	13,7656	0,3851
134	1110162,013	1133629,371	1149222,088	13553,9	22,0345	0,3165

Table 6. The results of BinAOA on UFL Problems for S2.

ID	BinAOA					
	Best	*Mean*	*Worst*	*Std*	*Gap*	*CPU Time*
71	932615,75	932615,8	932615,75	0	0	0,1392
72	977799,4	977799,4	977799,4	0	0	0,1261
73	1010641,45	1010649,786	1010808,163	37,28	0,0008	0,1049
74	1034976,975	1035936,01	1037717,075	1340,90	0,0927	0,0995
101	796648,4375	798713,3969	799914,25	843,37	0,2592	0,2186
102	854704,2	859011,9656	862507,25	1955,39	0,5040	0,1943
103	897440,6125	901003,3019	906402,4875	2960,93	0,8079	0,1778
104	939030,825	949161,4175	956035,7375	5132,33	2,1766	0,2008
131	824672,3625	831100,9688	839150,6	3747,64	4,7466	0,3699
132	901434,35	920487,9031	931896,9875	7917,89	8,1025	0,3368
133	966397,675	995860,0744	1012662,4	10124,34	11,5089	0,3371
134	1047488,588	1099294,859	1134420,938	24630,64	18,3384	0,3361

Table 7. The results of BinAOA on UFL Problems for S3.

ID	BinAOA					
	Best	*Mean*	*Worst*	*Std*	*Gap*	*CPU Time*
71	932615,75	932615,8	932615,75	0	0	0,1477
72	977799,4	977853,245	978876,3	234,70	0,0055	0,1405
73	1010641,45	1010666,457	1010808,163	59,53	0,0025	0,1160
74	1034976,975	1035113,98	1037717,075	597,19	0,0132	0,1103
101	797657,275	800536,5506	803091,575	1267,03	0,4881	0,2322
102	854704,2	859701,9219	863064,9375	2270,49	0,5847	0,1983
103	898800,0375	902659,695	906485,525	2157,28	0,9933	0,1820
104	944168,3	950772,2194	962475,5	4331,34	2,3500	0,1473
131	822311,4375	830496,8688	836373,1875	3646,85	4,6705	0,2914
132	903529,85	915021,2556	925101,9875	5038,91	7,4605	0,3243
133	964312,3375	982250,0463	999685,4375	11059,79	9,9850	0,2909
134	1052245,363	1091590,101	1116783,088	19655,36	17,5090	0,2815

Table 8. The results of BinAOA on UFL Problems for S4.

ID	BinAOA					
	Best	*Mean*	*Worst*	*Std*	*Gap*	*CPU Time*
71	932615,75	932663,4075	933568,9	207,73	0,0051	0,1197
72	977799,4	977907,09	978876,3	323,07	0,0110	0,0986
73	1010641,45	1010658,121	1010808,163	50,01	0,0016	0,1004
74	1034976,975	1035524,995	1037717,075	1096,04	0,0529	0,1133
101	799335,1625	801212,5331	803138,8625	928,90	0,5729	0,2078
102	856767,0625	860472,1794	863642,4875	2091,03	0,6749	0,1757
103	897558,6625	902370,5788	906771,6875	2727,91	0,9609	0,1600
104	932985,325	946473,3088	956468,7625	6724,14	1,8873	0,1370
131	822176,9875	830485,3356	835913,8	3915,01	4,6690	0,3026
132	905280,9875	913476,2956	920965,825	5104,19	7,2791	0,2644
133	951846,6	983071,25	999685,4375	10520,15	10,0769	0,3136
134	1056171,138	1087346,608	1103423,063	12946,10	17,0522	0,2705

Table 9. The results of BinAOA on UFL Problems for V1.

ID	BinAOA					
	Best	*Mean*	*Worst*	*Std*	*Gap*	*CPU Time*
71	932615,75	932615,75	932615,75	0	0	0,1151
72	977799,4	977799,4	977799,4	0	0	0,0956
73	1010641,45	1010641,45	1010641,45	0	0	0,0598
74	1034976,98	1034976,98	1034976,98	0	0	0,0537
101	796648,4375	797325,5775	799144,6875	722,27	0,0850	0,1579
102	854704,2	854704,2	854704,2	0	0	0,1395
103	893782,1125	893866,9688	894801,1625	228,88	0,0095	0,1012
104	928941,75	928941,75	928941,75	0	0	0,0675
131	794299,85	796458,8144	799304,6625	1369,80	0,3805	0,1675
132	851495,325	851925,6088	855005,5625	797,27	0,0505	0,1613
133	893076,7125	893629,3175	894801,1625	569,29	0,0619	0,1320
134	928941,75	928941,75	928941,75	0	0	0,0930

Table 10. The results of BinAOA on UFL Problems for V2.

ID	BinAOA					
	Best	*Mean*	*Worst*	*Std*	*Gap*	*CPU Time*
71	932615,75	932615,75	932615,75	0	0	0,1101
72	977799,4	977799,4	977799,4	0	0	0,0861
73	1010641,45	1010641,45	1010641,45	0	0	0,0584
74	1034976,98	1034976,98	1034976,98	0	0	0,0537
101	796648,4375	797402,6925	799103,6	814,06	0,0947	0,1395
102	854704,2	854704,2	854704,2	0	0	0,1115
103	893782,1125	893951,825	895027,1875	336,75	0,0190	0,0902
104	928941,75	928941,75	928941,75	0	0	0,0738
131	794299,85	796997,2213	799155,6125	1299,95	0,4484	0,1505
132	851495,325	852073,13	854166,6375	641,64	0,0679	0,1244
133	893076,7125	893558,7775	894801,1625	589,34	0,0540	0,1047
134	928941,75	928941,75	928941,75	0	0	0,0686

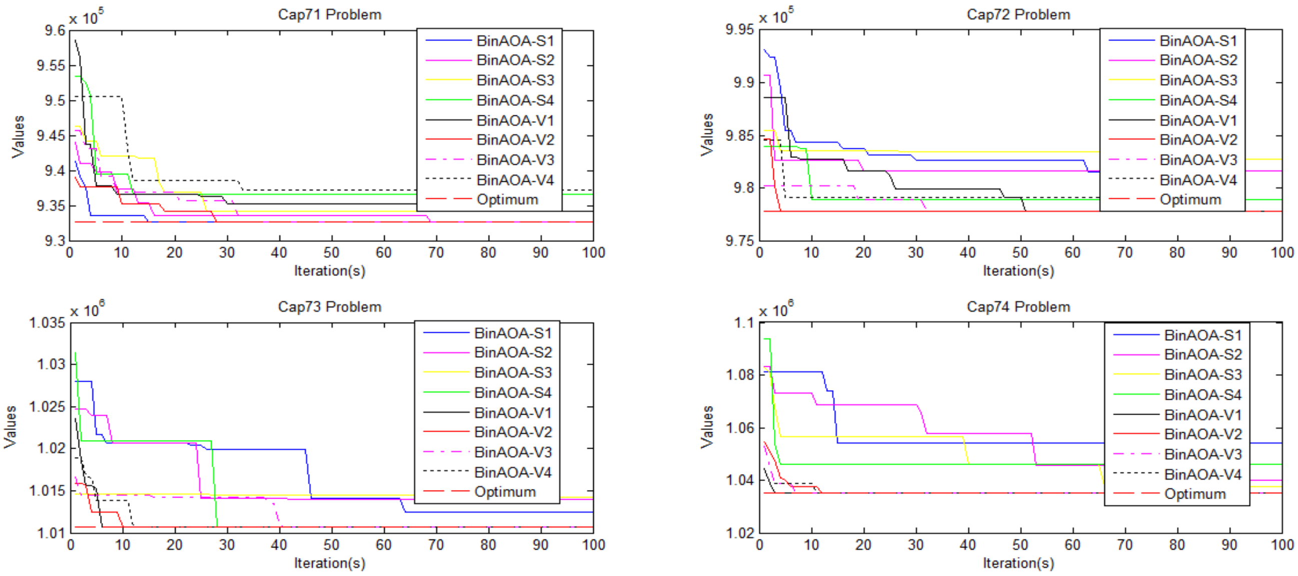
Table 11. The results of BinAOA on UFL Problems for V3.

ID	BinAOA					
	Best	*Mean*	*Worst*	*Std*	*Gap*	*CPU Time*
71	932615,75	932615,75	932615,75	0	0	0,1065
72	977799,4	977799,4	977799,4	0	0	0,0906
73	1010641,45	1010641,45	1010641,45	0	0	0,0550
74	1034976,98	1034976,98	1034976,98	0	0	0,0483
101	796648,4375	798468,9313	802364,4875	1361,98	0,2285	0,1287
102	854704,2	854742,3325	855466,85	166,22	0,0045	0,1114
103	893782,1125	894246,2388	895027,1875	485,57	0,0519	0,0744
104	928941,75	928941,75	928941,75	0	0	0,0558
131	796550,9875	800196,3663	803265,8625	1926,46	0,8516	0,1610

132	851495,325	852547,0688	855054,5875	944,99	0,1235	0,1366
133	893076,7125	893766,4925	894801,1625	595,21	0,0772	0,1138
134	928941,75	928941,75	928941,75	0	0	0,0742

Table 12. The results of BinAOA on UFL Problems for V4.

ID	BinAOA					
	Best	***Mean***	***Worst***	***Std***	***Gap***	***CPU Time***
71	932615,75	932884,0019	934199,1375	540,13	0,0288	0,1009
72	977799,4	977799,4	977799,4	0	0	0,0970
73	1010641,45	1010641,45	1010641,45	0	0	0,0596
74	1034976,98	1034976,98	1034976,98	0	0	0,0515
101	797508,725	800141,875	803632,0375	1599,30	0,4385	0,1332
102	854704,2	855023,2506	856004,4125	456,59	0,0373	0,1066
103	893782,1125	894382,0475	895027,1875	498,84	0,0671	0,0768
104	928941,75	928941,75	928941,75	0	0	0,0557
131	796152,15	801548,4163	805865,775	2383,77	1,0220	0,1535
132	852151,5875	854174,6388	856941,8625	1545,83	0,3147	0,1201
133	893076,7125	893771,305	894801,1625	615,68	0,0778	0,1074
134	928941,75	928941,75	928941,75	0	0	0,0810



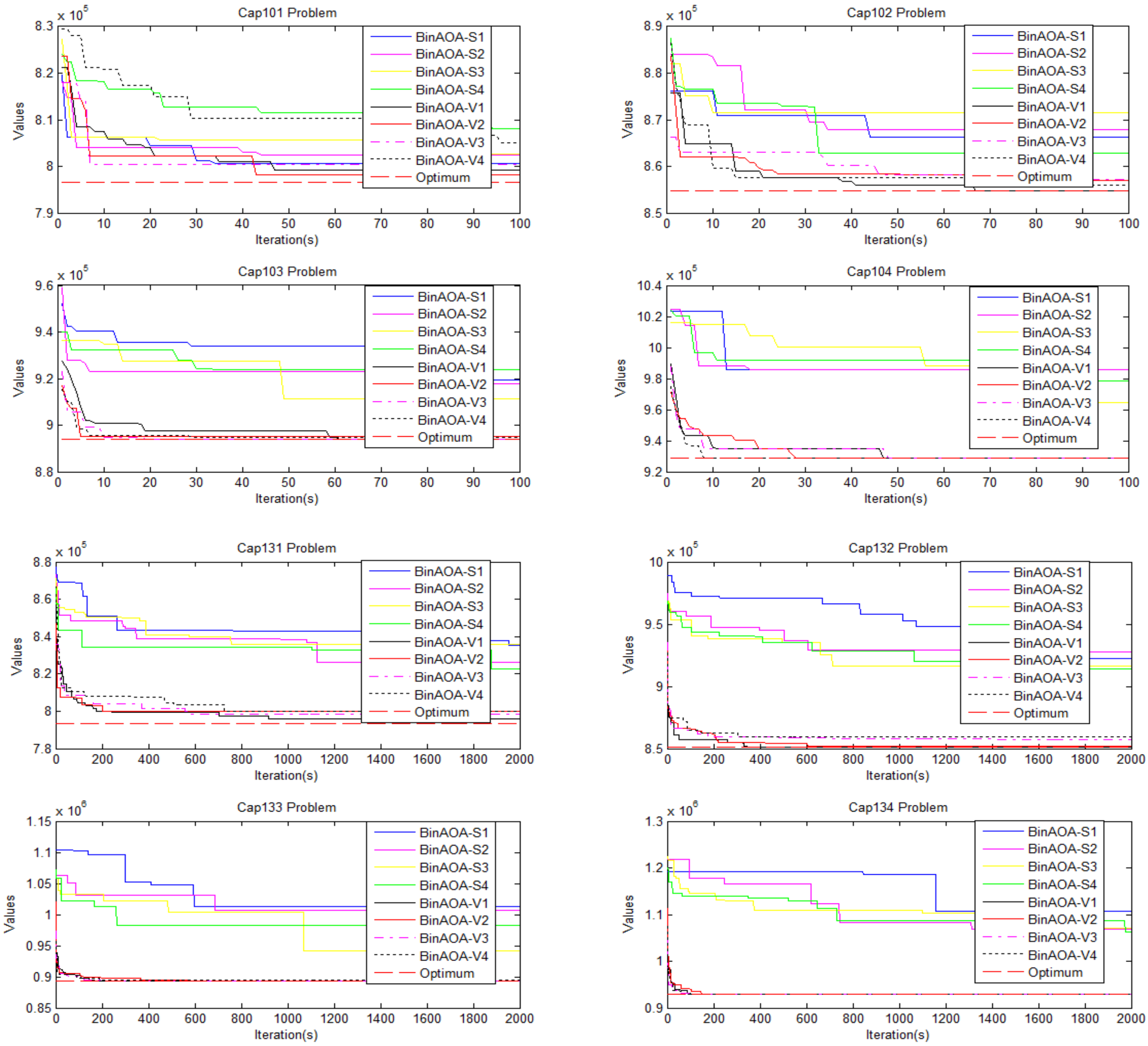


Figure 1. The convergence graphs of the BinAOA for twelve different UFLP data sets

3.2.2. The results of BinAOAX on S and V-shaped transfer functions

The performance of BinAOAX has been successfully tested on eight different transfer functions. As parameter settings, population number and maximum evaluation number were selected as 40 and 8,00E+04, respectively. Experiment results for BinAOAX are shown in Tables 13-20. The convergence graphs of the transfer functions S1, S2, S3, S4, V1, V2, V3, and V4 for twelve different UFLP data sets are shown in Figure 2. Eight different transfer functions are compared according to six different comparison criteria. These are the best, mean, the worst, standard deviation (Std), gap, and CPU time.

The success of BinAOAX has been thoroughly tested in four S-shaped and four V-shaped transfer functions. It has been shown that V-shaped transfer functions give more successful results than S-shaped transfer functions. According to the Gap and Std results, the most successful transfer functions were V1 and V2, while the most unsuccessful transfer function was S4. V1 and V2 transfer functions reached optimum results in 12 out of 12 benchmark datasets (71, 72, 73, 74, 102, 103, 104, 131, 132, 133, and 134).

Table 13. The results of BinAOAX on UFL Problems for S1.

ID	BinAOAX					
	Best	***Mean***	***Worst***	***Std***	***Gap***	***CPU Time***
71	932615,75	932615,75	932615,75	0	0	0,1486

72	977799,4	977799,4	977799,4	0	0	0,1238
73	1010641,45	1010641,45	1010641,45	0	0	0,1282
74	1034976,98	1034976,98	1034976,98	0	0	0,1073
101	796648,4375	797557,2063	799144,6875	689,62	0,1141	0,2036
102	854704,2	858518,2144	861850,7875	1995,97	0,4462	0,1647
103	893782,1125	902398,1569	908183,2375	3669,14	0,9640	0,1622
104	932007,9625	950505,7913	964093,05	8679,63	2,3214	0,1553
131	831163,1875	837227,1675	844537,2375	3639,80	5,5187	0,3855
132	914447,9125	929423,1069	941688,75	7516,95	9,1519	0,3329
133	996954,875	1014535,522	1033251,663	9542,86	13,6000	0,3092
134	1094624,388	1129905,497	1159764,825	17038,31	21,6336	0,3230

Table 14. The results of BinAOAX on UFL Problems for S2.

ID	**BinAOAX**					
	Best	***Mean***	***Worst***	***Std***	***Gap***	***CPU Time***
71	932615,75	932615,75	932615,75	0	0	0,1324
72	977799,4	977799,4	977799,4	0	0	0,1238
73	1010641,45	1010641,45	1010641,45	0	0	0,1008
74	1034976,975	1035524,995	1037717,08	1096,04	0,0529	0,1029
101	797508,725	798812,9906	800527,34	921,47	0,2717	0,2014
102	855466,85	858818,5425	862379,04	2088,25	0,4814	0,2086
103	893782,1125	901452,3094	907101,375	3731,63	0,8582	0,1856
104	930026,55	947366,0313	957732,45	7294,99	1,9834	0,1762
131	827763,7	832685,075	839864,725	3637,02	4,9463	0,5913
132	902781,15	920465,8331	933066,575	7525,36	8,0999	0,3951
133	978194,65	999305,3419	1012022,51	9799,77	11,8947	0,4015
134	1052082,313	1100257,473	1129869,05	21471,48	18,4420	0,3967

Table 15. The results of BinAOAX on UFL Problems for S3.

ID	**BinAOAX**					
	Best	***Mean***	***Worst***	***Std***	***Gap***	***CPU Time***
71	932615,75	932615,75	932615,75	0	0	0,1252
72	977799,4	977799,4	977799,4	0	0	0,1084
73	1010641,45	1010679,431	1011234,363	132,38	0,0038	0,0909
74	1034976,975	1035662	1037717,075	1186,50	0,0662	0,0827
101	797508,725	800159,5206	801959,25	1138,93	0,4407	0,1800
102	855466,85	859041,7181	862429,05	1872,01	0,5075	0,1575
103	894008,1375	901547,1681	907761,7625	3310,37	0,8688	0,1393
104	934650,3	946568,99	959928,775	6536,92	1,8976	0,1733
131	823145,4375	831019,51	836188,9	3693,40	4,7363	0,3779
132	900140,525	915623,57	923588,2	5713,89	7,5312	0,3557
133	978750,55	992054,8656	1008248,4	7258,62	11,0828	0,3441
134	1064072,3	1089013,726	1112509,663	12877,86	17,2316	0,3586

Table 16. The results of BinAOAX on UFL Problems for S4.

ID	**BinAOAX**					
	Best	***Mean***	***Worst***	***Std***	***Gap***	***CPU Time***
71	932615,75	932615,75	932615,75	0	0	0,1598
72	977799,4	977853,245	978876,3	234,70	0,0055	0,1701
73	1010641,45	1010679,431	1011067,65	102,07	0,0038	0,1118
74	1034976,975	1035500,42	1038841,375	1262,84	0,0506	0,1115
101	798243,3125	800436,2031	802513,0375	1177,64	0,4755	0,2382
102	857555,1375	860375,2656	863866,325	1718,95	0,6635	0,2310
103	896993,0875	902081,9788	906301,525	2370,64	0,9286	0,1789
104	936507,5	949626,5513	959391,2625	5500,13	2,2267	0,2009
131	822501	830952,4094	836442,2875	3573,90	4,7279	0,3790
132	899483,85	914055,3513	927896,8125	7055,31	7,3471	0,3446
133	970313,2	985239,99	1001528,9	9625,60	10,3197	0,3652
134	1048731,975	1084621,239	1111667,663	15861,28	16,7588	0,4154

Table 17. The results of BinAOAX on UFL Problems for V1.

ID	BinAOAX					
	Best	***Mean***	***Worst***	***Std***	***Gap***	***CPU Time***
71	932615,75	932615,75	932615,75	0	0	0,1353
72	977799,4	977799,4	977799,4	0	0	0,1030
73	1010641,45	1010641,45	1010641,45	0	0	0,0691
74	1034976,98	1034976,98	1034976,98	0	0	0,0539
101	796648,4375	796648,4375	796648,4375	0	0	0,1535
102	854704,2	854704,2	854704,2	0	0	0,1237
103	893782,11	893782,11	893782,11	0	0	0,0952
104	928941,75	928941,75	928941,75	0	0	0,0657
131	793439,56	793439,56	793439,56	0	0	0,1772
132	851495,33	851495,33	851495,33	0	0	0,1591
133	893076,71	893076,71	893076,71	0	0	0,1280
134	928941,75	928941,75	928941,75	0	0	0,0925

Table 18. The results of BinAOAX on UFL Problems for V2.

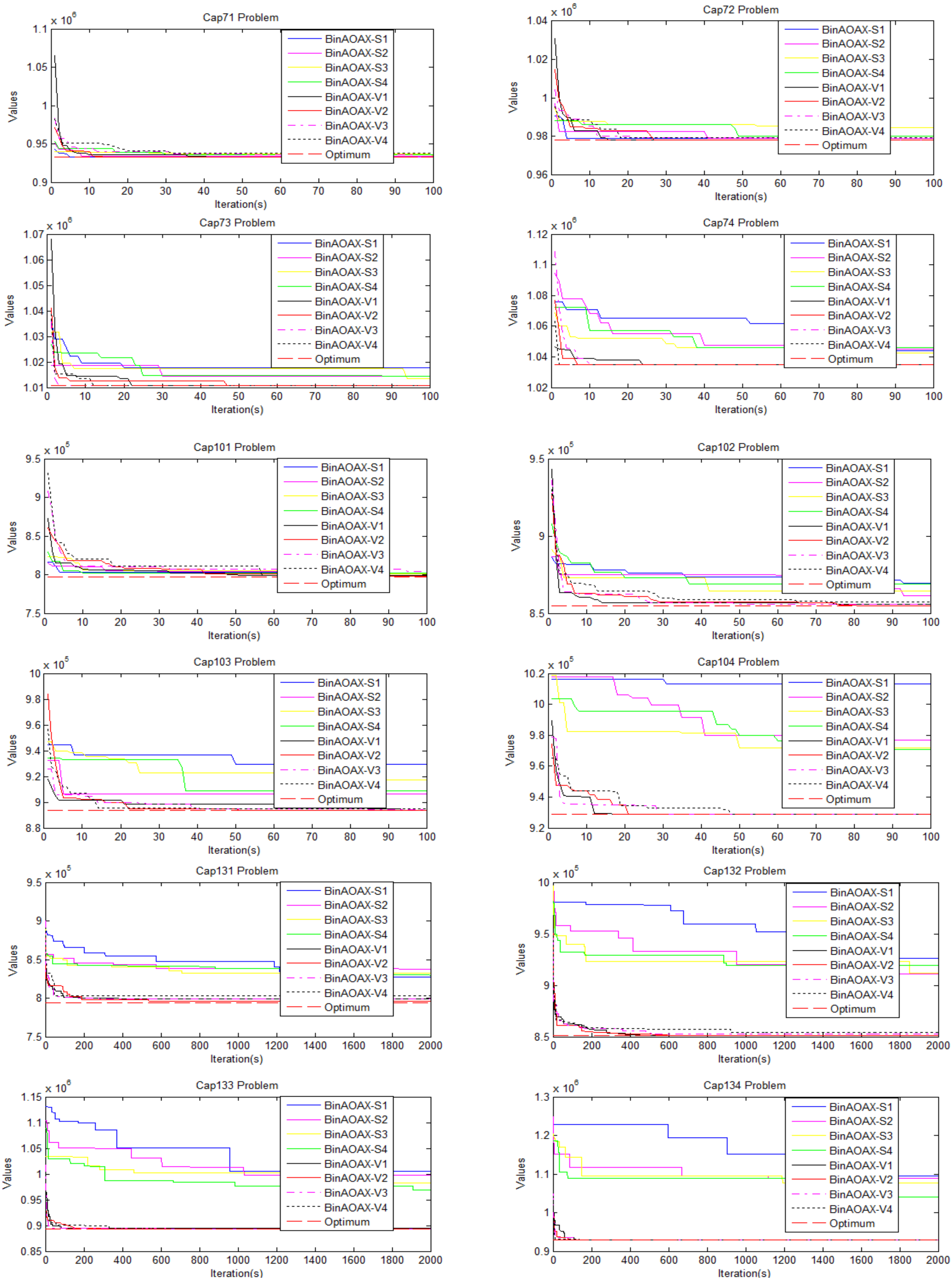
ID	BinAOAX					
	Best	***Mean***	***Worst***	***Std***	***Gap***	***CPU Time***
71	932615,75	932615,75	932615,75	0	0	0,1315
72	977799,4	977799,4	977799,4	0	0	0,0926
73	1010641,45	1010641,45	1010641,45	0	0	0,0640
74	1034976,98	1034976,98	1034976,98	0	0	0,0528
101	796648,4375	796648,4375	796648,4375	0	0	0,1530
102	854704,2	854704,2	854704,2	0	0	0,1262
103	893782,11	893782,11	893782,11	0	0	0,0927
104	928941,75	928941,75	928941,75	0	0	0,0564
131	793439,56	793439,56	793439,56	0	0	0,1625
132	851495,33	851495,33	851495,33	0	0	0,1207
133	893076,71	893076,71	893076,71	0	0	0,1048
134	928941,75	928941,75	928941,75	0	0	0,0729

Table 19. The results of BinAOAX on UFL Problems for V3.

ID	BinAOAX					
	Best	***Mean***	***Worst***	***Std***	***Gap***	***CPU Time***
71	932615,75	932615,75	932615,75	0	0	0,1161
72	977799,4	977799,4	977799,4	0	0	0,0967
73	1010641,45	1010641,45	1010641,45	0	0	0,0578
74	1034976,98	1034976,98	1034976,98	0	0	0,0509
101	796648,4375	798127,8688	799920,8125	781,83	0,1857	0,1402
102	854704,2	854767,5775	855971,75	276,26	0,0074	0,1196
103	893782,1125	894183,985	895027,1875	484,99	0,0450	0,0971
104	928941,75	928941,75	928941,75	0	0	0,0545
131	796255,6625	798757,4256	803759,775	1798,34	0,6702	0,1869
132	851495,325	852414,2981	855041,725	931,83	0,1079	0,1315
133	893076,7125	893625,4125	894801,1625	534,52	0,0614	0,1101
134	928941,75	928941,75	928941,75	0	0	0,0817

Table 20. The results of BinAOAX on UFL Problems for V4.

ID	BinAOAX					
	Best	***Mean***	***Worst***	***Std***	***Gap***	***CPU Time***
71	932615,75	932615,75	932615,75	0	0	0,1085
72	977799,4	977864,4106	979099,6125	283,37	0,0066	0,0937
73	1010641,45	1010641,45	1010641,45	0	0	0,0597
74	1034976,98	1034976,98	1034976,98	0	0	0,0473
101	797508,725	800509,9588	803277,35	1462,78	0,4847	0,1341
102	854704,2	854983,485	855971,75	446,09	0,0327	0,1100
103	893782,1125	894240,4913	895027,1875	495,42	0,0513	0,0843
104	928941,75	928941,75	928941,75	0	0	0,0521
131	797735,5375	802427,8825	808941,2875	2747,02	1,1328	0,1460
132	851495,325	853936,7125	857627,3875	1744,50	0,2867	0,1536
133	893076,7125	893807,5994	894801,1625	548,22	0,0818	0,1146
134	928941,75	928941,75	928941,75	0	0	0,0817



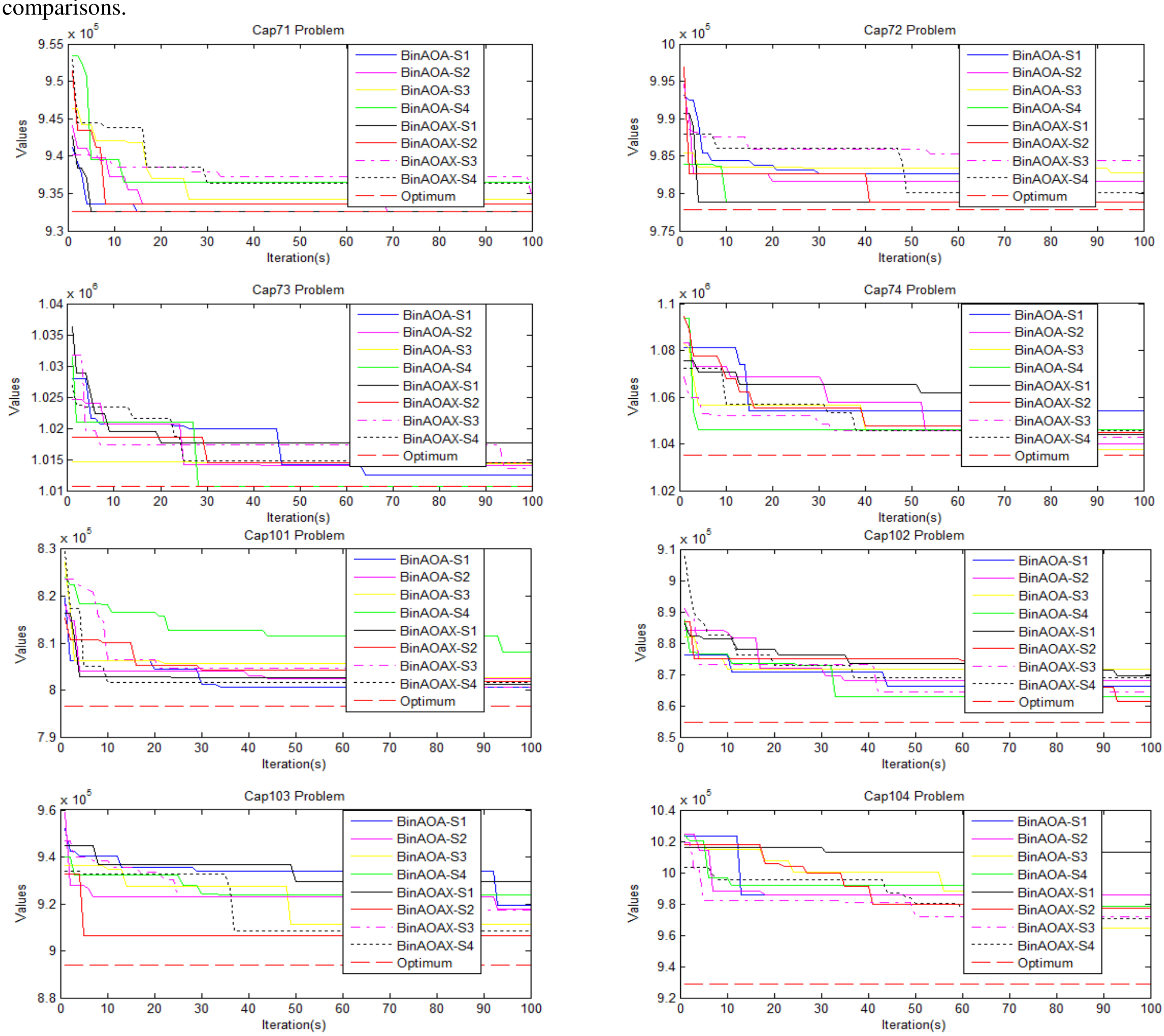
Cap71 Problem
Cap72 Problem
Cap73 Problem
Cap74 Problem
Cap101 Problem
Cap102 Problem
Cap103 Problem
Cap104 Problem
Cap131 Problem
Cap132 Problem
Cap133 Problem
Cap134 Problem
BinAOAX-S1
BinAOAX-S2
BinAOAX-S3
BinAOAX-S4
BinAOAX-V1
BinAOAX-V2
BinAOAX-V3
BinAOAX-V4
Optimum
Values
Iteration(s)

Figure 2. The convergence graphs of the BinAOAX for twelve different UFLP data sets

3.2.3. The comparison of BinAOA and BinAOAX results on S and V-shaped transfer functions

The success of BinAOA and BinAOAX algorithms on twelve different UFL problems in eight different transfer functions were compared. Comparison results are shown in Figure 3 and Figure 4. According to the comparison results, it has been proven that BinAOAX achieves better results than BinAOA. BinAOAX converged to optimum results faster than BinAOA. The new candidate generation method developed using the xor logic gate has increased the success of BinAOA.

Statistical test results are shown in Table 21. The confidence interval of the Wilcoxon Signed-Rank test results of the BinAOA and BinAOAX is 0.05 in Table 21. According to the results, BinAOA and BinAOAX produced similar results in different transfer functions. Among the transfer functions, V-shaped transfer functions produced better results than S-shaped transfer functions. Therefore, V1 and V2 transfer functions were preferred in literature comparisons.



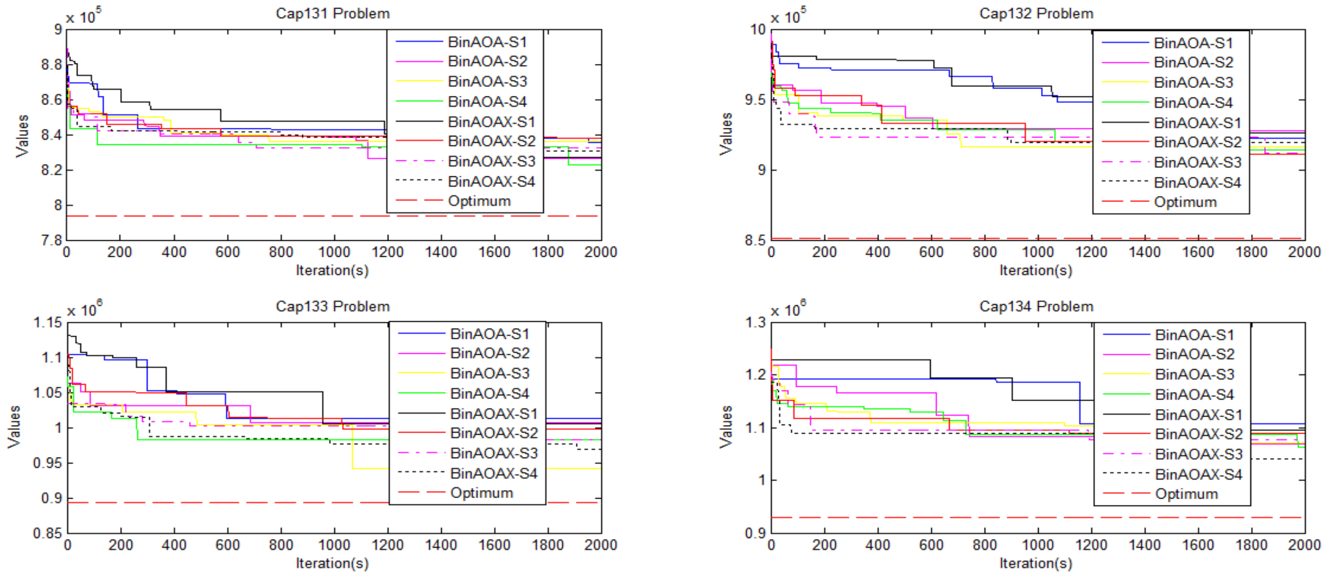
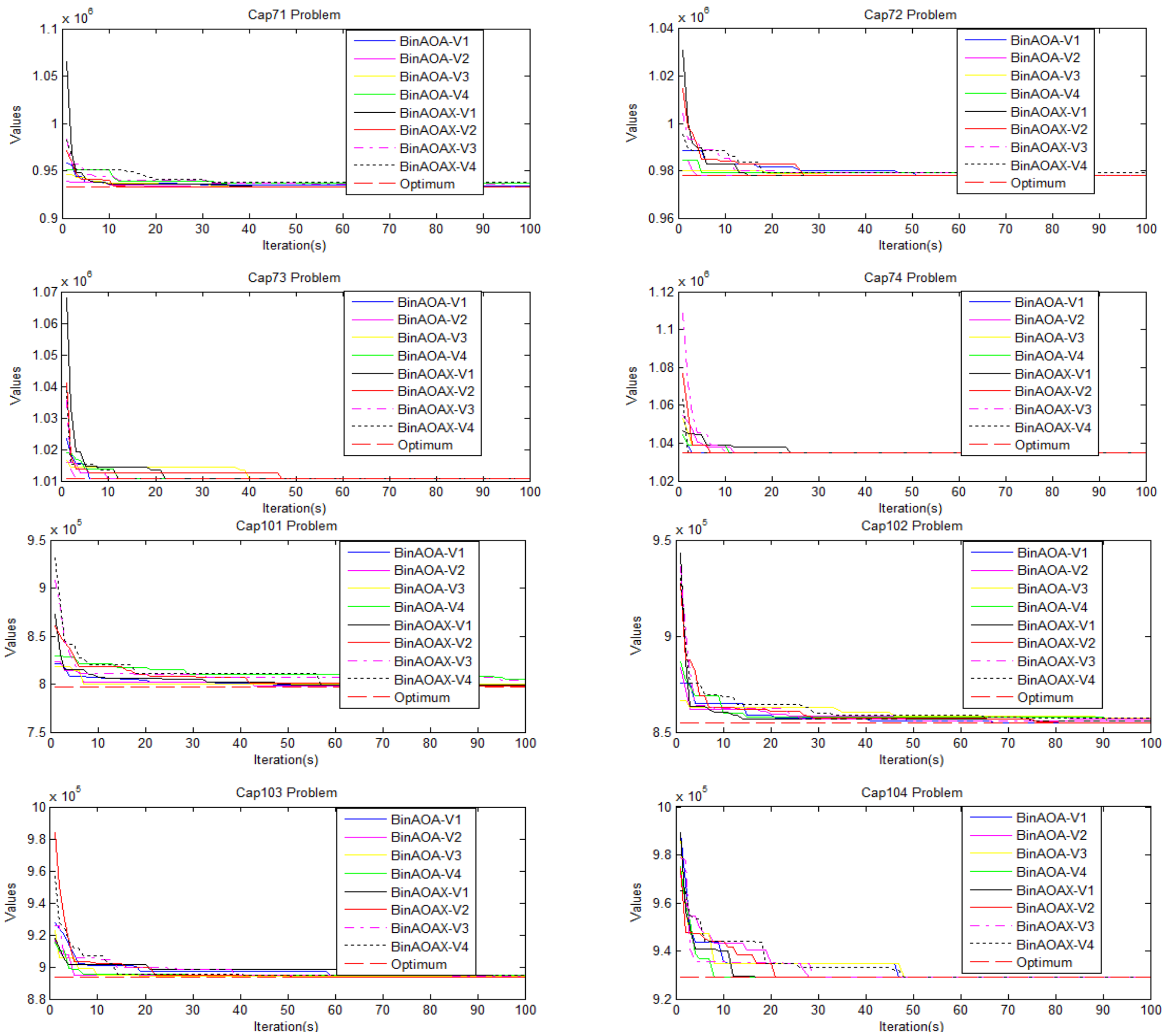


Figure 3. The convergence graphs of the BinAOA and BinAOAX for twelve different UFLP data sets (S-Shape transfer functions)



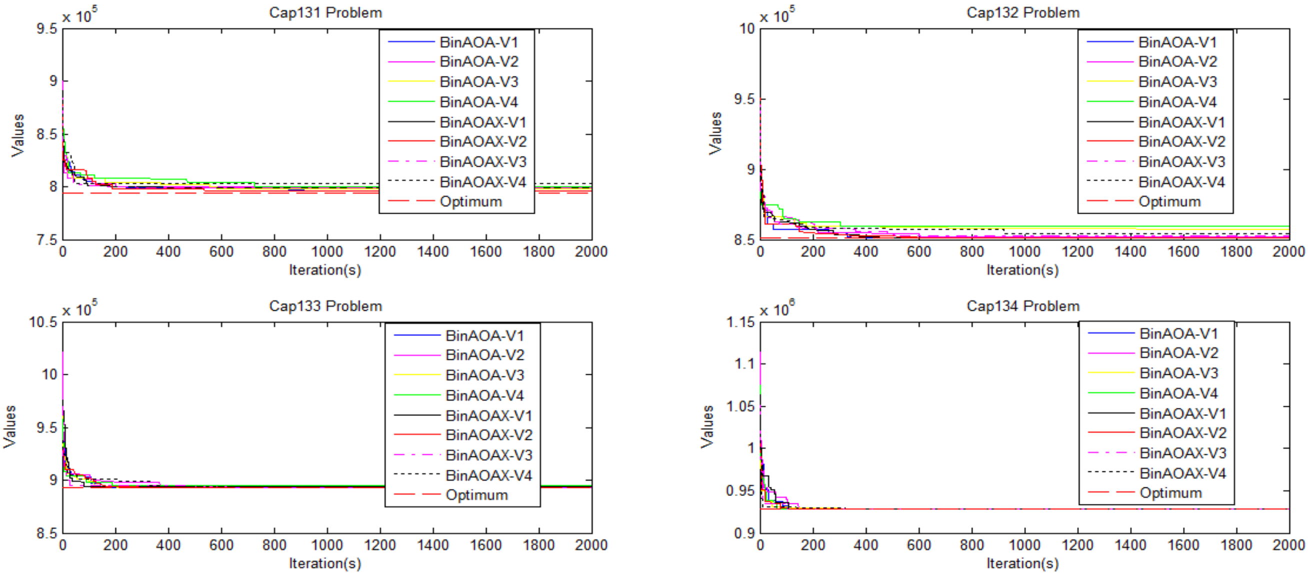


Figure 4. The convergence graphs of the BinAOA and BinAOAX for twelve different UFLP data sets (V-Shape transfer functions)

Table 21. The statistical test results of BinAOA with BinAOAX (S-shaped and V-shaped transfer functions).

Cap_ID	S1	S2	S3	S4	V1	V2	V3	V4
	p	p	p	p	p	p	p	p
71	1	1	1	1	1	1	1	0,125
72	7,74E-06	1	1	1	1	1	1	1
73	1	1	1	0,75	1	1	1	1
74	0,21726	0,453125	0,21875	0,671875	7,74E-06	1	1	1
101	0,235116	0,97022	0,27071	0,044208	0,000488	0,000488	0,513142	0,350405
102	0,085897	0,793839	0,313463	0,708905	1	1	1	0,755859
103	0,575486	0,681322	0,501591	0,82276	0,125	0,015625	0,740723	0,470642
104	0,411465	0,370261	0,016881	0,061953	1	1	1	1
131	0,067355	0,100458	0,681322	0,851925	8,77E-05	8,79E-05	0,026318	0,247145
132	0,65415	0,881293	0,331723	0,708905	0,001953	0,000488	0,571243	0,736875
133	0,525653	0,20433	0,003592	0,765198	0,000977	0,003906	0,492676	0,886146
134	0,350656	0,97022	0,681322	0,601213	1	1	1	1

3.2.4. The comparison of BinAOA and BinAOAX results with TSA, ISS, and BinSSA

The BinAOAX and BinAOA have been compared to TSA, JayaX, ISS, and BinSSA for twelve UFL problems. Comparison results have been obtained from various sources (Çınar and Kiran, 2018; Baş and Ülker, 2020a; Hakli and Ortacay, 2019). All algorithms were run under similar conditions to ensure a fair comparison of results. In the comparison process, the population size is determined as 40 and the maximum evolution as 8E+04 equally. The comparison results are shown in Table 22.

According to the results, BinAOAX results are quite good. Like TSA, ISS, BinSSA, BinAOAX also achieved optimum results and its rank value was 1. BinAOA, on the other hand, failed to pass TSA, ISS, BinSSA, and BinAOAX in 7 of 12 benchmark datasets.

Table 22. The comparison results of BinAOA, BinAOAX, TSA, ISS, and BinSSA.

Cap_ID	TSA		ISS		BinSSA		BinAOA		BinAOAX	
	Gap	Rank	Gap	Rank	Gap	Rank	Gap	Rank	Gap	Rank
71	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
72	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
73	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
74	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
101	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	0,0850	2	**0,0E+00**	**1**
102	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
103	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	0,0095	2	**0,0E+00**	**1**
104	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
131	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	0,3805	2	**0,0E+00**	**1**

132	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	0,0505	2	**0,0E+00**	**1**
133	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	0,0619	2	**0,0E+00**	**1**
134	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**

3.2.5. The comparison of BinAOA and BinAOAX results with BJA, JayaX, and JayaX-LSM

The BinAOAX and BinAOA have been compared to BJA, JayaX, and JayaX-LSM for twelve UFL problems. Comparison results have been obtained from (Aslan et al., 2019). All algorithms were run under similar conditions to ensure a fair comparison of results. In the comparison process, the population size is determined as 40 and the maximum evolution as 8E+04 equally. The comparison results are shown in Table 23.
According to the results, BinAOAX results are quite good. Like JayaX, JayaX-LSM, BinAOAX also achieved optimum results and its rank value was 1. BinAOAX has passed BJA on all benchmark datasets.

Table 23. The comparison results of BinAOA, BinAOAX, BJA, JayaX, and JayaX_LSM.

Cap_ID	BJA		JayaX		JayaX-LSM		BinAOA		BinAOAX	
	Gap	Rank	Gap	Rank	Gap	Rank	Gap	Rank	Gap	Rank
71	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
72	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
73	0,01211	2	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
74	0,04412	2	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
101	0,01800	2	**0,0E+00**	**1**	**0,0E+00**	**1**	0,0850	3	**0,0E+00**	**1**
102	0,01509	2	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
103	0,02153	3	**0,0E+00**	**1**	**0,0E+00**	**1**	0,0095	2	**0,0E+00**	**1**
104	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
131	0,14290	2	**0,0E+00**	**1**	**0,0E+00**	**1**	0,3805	3	**0,0E+00**	**1**
132	0,11215	3	**0,0E+00**	**1**	**0,0E+00**	**1**	0,0505	2	**0,0E+00**	**1**
133	0,13623	3	**0,0E+00**	**1**	**0,0E+00**	**1**	0,0619	2	**0,0E+00**	**1**
134	0,02459	2	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**

3.2.6. The comparison of BinAOA and BinAOAX results with BPSO, and CPSO

The BinAOAX and BinAOA have been compared to BPSO and CPSO for twelve UFL problems. Comparison results have been obtained from (Kashan et al., 2013; Korkmaz and Kiran, 2018). All algorithms were run under similar conditions to ensure a fair comparison of results. In the comparison process, the population size is determined as 40 and the maximum evolution as 8E+04 equally. The comparison results are shown in Table 24.
According to the results, BinAOAX results are quite good. BinAOAX has passed BPSO, CPSO, and BinAOA on all benchmark datasets.

Table 24. The comparison results of BinAOA, BinAOAX, BPSO, and CPSO.

Cap_ID	BPSO		CPSO		BinAOA		BinAOAX	
	Gap	Rank	Gap	Rank	Gap	Rank	Gap	Rank
71	**0,0E+00**	**1**	5,0E-02	2	**0,0E+00**	**1**	**0,0E+00**	**1**
72	**0,0E+00**	**1**	7,0E-02	2	**0,0E+00**	**1**	**0,0E+00**	**1**
73	2,42E-02	2	6,0E-02	3	**0,0E+00**	**1**	**0,0E+00**	**1**
74	8,82E-03	2	7,0E-02	3	**0,0E+00**	**1**	**0,0E+00**	**1**
101	4,32E-02	2	1,4E-01	4	0,0850	3	**0,0E+00**	**1**
102	9,89E-03	2	1,5E-01	3	**0,0E+00**	**1**	**0,0E+00**	**1**
103	4,94E-02	3	1,6E-01	4	0,0095	2	**0,0E+00**	**1**
104	4,05E-02	2	1,8E-01	3	**0,0E+00**	**1**	**0,0E+00**	**1**
131	1,71E-01	2	7,5E-01	4	0,3805	3	**0,0E+00**	**1**
132	5,83E-02	3	7,8E-01	4	0,0505	2	**0,0E+00**	**1**
133	8,29E-02	3	7,3E-01	4	0,0619	2	**0,0E+00**	**1**
134	1,95E-01	2	8,9E-01	3	**0,0E+00**	**1**	**0,0E+00**	**1**

3.2.7. The comparison of BinAOA and BinAOAX results with DisDE, and BinDE

The BinAOAX and BinAOA have been compared to DisDE and BinDE for twelve UFL problems. Comparison results have been obtained from (Kashan et al., 2013). All algorithms were run under similar conditions to ensure

a fair comparison of results. In the comparison process, the population size is determined as 40 and the maximum evolution as 8E+04 equally. The comparison results are shown in Table 25.
According to the results, BinAOAX results are quite good. BinAOAX has passed DisDE, BinDE, and BinAOA on all benchmark datasets.

Table 25. The comparison results of BinAOA, BinAOAX, DisDE, and BinDE.

Cap_ID	DisDE		BinDE		BinAOA		BinAOAX	
	Gap	Rank	Gap	Rank	Gap	Rank	Gap	Rank
71	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
72	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
73	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
74	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
101	7,2E-03	2	**0,0E+00**	**1**	0,0850	3	**0,0E+00**	**1**
102	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
103	8,4E-04	2	**0,0E+00**	**1**	0,0095	3	**0,0E+00**	**1**
104	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
131	**0,0E+00**	**1**	3,6E-03	2	0,3805	3	**0,0E+00**	**1**
132	**0,0E+00**	**1**	5,0E-03	2	0,0505	3	**0,0E+00**	**1**
133	1,5E-02	3	1,4E-02	2	0,0619	4	**0,0E+00**	**1**
134	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**

3.2.8. The comparison of BinAOA and BinAOAX results with ABC_{Bin}, DisABC, and BinABC

The BinAOAX and BinAOA have been compared to ABC_{Bin}, DisABC, and BinABC for twelve UFL problems. Comparison results have been obtained from (Kiran and Gunduz, 2013; Kiran, 2015). All algorithms were run under similar conditions to ensure a fair comparison of results. In the comparison process, the population size is determined as 40 and the maximum evolution as 8E+04 equally. The comparison results are shown in Table 26.
According to the results, BinAOAX results are quite good. BinAOAX has passed ABC_{Bin}, DisABC, BinABC, and BinAOA on all benchmark datasets.

Table 26. The comparison results of BinAOA, BinAOAX, ABC_{Bin}, DisABC, and BinABC.

Cap_ID	ABC_{Bin}		DisABC		BinABC		BinAOA		BinAOAX	
	Gap	Rank	Gap	Rank	Gap	Rank	Gap	Rank	Gap	Rank
71	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
72	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
73	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
74	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
101	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	0,0850	2	**0,0E+00**	**1**
102	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
103	5,1E-03	2	**0,0E+00**	**1**	**0,0E+00**	**1**	0,0095	3	**0,0E+00**	**1**
104	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**
131	2,0E-01	2	6,2E-01	4	**0,0E+00**	**1**	0,3805	3	**0,0E+00**	**1**
132	2,0E-02	2	9,5E-02	4	**0,0E+00**	**1**	0,0505	3	**0,0E+00**	**1**
133	7,5E-02	4	3,1E-02	2	1,2E-01	5	0,0619	3	**0,0E+00**	**1**
134	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**	**0,0E+00**	**1**

3.3. The performances of the BinAOA and BinAOAX on the classical benchmark functions

The BinAOA and BinAOAX are also tested on eleven unimodal and multimodal benchmark functions. These functions are shown in Table 27. With these tests, the success of both algorithms has been demonstrated again. Also, the effect of xor gate on the performance of BinAOA has been shown. BinaAOA and BinaAOAX methods are compared with the binary versions of Jaya (Rao, 2016; Aslan et al., 2019), Gray Wolf Optimization (GWO) (Mirjalili et al., 2014), and Tree Seed Algorithm (TSA) (Cinar et al., 2017) algorithms, which have been recently developed and are well-known in the literature. Parameter setups for BinAOA, BinAOAX, Jaya, GWO, and TSA are shown in Table 28. The comparison results are shown in Table 29. All algorithms have been studied independently 20 times. The average (Avg), standard deviation (Std), Best and Worst values of the obtained results were calculated. Continuous values are used as decision variables in the benchmark problems in this study. Each candidate solution's dimension is represented by 50 bits, for a total of 500 bits for each candidate solution. This dimensional length is used to obtain the binary equivalents of the continuous values. Because each candidate solution is made up of binary data, they must be converted to continuous values before the cost of the candidate solutions can be determined. This converting process is as follows:

$$ContinuousValue_i = LowerB_i + \frac{(UpperB_i - LowerB_i) \times DecimalValue_i}{2^m - 1} \quad (13)$$

where ContinuousValue$_i$ is the continuous value for the ith dimension of the numeric vector and *DecimalValue$_i$* is a decimal value (*DecimalValue*) of *m*-dimensional binary vector for ith dimension numeric vector. *UpperB$_i$* is the upper bound of the ith dimension and *LowerB$_i$* is the lower bound of the ith dimension.

The new candidate generation scenes of the basic BinAOA, BinAOAX, Jaya, TSA, and GWO are replaced as follows, respectively:

$$Pnew_{i,j} = \begin{cases} P_{i,j} \oplus P_{rand} & if\ (rand_{i,j} < 0.5) \\ P_{i,j} & otherwise \end{cases} \quad \text{(for BinAOA)} \quad (14)$$

$$Pnew_{i,j} = \begin{cases} P_{best} \oplus P_{i,j} & if\ (rand_{i,j} < 0.5) \\ P_{i,j} & otherwise \end{cases} \quad \text{(for BinAOAX)} \quad (15)$$

$$Pnew_{i,j} = \begin{cases} P_{i,j} \oplus (P_{best} \oplus P_{worst}) & if\ (rand_{i,j} < 0.5) \\ P_{i,j} & otherwise \end{cases} \quad \text{(for Jaya)} \quad (16)$$

$$Pnew_{i,j} = \begin{cases} P_{i,j} \oplus (P_{best} \oplus P_{rand}) & if\ (rand_{i,j} < 0.5) \\ P_{i,j} & otherwise \end{cases} \quad \text{(for TSA)} \quad (17)$$

$$Pnew_{i,j} = \begin{cases} P_{alpha} \oplus (P_{beta} \oplus P_{delta}) & if\ (rand_{i,j} < 0.5) \\ P_{i,j} & otherwise \end{cases} \quad \text{(for GWO)} \quad (18)$$

where $Pnew_{i,j}$ is the jth dimension of ith new candidate produced for ith solution, $P_{i,j}$ is the jth dimension of ith solution, P_{best} is the jth dimension of the best solution obtained so far, P_{worst} is the jth dimension of the worst solution obtained so far, and P_{rand} is the jth dimension of neighbor tree randomly selected from the population. $\oplus$ is *xor* gate. P_{alpha}, P_{beta}, and P_{delta} are solutions calculated according to certain fitness values in GWO. Statistical test results are shown in Table 30. The confidence interval of the Wilcoxon Signed-Rank test results of the BinAOA, BinAOAX, Jaya, GWO, and TSA is 0.05 in Table 30. According to the results, BinAOAX obtained statistically significantly different results from other compared algorithms.

Based on average results, BinAOAX outperformed 9 out of 11 benchmark functions (except f1 and f8). Based on standard deviation results, BinAOAX outperformed 7 out of 11 benchmark functions (except f1, f3, f4, and f8). After BinAOAX, the most successful algorithms were Jaya and TSA. The results showed that xor gate improved the performance of BinAOA.

Table 27. Unimodal and multimodal benchmark functions

Function	Range	f_{min}
$f_1(x)=\sum_{i=1}^{n} x_i^2$	[-100,100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10,10]	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	[-100,100]	0
$f_4(x) = max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	[-100,100]	0
$f_5(x)=\sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-30,30]	0

$f_6(x)=\sum_{i=1}^{n}([x_i+0.5])^2$	[-100,100]	0
$f_7(x)=\sum_{i=1}^{n} ix_i^4 + random[0,1)$	[-1.28,1.28]	0
$f_8(x)=\sum_{i=1}^{n} -x_i\, sin(\sqrt{\lvert x_i \rvert})$	[-500,500]	-418.9829×5
$f_9(x)=\sum_{i=1}^{n}[x_i^2 - 10cos(2\pi x_i) + 10]$	[-5.12,5.12]	0
$f_{10}(x)=-20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + 20 + exp(1)$	[-32,32]	0
$f_{11(x)} = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[-600,600]	0

Table 28. Parameter setups for BinAOA, BinAOAX, Jaya, GWO, and TSA.

Methods	Population size (N)	MaxFEs	Dimension (n)	Bit size for each dimension (m)	The dimension of the candidate solution (n×m)
BinAOA	50	n×10000	10	50	500
BinAOAX	50	n×10000	10	50	500
Jaya	50	n×10000	10	50	500
GWO	50	n×10000	10	50	500
TSA	50	n×10000	10	50	500

Table 29. BinAOA, BinAOAX, GWO, TSA, Jaya algorithms compared on dimension=10

		Jaya	**TSA**	**GWO**	**BinAOA**	**BinAOAX**
f1						
	Best	3,05E+03	**2,91E+03**	7,24E+03	7,39E+03	3,26E+03
	Worst	6,82E+03	7,01E+03	1,63E+04	1,32E+04	7,92E+03
	Avg	5,02E+03	**4,86E+03**	1,24E+04	1,08E+04	5,27E+03
	Std	1,27E+03	**1,22E+03**	2,60E+03	1,64E+03	1,27E+03
f2						
	Best	**1,30E+01**	1,46E+01	3,07E+01	1,55E+01	1,64E+01
	Worst	4,31E+01	3,84E+01	7,15E+02	3,78E+01	3,34E+01
	Avg	2,61E+01	2,58E+01	2,12E+02	2,92E+01	**2,57E+01**
	Std	6,81E+00	5,88E+00	2,11E+02	5,13E+00	**4,47E+00**
f3						
	Best	4,30E+03	**3,43E+03**	4,57E+03	3,82E+03	4,95E+03
	Worst	8,90E+03	1,09E+04	1,94E+04	8,90E+03	1,03E+04
	Avg	6,98E+03	8,15E+03	1,29E+04	6,68E+03	**6,07E+03**
	Std	**1,15E+03**	1,63E+03	4,17E+03	1,29E+03	1,32E+03
f4						
	Best	3,48E+01	3,65E+01	4,65E+01	**3,31E+01**	3,36E+01
	Worst	5,13E+01	5,41E+01	7,26E+01	5,83E+01	5,67E+01
	Avg	4,29E+01	4,57E+01	6,28E+01	4,76E+01	**3,89E+01**
	Std	4,97E+00	**4,23E+00**	6,11E+00	6,35E+00	4,87E+00
f5						
	Best	5,83E+05	8,53E+05	7,91E+05	4,57E+06	**4,99E+05**
	Worst	7,18E+06	1,54E+07	4,15E+07	1,69E+07	8,11E+06
	Avg	2,91E+06	5,02E+06	2,22E+07	9,99E+06	**2,68E+06**
	Std	1,67E+06	4,19E+06	1,07E+07	3,05E+06	**1,63E+06**
f6						
	Best	**1,85E+03**	2,97E+03	7,86E+03	5,84E+03	1,94E+03
	Worst	7,16E+03	7,42E+03	1,82E+04	1,30E+04	8,78E+03
	Avg	4,83E+03	4,86E+03	1,31E+04	1,00E+04	**2,78E+03**
	Std	1,33E+03	1,23E+03	3,00E+03	1,82E+03	**1,24E+03**
f7						
	Best	1,51E-01	1,29E-01	1,42E+00	2,08E+00	**4,17E-02**
	Worst	1,74E+00	1,54E+00	1,03E+01	4,51E+00	1,84E+00
	Avg	6,18E-01	7,85E-01	6,15E+00	3,08E+00	**4,60E-01**
	Std	3,33E-01	3,60E-01	2,42E+00	8,10E-01	**2,49E-01**
f8						
	Best	**-2,77E+03**	-2,41E+03	-1,96E+03	-2,18E+03	-2,63E+03
	Worst	-1,76E+03	-1,57E+03	-1,14E+03	-1,81E+03	-1,59E+03
	Avg	**-2,15E+03**	-2,02E+03	-1,41E+03	-1,89E+03	-2,09E+03
	Std	2,74E+02	2,23E+02	2,07E+02	**1,05E+02**	2,63E+02
f9						
	Best	**3,30E+01**	6,95E+01	6,95E+01	6,62E+01	5,96E+01
	Worst	9,79E+01	9,49E+01	1,29E+02	9,67E+01	1,00E+02

	Avg	8,37E+01	8,52E+01	1,11E+02	8,40E+01	**8,30E+01**
	Std	1,37E+01	6,13E+00	1,45E+01	7,81E+00	**1,12E+01**
f10						
	Best	1,48E+01	**1,45E+01**	1,81E+01	1,69E+01	1,55E+01
	Worst	1,90E+01	1,90E+01	2,03E+01	1,95E+01	1,89E+01
	Avg	1,73E+01	1,69E+01	1,95E+01	1,85E+01	**1,59E+01**
	Std	1,15E+00	1,13E+00	5,82E-01	7,24E-01	**7,00E-01**
f11						
	Best	3,10E+01	2,54E+01	6,54E+01	4,77E+01	**1,74E+01**
	Worst	7,50E+01	6,98E+01	1,71E+02	1,14E+02	9,48E+01
	Avg	5,31E+01	4,73E+01	1,20E+02	9,19E+01	**3,55E+01**
	Std	1,20E+01	1,49E+01	2,77E+01	1,92E+01	**1,42E+01**

Table 30. The statistical test results on BinAOA, BinAOAX, GWO, TSA, and Jaya algorithms results

f_no	**BinAOAX-BinAOA**	**BinAOAX-Jaya**	**BinAOAX-TSA**	**BinAOAX-GWO**
	p	**p**	**p**	**p**
f1(x)	8,86E-05	6,81E-01	2,96E-01	8,86E-05
f2(x)	3,66E-02	8,81E-01	9,11E-01	8,86E-05
f3(x)	1,91E-01	2,28E-02	2,20E-03	2,19E-04
f4(x)	3,38E-04	6,20E-02	2,54E-04	8,86E-05
f5(x)	8,86E-05	7,65E-01	5,69E-02	8,86E-05
f6(x)	8,86E-05	7,80E-04	1,40E-04	8,86E-05
f7(x)	8,86E-05	1,67E-01	2,51E-02	8,86E-05
f8(x)	8,03E-03	4,78E-01	3,32E-01	8,86E-05
f9(x)	8,23E-01	6,54E-01	5,50E-01	2,93E-04
f10(x)	8,86E-05	8,92E-04	1,69E-02	8,86E-05
f11(x)	8,86E-05	3,59E-03	2,76E-02	8,86E-05

4. Conclusion

AOA is a newly developed heuristic algorithm in recent years. AOA is recommended for continuous optimization tasks. The success of AOA on binary optimization problems has not been tested in the literature. Binary AOA (BinAOA) has been proposed in this study. The structure of AOA has been updated and has gained the ability to solve binary optimization problems as well. In this study, the second version of BinAOA is proposed. A candidate solution strategy has been added to BinAOA. Logic gates are used in this candidate solution strategy. Xor logic gate was used between the best solution of the population and a random solution of the population. New candidate solutions produced have increased the performance of BinAOA. The new version of this proposed BinAOA is named BinAOAX. BinAOA and BinAOAX methods have been tested on UFL problems. UFL problems are binary optimization problems whose optimum results are known in the literature. It is frequently used as binary optimization test problems in the literature. When BinAOA and BinAOAX are compared with the heuristic binary optimization algorithms (TSA, ISS, the variations of the binary Jaya, the variations of the binary PSO, the variations of the binary DE, the variations of the binary ABC) used in the literature, BinAOA and BinAOAX can be preferred in the solutions of binary optimization problems. As a second test application, the performance of BinAOA and BinAOAX algorithms are also tested on unimodal and multimodal classical benchmark functions. The binary forms of AOA, AOAX, Jaya, Tree Seed Algorithm (TSA), and Gray Wolf Optimization (GWO) algorithms were compared in different candidate generation scenarios. BinAOAX has shown a very successful performance.
In future studies, the success of AOA on the feature selection problem, which is a different binary optimization problem, will be tested.

Credit authorship contribution statement
Emine BAŞ: Conceptualization, Investigation, Methodology, Software, Writing – review, original draft & editing.
Gülnur Yıldızdan: Review, original draft & editing.

Declaration of competing interest
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding: This study was not funded by any institution.

Acknowledgements: The BAOA algorithm was previously presented at the 3rd HAGIA SOPHIA INTERNATIONALCONFERENCE ON MULTIDISCIPLINARY SCIENTIFIC STUDIES conference and the details of BAOA are given in this paper.

References

Abualigah, L., Diabat, A., Mirjalili, S., Elaziz, MA., Gandomi, A.H., 2021. The Arithmetic Optimization Algorithm, Comput. Methods Appl. Mech. Engrg. 376 (2021) 113609.

Aslan, M., Gunduz, M., Kiran M.S., 2019. JayaX: Jaya algorithm with xor operator for binary optimization, Applied Soft Computing Journal 82, 105576.

Baş, E., Ülker, E., 2020a. A binary social spider algorithm for uncapacitated facility location problem, Expert Systems with Applications 161 (2020) 113618.

Baş, E., Ülker, E., 2020b. An efficient binary social spider algorithm for feature selection problem, Expert Systems With Applications 146, 113185.

Beşkirli M., Koc I., Hakli H., Kodaz H., 2018, A new optimization algorithm for solving wind turbine placement problem: binary artificial algae algorithm, Renew Energy 121:301–308.

Çınar, A.C., Kiran, M.S., 2018. Similarity and Logic Gate-Based Tree-Seed Algorithms for Binary Optimization, Computers & Industrial Engineering 115, 631–646.

Çınar, A.C., Iscan, H., Kiran, M.S., 2017. Tree-Seed algorithm for large-scale binary optimization, IAIT Conference Proceedings, The 9th International Conference on Advances in Information Technology Volume 2017.

Hakli, H., Ortacay, Z., 2019. An improved scatter search algorithm for the uncapacitated facility location problem, Computers & Industrial Engineering, 135, 855–867.

Kashan, M.H., Kashan, A.H., Nahavandi, N., 2013. A novel differential evolution algorithm for binary optimization, Comput. Optim. Appl. 55 (2) 481–513.

Kiran, M.S. (2015). The continuous artificial bee colony algorithm for binary optimization, Appl. Soft Comput. 33, 15–23.

Kiran, M.S., Gunduz, M., 2013. XOR-based artificial bee colony algorithm for binary optimization, Turkish Journal of Electrical Engineering & Computer Sciences, 21(sup. 2), 2307–2328.

Korkmaz, S., Kiran, M.S., 2018. An artificial algae algorithm with stigmergic behavior for binary optimization, Applied Soft Computing 64, 627–640.

Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer, Adv. Eng. Softw. 69, 46 – 61.

Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremii, S., Faris, H., Mirjalili, S.M., 2017, Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114, 163 – 191.

Rao, R. (2016). Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems.International Journal of Industrial Engineering Computations, 7(1), 19-34.