Pamukkale Univ Muh Bilim Derg, 29(7), 737-751, 2023

Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi

Pamukkale University Journal of Engineering Sciences

A binary enhanced moth flame optimization algorithm for uncapacitated facility location problems

Kapasitesiz tesis yerleşim problemleri için geliştirilmiş ikili güve alevi optimizasyon algoritması

Ahmet ÖZKIŞ[1*], *Murat KARAKOYUN*[2]

[1]Department of Computer Forensics Engineering, Faculty of Engineering, Necmettin Erbakan University, Konya, Turkey.
aozkis@erbakan.edu.tr

[2]Department of Computer Engineering, Faculty of Engineering, Necmettin Erbakan University, Konya, Turkey.
mkarakoyun@erbakan.edu.tr

Received/Geliş Tarihi: 26.07.2022 Revision/Düzeltme Tarihi: 06.11.2022 doi: 10.5505/pajes.2023.49576
Accepted/Kabul Tarihi: 06.01.2023 Research Article/Araştırma Makalesi

Abstract

Moth Flame Optimization is a nature-inspired meta-heuristic algorithm for constantly solving real-world problems. In this study, a modified version of MFO called binary Enhanced MFO Desert Bush (binEMFO-DB) algorithm is proposed to solve uncapacitated facility location problems. The proposed algorithm includes three modifications: i) chaotic map-based population initialization, ii) random flame selection, and iii) desert bush strategy. The performance of the proposed binEMFO-DB algorithm was tested on 15 different UFL problems from the OR-Library and Taguchi orthogonal array design was used for parameter analysis. The average, gap and hit values of the results obtained by the algorithms were used as performance metrics. The performance of binEMFO-DB is compared with the performance of state-of-the-art algorithms. The results show that the proposed binEMFO-DB has a successful and competitive performance in the test environment.

Keywords: Moth flame optimization, Uncapacitated facility location problem, Binary optimization, Desert bush, Transfer functions, Taguchi.

Öz

Güve Alevi Optimizasyonu, sürekli gerçek dünya problemlerini çözmek için doğadan ilham alan bir meta-sezgisel algoritmadır. Bu çalışmada, kapasitesiz tesis yerleşim problemlerini çözmek için ikili Enhanced MFO Desert Bush (binEMFO-DB) algoritması olarak adlandırılan MFO'nun değiştirilmiş bir versiyonu önerilmiştir. Önerilen algoritma üç değişiklik içermektedir: i) kaotik harita tabanlı popülasyon başlatma, ii) rastgele alev seçimi ve iii) çöl çalısı stratejisi. Önerilen binEMFO-DB algoritmasının performansı, OR-Library'den alınan 15 farklı UFL problemi üzerinde test edilmiş ve parametre analizi için Taguchi ortogonal dizi tasarımı kullanılmıştır. Algoritmalar ile elde edilen sonuçların ortalama, boşluk ve isabet değerleri performans metriği olarak kullanılmıştır. binEMFO-DB'nin performansı, son teknoloji algoritmaların performanslarıyla karşılaştırılmıştır. Elde edilen sonuçlar, önerilen binEMFO-DB'nin test ortamında başarılı ve rekabetçi bir performansa sahip olduğunu göstermektedir.

Anahtar kelimeler: Güve alevi optimizasyonu, Kapasitesiz tesis yerleşim problemi, İkili optimizasyon, Çöl çalısı, Transfer fonksiyonları, Taguchi.

1 Introduction

In the last 20 years, the use of metaheuristic algorithms has become increasingly common for optimization problems that cannot be solved by classical mathematical approaches or that take a long time to solve. Metaheuristic algorithms attract the attention of researchers because they can be easily adapted to different optimization problems. And these algorithms obtain near-optimal results regardless of the number of objective functions (such as single-objective and multi-objective) and decision variable structure (continuous, unconstrained, constrained, discrete, and binary) [1].

In binary optimization problems (BOPs), the decision variables can take one of the two values represented by 0 and 1. For example, in power systems, 0 represents the "off" state, 1 the "on" state [2]; in binary image processing, while 0 means black color, 1 means white color [3]. Many real-world issues, especially classification and clustering problems, cell formation, network optimization, unit commitment, knapsack problems, compression-related problems, seat scheduling are considered as BOPs [4],[5]. In this study, the solution of uncapacitated facility location problems (UFLPs) – one of the binary optimization problems-with metaheuristic algorithms will be emphasized.

In UFLP there are customers and facilities and it is decided which facilities should be open and which should be closed in order to provide the most cost-effective service to customers in different locations (at least one facility must be open). Assuming there is a total of *n* facilities, the facilities can exist in 2^n-1 different states.

Since the complexity of the problem increases as the number of facilities accrue, UFLP is considered as an NP-Hard problem [6]. While traditional methods such as, branch-and-bound, lagrangian techniques, relaxation methods, reduction schemes, and integer programming suggested in the literature are successful in solving low-dimensional problems, they do not perform well in high-dimensional problems [7],[8]. For this reason, researchers have turned to metaheuristic algorithms that can guarantee near-optimal results, can easily adapt to different types of problems and reach solutions in a reasonable time with having a simple structure [6]. Most of metaheuristic algorithms, such as particle swarm optimization (PSO) [9], grey wolf optimizer (GWO) [10], artificial bee colony (ABC) [11],

*Corresponding author/Yazışılan Yazar

artificial algae algorithm (AAA) [12] provide solutions to optimization problems have continuous search space in their basic versions. For this reason, these algorithms need to be adapted to binary problems by using techniques such as the following [4]:

- Transfer Functions: Kennedy and Eberhart [13] in 1997 has provided the conversion of continuous variables to binary values via the sigmoid function. After this study, new transfer functions that convert continuous variables into binary values have been proposed by different researchers [14],[15],
- Angle modulation: This technique was first used in the field of signal processing. In this method, four real-valued parameters are converted to binary values using the sine and cosine function [16],
- Genetic operators: Crossover operators such as single point, n-point, uniform, discrete, simulated binary etc. [17] are commonly used in evolutionary algorithm. Most of these operators can be adapted to solve continuous [11], discrete [18] and binary problems [19],
- Logical operators: In metaheuristic algorithms, logical operators and, or, xor and not can be used to generate candidate solutions. If the search space is binary, logical operators can be used directly [1],[6]; if the search space is continuous, a strategy such as transfer functions should be used first to convert the solutions to the binary values [7],
- Measure of dissimilarity: These are the metrics used to calculate the dissimilarity of two arrays in binary structure. Jaccard's and Dice's similarities [7]; Euclidean, Hamming, Manhattan metrics [20] etc. are commonly used as dissimilarity metrics of measurement.

Heuristic methods [21] and Quantum-inspired bits [22] are also used for binarization.

Of the methods listed above, transfer functions, angle modulation, heuristic methods can be used to convert continuous space to binary space. Other methods are widely used to increase the variety of solutions already transferred to binary space. In this study, transfer functions are used to apply MFO, which is a continuous algorithm, to UFL, which is a binary problem. However, different modifications have been applied to increase the performance of the algorithm. The performance of the proposed algorithm has been compared with the performance of different algorithms presented in the literature.

The remainder of the paper is organized as follows: In Section 2, a literature review is given for metaheuristic-based approaches proposed to solve BOPs. In Section 2.1, main motivation and contribution of the study is accentuated. UFLP problems are described in Section 3. In Section 4, the original MFO and the proposed approach are described in detail. In Section 5, parameter analysis, experimental studies and comparison results are given. Finally, Section 6 contains conclusions and discussions.

2 Literature review

A large number of various metaheuristic algorithms from past to present that have been successful in solving UFL problems are mentioned. While some of these algorithms are created by initializing the position values directly with binary coding, some of them are created by converting the continuous position values into binary with methods called transfer functions. Both methods are widely used in the literature. In the continuation of this section, some prominent metaheuristic algorithms are introduced and their suggested variants for solving binary problems are briefly mentioned.

2.1 Binary PSO variants

PSO [8] was proposed in 1995 inspiring by the foraging behavior of bird and fish flocks and is one of the most reputable metaheuristic algorithms. In 1997, the first attempt to binarization of the PSO was made by researchers Kennedy and Eberhart who proposed the algorithm. In this binary PSO (BPSO) [12], real variables were easily converted to binary values using the sigmoid function. It has been observed that the exploration capability of the BPSO is insufficient in high dimensional problems. To overcome this problem, a new algorithm has been proposed by Khanesar et al. [23] that changes the velocity vector in PSO. Lin et al. [24] proposed a binary PSO approach that extracts high utility item sets. Yuan et al. proposed IBPSO [2] algorithm and solve with this algorithm unit commitment problems. Main difference of the IBPSO from basic PSO is that population initialized and updated in binary space. Other researchers that suggested PSO variants to solve BOPs can be mentioned as follows: Beheshti et al [14], Nezamabadi-pour et al. [25], Guner and Sevkli [26] and Saha et al. [27].

2.2 Binary ABC variants

ABC [28] is a metaheuristic algorithm developed by Karaboga in 2005, inspired by the foraging behavior of honey bees. Kashan proposed the DisABC [29] algorithm, which initialized directly with binary values and uses Jaccard's similarity for position update and used the UFLPs set for performance testing. Kiran and Gunduz proposed binABC [30] which used XOR logical operator for position update and tested this algorithm on UFLPs. Similarly, Kiran proposed a new stigmergic behavior-based ABC algorithm [31] and done its performance assessment on CEC 2015 functions and UFLPs. Jia et al. proposed bitwise based ABC, shortly bitABC [32] and tested it on a continuous benchmark set. Ozturk proposed a new binary ABC named GB-ABC [19] using genetic operators and done performance assessment on a dynamic image dataset and knapsack problems.

2.3 Binary DE variants

Differential evolution (DE) [33] is a well-known evolutionary based algorithm proposed by Storn and Price in 1997. Pampara proposed angle modulated DE (AMDE) [16]-a binary variant of DE using angle modulation technique-and tested the performance of the AMDE on classical benchmark functions. Engelbrecht and Pampara [34] proposed two different binary DE approaches: of these binDE, borrows the concept of binPSO, while normDE uses normalization strategy in continuous spaces between lower and upper bound. If the normalized value lower than 0.5, it set to 0, otherwise it set to 1. Su and Yang suggested a quantum-based DE (QDE) [35] algorithm. Chen et al. suggested BLDE algorithm [21] which is learning from already explored solutions and tested the algorithm on knapsack problems. He et al. suggested binary DE (BDE) [36] and addressed the BDE as feature selector on 6 different UCI datasets. Deng et al. [37] proposed another DE variant by using a mapping operator and s operator to solve knapsack problems.

Other DE variants for BOPs can be referred as follows: Yang [38], Wang et al. [39], Kashan et al. [40].

2.4 Binary GSA variants

Gravitational search algorithm, GSA for short [41] is suggested in 2009 by Rashedi et al. Main motivation of the GSA is that mass interactions and Newtonian law of gravity. Scientists recommending the GSA also suggested binary GSA (BGSA) [42] in 2010 by using transfer functions. Nezamabadi-pour proposed a binary quantum-inspired GSA (BQIGSA) [22] and used combinatorial 0–1 knapsack problems to measure the performance of the algorithm. Khanesar and Branson suggested XOR based binary GSA (XOR-BGSA) [43] and used knapsack dataset for performance evaluation.

2.5 Other variants

Other metaheuristic approaches recommended to solve BOPs can be given as follows: Aslan et al. [6] proposed 2 different variants of basic Jaya algorithm [44]: i) XOR-based Jaya algorithm (JayaX), ii) local search mechanism added version of JayaX (JayaX-LSM). Proposed approaches compared on CEC2015 functions and UFLPs and it was seen that JayaX-LSM outperformed than JayaX. Cinar and Kiran [45] modified the basic tree-seed algorithm (TSA) [46] in three different approaches: i) logic gate based (LogicTSA), ii) similarity measurement based TSA (SimTSA), iii) Hybrid variant (SimLogicTSA). These 3 approaches were handled on UFLP suit and SimLogicTSA obtained better results than compared techniques. Hakli and Ortacay proposed an improved scatter search algorithm (scatter search-ensemble crossover, SS-EC) [47] and compared this techniques against other techniques founded in the literature. Bas and Ulker proposed BinSSA [7] by modifiying basic SSA [48] with transfer functions, similarity measures and logic gates. In [49], a crossover operator added variant of the BinSSA was run on the feature selection problems and obtained successful results. In [50], S-shaped and V-shaped four different binary variants of the SSA were proposed and this variants were run on continuous benchmark tasks. Korkmaz et al. proposed a binary initialized AAA method [51], binAAA for short, and compared this technique to recently proposed other algorithms on UFLPs. Cinar proposed a binary Archimedes optimization algorithm [52] by using 17 different transfer functions. Karakoyun and Ozkis proposed a binary variant of the TSA with enhanced local search module on CAP and M* problems [53].

2.6 Main motivation and contribution of the study

As can be seen from the literature review, many metaheuristic algorithms have been suggested for the solution of BOPs and this effort is still ongoing by many researchers today. Although this situation has been criticized by some scientists [54], proposing of new techniques is highly necessary according to the No Free Lunch (NFL) theorem [55]. According to the NFL theorem, the high performance of any algorithm on a class of problem is balanced by its performance on another class. That is, no algorithm can guarantee to find the optimal solution for all problem types. This issue encourages researchers in the matter of recommending new techniques that produce better results than already proposed algorithms to different types of problems. With this motivation, in this study a recently proposed metaheuristic algorithm, moth flame optimization (MFO), is handled and a novel method suggested to solve BOPs.

The MFO was recommended by Mirjalili [56] in 2015, inspired by a navigation technique that real moths use to navigate at night. The MFO has been used by researchers [57] to solve various optimization problems due to its simplicity, flexibility and easy adaptability. These can be summarized as classification [58], image processing [59], medical [60], power energy [61], inverse problem and parameter estimation [62], [63], scheduling [64], engineering design [65], and economic [66]. In addition, multi-objective [67]-[69], binary [70] and hybrid [71]-[79] variants of the MFO is available in the literature. While MFO is a widely used type of optimizer, there is only one MFO variant recommended for solving BOPs, as far as we can find.

The main contribution of this study is that some performance improvement modifications are made on the original MFO algorithm and the binary Enhanced MFO Desert Bush (binEMFO-DB) algorithm is suggested. The details about the proposed algorithm are presented in section 3.2. The binEMFO-DB algorithm was run on the UFLPs taken from OR-Library [80] and the obtained results were compared with the results of similar studies in the literature. The experimental results show that the proposed algorithm is generally successful on UFLP and has better scores when compared with the performance of the other algorithms.

3 Problem definition: uncapacitated facility location problem (UFLP)

UFLP is one of the main and hard binary problems faced in real life. The problem basically consists of the facilities providing service and the customers receiving service from these facilities. The location of the facilities and the cost of the service to be provided to the customers from these facilities determine the total cost. The main purpose in solving the problem is to determine the optimum facility location that will minimize the total cost. Assume that n is the total number of facilities (consisting of opened or closed facilities), the number of possible solutions for the location of the facilities is 2^n. However, considering that at least one facility must be open in the UFL problem, the number of these solutions becomes 2^n-1. It is clearly seen that the number of facilities (n) directly affects the complexity of the problem. Besides, the installation cost of the facilities causes the problem to be included in the NP-Hard problem class [81]-[84].

In the UFL problem, while the total potential facility locations are known, it is not known which of these facilities will be open. A constant installation cost is required for each facility. Besides, there is also a transportation fee between the customers and the facility, and each customer is associated with the facility that is easiest to reach (least cost). The main objective of the UFLP is to minimizing the overall cost that consists of installing the facilities and supplying customers from the facilities [84]-[86]. Assume that F_T is the set of the all facilities; the purpose is to determine a subset (F_{sub}) of facilities that minimize the total cost. Equation (1) shows the total cost where $F_{sub} \subseteq F_T$.

$$f(F_{sub}) = \sum_{i \epsilon F_{sub}} f_i + \sum_{j \epsilon P} \min\{c_{ij} \mid i \epsilon F_{sub}\} \quad (1)$$

Where f_i is the cost of an open facility, c_{ij} is the cost between i_{th} facility and j_{th} customer and P is set of the customers. In UFLP, a solution (x) is presented with a binary vector ($x \in \{0,1\}^q$, where q is number of potential facilities) which $x_i = 1$ or $x_i = 0$ if i_{th} facility is open or close, respectively. The notation $F_{sub^1} = F_{sub^1}(x) = \{i \in F_{sub} : x_i = 1\}$ is used to

represent the open facilities in solution x. Then the fitness function of the UFLP can be expressed mathematically as follow [84]:

$$Min\ f(x) = \sum_{i \epsilon F^1_{sub}(x)} f_i + \sum_{j \epsilon P} \min\{c_{ij} \mid i \epsilon F^1_{sub}(x) \mid x \in \{0,1\}^q - \{0\}\} \quad (2)$$

OR-Library [80] is a useful resource that presents to the researchers a large data set for the UFLP. The properties of 15 different problems taken from the OR-Library are given in Table 1.

Table 1. The properties of the OR-Library problems

Problem	Type	Dimension	Optimum Cost
Cap71	Small	16 x 50	932615.750
Cap72		16 x 50	97779.400
Cap73		16 x 50	1010641.450
Cap74		16 x 50	1034976.975
Cap101	Medium	25 x 50	796648.438
Cap102		25 x 50	854704.200
Cap103		25 x 50	893782.113
Cap104		25 x 50	928941.750
Cap131	Large	50 x 50	793439.563
Cap132		50 x 50	851495.325
Cap133		50 x 50	893076.713
Cap134		50 x 50	928941.750
CapA	Huge	100 x 1000	17156454.478
CapB		100 x 1000	12979071.580
CapC		100 x 1000	11505594.330

When categorized according to their dimensions, it is seen that there are 4 different problem types. While Cap71-74 problems are in the small problem type with 16 facilities and 50 customers, Cap101-104 problems constitute the medium problem type with 25 facilities and 50 customers. Cap131-134 is included in the big problem type with 50 facilities and 50 customers. Finally, CapA, CapB and CapC problems provide a very large problem type with 100 facilities and 1000 customers.

4 Moth flame optimization (MFO) algorithms

In this section, the basic MFO algorithm, binary MFO algorithm and the proposed binEMFO-DB algorithm were presented with details.

4.1 Basic MFO algorithm

The MFO [56] algorithm which is proposed by Mirjalili is inspired by the nocturnal flight strategy of moths. Moths have a flying mechanism which uses the moon light with a stable angle. The mechanism that they use for navigation is called as transverse orientation. This strategy provides an effective and comfort travelling in a long straight distance. On the other hand, the moths are affected from artificial lights and try to act similar with having an angle with this artificial light.

The flying of the moths by keeping a constant angle between them and the light causes a spiral movement. Figure 1 shows the spiral flying of the moths around the light. It can be observed that the transverse orientation strategy is effective only for the far lights like moonlight [56]-[58],[87],[88].

According to the Figure 1 it can be seen that the moths eventually close towards the light source. The MFO algorithm was mathematically developed by modeling the behavior of moths with the light source. Like other metaheuristic algorithms, the MFO is also an iterative and population-based algorithm. The algorithm basically consists of moths and flames.

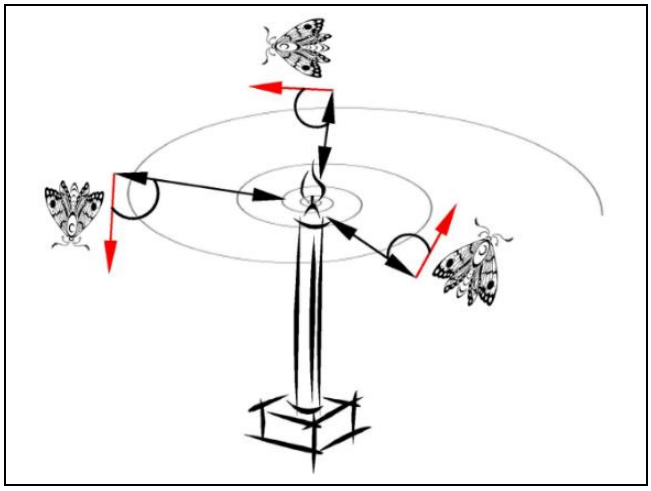

Figure 1. Spiral flying of moths around the light

While each moth in the population represents a possible solution, each variable that constitutes the position of the moth represents one dimension of the problem. As mentioned before the MFO is population based. Let's assume that N is the population size and D is the dimension of the problem then the population of the moths can be represented with a matrix as follow:

$$M = \begin{bmatrix} m_{11} & \cdots & m_{1D} \\ \vdots & \ddots & \vdots \\ m_{N1} & \cdots & m_{ND} \end{bmatrix} \quad (3)$$

Where M is the population of the moths there is an array of the fitness values that related with the positions. The array of the fitness values (OM) can be represented as follow:

$$OM = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_N \end{bmatrix} \quad (4)$$

The moths in population are required an updating process to improve their position. In updating process each moth needs a reference flame that is unique. With the location update of the moths by feeding from different flames, it is aimed to avoid the local optima and to make an effective search at the global level. The position of the flames has the same size as the moths and is represented similarly as follow:

$$F = \begin{bmatrix} f_{11} & \cdots & f_{1D} \\ \vdots & \ddots & \vdots \\ f_{N1} & \cdots & f_{ND} \end{bmatrix} \quad (5)$$

There is also an array of fitness values for these flames represented as follow:

$$OF = \begin{bmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_N \end{bmatrix} \quad (6)$$

It should not be forgotten that moths and flames are the same in terms of presentation and structure. The difference between them is the way they are treated within the population. The

moths update their positions for each iteration, while the flames are array of the best positions ever found. On the other hand, moths are assisted by a flame as a reference point during the position update process [56]. The mathematical model of the position update that inspired by Figure 1 is given in Eq. (7).

$$M_i = D_i * e^{bt} * \cos(2\pi t) + F_j \quad (7)$$

$$D_i = |F_j - M_i| \quad (8)$$

Where $M_i = (m_{i1}, m_{i2}, \dots m_{iD})$ and $F_j = (f_{i1}, f_{i2}, \dots f_{iD})$ are the positions of the i_{th} moth and j_{th} flame respectively, D_i is the distance between i_{th} moth and related j_{th} flame that calculated by Eq. (8), t is a number generated randomly in [-1, 1] and generated by Eq. (9) and b is a constant value to determine the form of the logarithmic spiral.

$$t = (a-1) * rand + 1$$
$$a = -1 + k * (-\frac{1}{K}) \quad (9)$$

where k is the current iteration number, and K is the maximum iteration number.

To have a better position updating process, the number of the flames is decreased for each iteration by using Eq. (10) as follow:

$$flame\_number = round\left(N - k * \frac{N-k}{K}\right) \quad (10)$$

where k is the current iteration number, K is the maximum iteration number and N is the maximum flame number that is equal to population size at the beginning of the algorithm.

The MFO algorithm has a similar processing mechanism as other metaheuristic algorithms. The parameters of the algorithm must be set in first step. Then a random population is generated within the boundary of the solution space. For each moth (position) in population, fitness values are calculated and the flames are assigned. The main loop of algorithm is started.

1. Set parameters of the algorithm
2. Generate first population randomly within solution space
3. **while** *termination criterion is not met* **do**
4. Update *flame_number* using Eq. (10)
5. Calculate fitness values of the each moth in population
6. **if** *first iteration* **then**
7. Sort the population according to the fitness values from best to worst
8. Assign the flames with the whole population
9. **else**
10. Merge the population and flames
11. Sort the merged solutions from best to worst
12. Select *flame_number* best solution and assign to flames
13. **end if**
14. **foreach** *moth* in *Population* with $i \leq N$ **do**
15. Generate t value using Eq. (9)
16. **if** $i <$ *flame_number* **then**
17. Update the position of the ith moth with ith flame using Eq. (7)
18. **else**
19. Update the position of the ith moth wit *flame_number*th flame using Eq. (7)
20. **end if**
21. **end foreach**
22. **end while**
23. return *best solution* as output

Algorithm 1. The main steps of the MFO algorithm

In this loop, for each moth the position update procedure works, the number of the flames is updated and best position is saved for each iteration step. The loop continues until the termination criterion is met [56], [57]. Algorithm 1 shows the main steps of the MFO algorithm.

4.2 Binary MFO algorithm

The basic MFO algorithm is proposed to solve continues optimization problems. However, in binary optimization problems such as UFLP, a binary solution structure is needed to calculate the objective function and to handle the problem. The position update strategy of continuous algorithms is not suitable for binary optimization problems. Therefore, using a private transfer function to convert from continuous form to binary form is an appropriate approach to solving the problem. The main purpose of a transfer function is to convert each dimension of a continuous solution into binary values (0 or 1). Transfer functions are generally classified into two topics as S-shaped and V-shaped according to the shape of the transfer function [15], [89]. Table 2 shows four S-shaped and four V-shaped transfer functions used in this study.

Table 2. Transfer functions: S-shaped and V-shaped.

S-shaped		V-shaped	
S1:	$\frac{1}{1+e^{-2x}}$	*V1:*	$\left\lvert \text{erf}\left(\frac{\sqrt{\pi}}{2}x\right)\right\rvert$
S2:	$\frac{1}{1+e^{-x}}$	*V2:*	$\lvert\tanh(x)\rvert$
S3:	$\frac{1}{1+e^{\frac{-x}{2}}}$	*V3:*	$\left\lvert\frac{x}{\sqrt{1+x^2}}\right\rvert$
S4:	$\frac{1}{1+e^{\frac{-x}{3}}}$	*V4:*	$\left\lvert\frac{2}{\pi}\arctan\left(\frac{\pi}{2}x\right)\right\rvert$

Since the MFO is a continuous algorithm, the transfers functions are given in Table 2 were used to achieve a binary MFO. In binary MFO, the moths generate and update their position in continuous form. However, before calculate the value of the objective function, a transfer function is used to generate the binary solution and then objective function is called.

4.3 Proposed algorithm

In this paper, an enhanced binary MFO algorithm is proposed with some modifications to improve the performance. In this section, the modifications are presented in sub sections and the proposed binary MFO algorithm explained with details.

4.3.1 Modifications

After converting the basic MFO algorithm to binary by using transfer functions, three modifications were applied to improve the performance of the algorithm.

4.3.1.1 Chaotic map-based initialization

In population-based optimization algorithms, first population initialization is a very important and critical process. The distribution of members in the population in the solution space directly affects the convergence of the algorithm and the quality of the solution it will obtain. Unless a specific method is specified, optimization algorithms randomly generate the first population. The success of the algorithm is compromised if the initial population cannot be effectively distributed in the

solution space. In order to eliminate this problem, researchers have suggested and used different approaches. Recently, randomly generated parameters of optimization algorithms have started to be obtained with chaotic maps. The initial population can also be included in these parameters. Researchers stated that the values generated by chaotic maps have a more balanced distribution than randomly generated values and use the search space more effectively [90] – [92]. In this study, chaotic maps with different characteristics were tested and the map suitable for the problem was selected and used. Details on the selection of the chaotic map are given in Section 5.1.

4.3.1.2 Flame selection strategy

In the MFO algorithm, each moth chooses a flame as a reference and updates its position with Eq. (7). In the basic MFO algorithm, how the flame (that used as a reference) is determined is given in lines 16-20 in Algorithm 1. Initially, the moths in the population match the flame of the same index in the flame array, respectively. However, in each iteration step, the number of flames decreases and the flames that moths will reference are limited. As the iteration progresses, since the number of flames is less than the number of moths, all the moths with an index number greater than the number of flames refer to the last flame in the flame array. This situation causes the moths in the population to tend to a specific position. To avoid this situation, a new flame selection strategy has been proposed. In this proposed strategy, moths with an index greater than the number of flames is ensured to refer to a randomly selected flame from the flame array instead of the flame in the last index. With this change, it is aimed to add diversity to the moths during the position update process. Flame selection of the basic MFO algorithm and the proposed strategy are shown in Figure 2(a) and Figure 2(b), respectively.

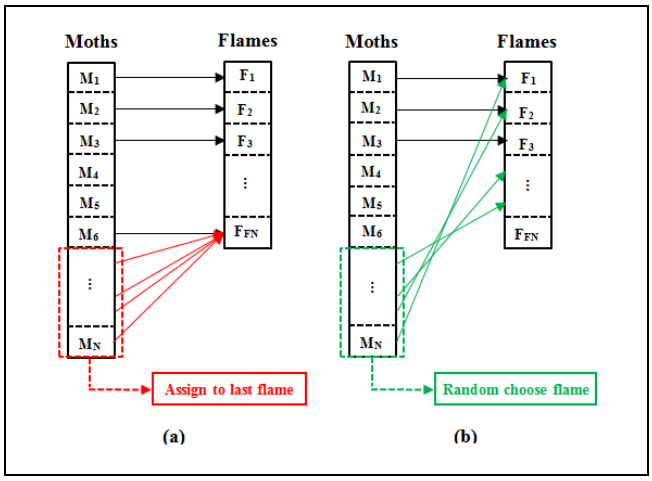

Figure 2. Flame selection. (a): Basic MFO. (b): Proposed strategy.

4.3.1.3 Desert bush

Desert bush, known as the resurrection plant, is a plant species famous for its longevity in arid desert environments. Adapted to the desert environment, desert bush can survive for years without water, in which case it dries up until it retains only 3% of its mass. When living conditions get too harsh, the plant's survival mechanism allows it to gradually dry out, turning its leaves brown and curling. It gives the plant a ball appearance and all its metabolic functions are minimized. When the drought situation increases, its roots are freed from the land and become a free drum plant drifting on dry ground under the influence of the blowing winds. No matter how dry or damaged it is, thanks to the special biological structure of its leaves, the plant retains its ability to absorb water and open itself even years after it dies. The desert bush reproduces by spores; it does not contain seeds or flowers in its structure. The plant, which drifts freely in an arid environment, opens its twisted branches when exposed to a humid environment, allowing the spores to spill, so that the spilled spores are revived in a humid environment [93], [94]. Figure (3) shows the life cycle of a desert bush.

As can be seen from Figure (3), a desert bush that encapsulates itself in bad conditions, resurrects when it finds a suitable environment. This feature of the desert bush has been mathematically modeled and applied to the proposed algorithm as a new strategy. In the proposed strategy, it is aimed to achieve resurgence if the positions of members in the population do not improve by a specified number of iterations and there is no improvement in the global best position. In the modeling, the global best position is selected as the reference point (desert bush), and members (spores) in the population are repositioned according to this point. Algorithm 2 shows the position update of a member according to the modelled desert bush strategy.

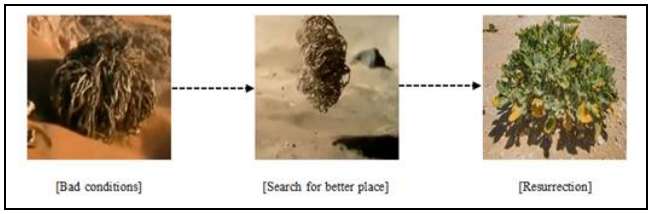

Figure 3. Life cycle of a desert bush.

```
function [newSol] = DesertBush(gBest, moth, lb, ub)
  sr = 0.02;
  newSol = gBest;
  for i=1:size(gBest,2)   //size(gBest,2) = dimension
    if rand < 0.5   //Choose if current dimension will change or not
      // To decide the direction of the step
      if rand<0.5   //A negative step
        lb_i = lb;
        ub_i = gBest(1,i);
        sSize = -(ub_i - lb_i)*sr;
      else   //A positive step
        ub_i = ub;
        lb_i = gBest(1,i);
        sSize = (ub_i - lb_i)*sr;
      end
      xNew = gBest(1,i) + sSize;   //Add negative or positive step size
      newSol(1,i) = xNew;
    else
      newSol(1,i) = moth(1,i);
    end
  end
end
```

Algorithm 2. The code of the modelled desert bush strategy

According to Algorithm 2, the member's position is first synchronized to the global best position. A loop is then started to determine the value of each dimension. For the current dimension, firstly, it is determined whether there will be a

change or not with a selection. If there will be a change, a selection is made again about which direction it will be. According to the selection made, the step size is determined and the position change of the dimension is performed. Here, sr is a constant variable that limits the step size.

4.3.2 The proposed binEMFO-DB algorithm

The MFO [54] that is proposed by Mirjalili is a continuous algorithm. In this study, the MFO algorithm was applied to solve the UFL problem. However, since UFLP is a binary problem, it is not possible to directly apply the basic MFO algorithm. For this reason, firstly, the MFO algorithm is binarized by using transfer functions. Then, some modifications were used to increase the success of the binary MFO algorithm on UFLP. The pseudo code of the proposed algorithm, named binEMFO-DB, is given in Algorithm 3.

The first step of the proposed algorithm is parameter settings. The parameters which are generated in parameter analyses process are set and algorithm starts. Then, first population is generated in boundaries by using chaotic map, instead of a random population and for entire population objective function is called and fitness values are generated. The main loop is started, in which members update their positions. In this loop, different from the basic MFO, a new selection strategy was applied in flame selection. In this selection strategy given between lines 18-23 of Algorithm 3, it is aimed to prevent the population from being directed to a specific flame and to provide diversity as a solution to the algorithm. In addition, desert bush strategy has been applied to the basic MFO algorithm. With this strategy, it is aimed to give the population resurgence when it cannot generate better positions. When the termination criterion is met and the loop is completed, the best position found is given as the solution and the algorithm is finished.

1. Set parameters of the algorithm
2. Generate first population within solution space by using chaotic map
3. Generate binary position by using transfer function
4. Calculate fitness values of the members in population
5. **while** *termination criterion is not met* **do**
6. Update *flame_number* using Eq. (10)
7. Calculate fitness values of the each moth in population
8. **if** *first iteration* **then**
9. Sort the population according to the fitness values from best to worst
10. Assign the flames with the whole population
11. **else**
12. Merge the population and flames
13. Sort the merged solutions from best to worst
14. Select *flame_number* best solution and assign to flames
15. **end if**
16. **foreach** *moth* in *Population* with $i \leq N$ **do**
17. Generate t value using Eq. (9)
18. **if** $i <$ *flame_number* **then**
19. Update the position of the ith moth with ith flame using Eq. (7)
20. **else**
21. Choose a random flame as reference flame
22. Update the position of the ith moth with random selected flame using Eq. (7)
23. **end if**
24. Generate binary position of the moth by using transfer function
25. Calculate fitness value of the moth
26. **end foreach**
27. Update global best position
28. //Control Desert Bush strategy
29. **if** global best has better position
30. Reset *DBCounter* //DBCounter = 0
31. **else**
32. Increase *DBCounter* //DBCounter ++
33. **if** *DBCounter* >= *dbMax* //dbMax: Maximum fail number
34. Apply Desert Bush strategy given with Algorithm 2
35. Reset *DBCounter*
36. **end if**
37. **end if**
38. **end while**
39. return *best solution* as output

Algorithm 3. The pseudo code of the binEMFO-DB algorithm

5 Experimental study

In this section, the parameter analyses process and the experimental results obtained were presented comparatively.

5.1 Parameter analyses

Here, the analysis made on the selection of the transfer function to use MFO as a binary algorithm and the analysis made to obtain the optimum values of the specific parameters of the proposed algorithm were presented.

5.1.1 Transfer function analyses for binary MFO

In order to apply the MFO algorithm developed for the solution of continuous problems to a binary problem, continuous values need to be converted into binary. Transfer functions are generally used for this process. It was desired to obtain the best results by applying eight different transfer functions given in Table 2. The algorithm was applied with 30 runs for each transfer function. As a success criterion, the hit value obtained from the total runs was used. Hit is the case of finding the optimum value of the problem studied.

According to the results in Table 3, the binary MFO algorithm achieved a total of 300 hits with the ***S3*** transfer function as a result of 450 runs (problem number × run number). Therefore, the basic binary MFO algorithm was run with the ***S3*** transfer function and the obtained results were used in comparisons.

Table 3. Transfer functions analyses for binary MFO by using hit values.

P/TF	S1	S2	S3	S4	V1	V2	V3	V4
Cap71	30	30	30	30	26	26	30	30
Cap72	30	30	30	30	20	19	30	30
Cap73	30	30	30	30	15	12	30	30
Cap74	30	30	30	30	21	18	27	30
Cap101	29	29	30	30	6	8	17	27
Cap102	29	30	30	30	9	10	14	17
Cap103	27	25	29	27	11	9	15	22
Cap104	29	30	30	30	13	19	20	23
Cap131	5	13	16	7	2	1	0	0
Cap132	11	14	10	2	2	0	0	0
Cap133	6	6	12	3	1	2	1	0
Cap134	19	23	23	16	5	3	2	0
CapA	1	2	0	0	0	0	0	0
CapB	0	0	0	0	0	0	0	0
CapC	0	0	0	0	0	0	0	0
Total Hit	276	292	300	265	131	127	186	209

5.1.2 Parameter analyses of the proposed method

As mentioned in previous sections, the proposed method has some modifications. Based on these modifications some specific

parameters should be determined. The selection of the transfer function, the determination of the chaotic map from which the initial population will be generated, and the maximum number of fail for the applying of the desert bush strategy are the special parameters that the algorithm needs to be optimized. However, population size is also considered as a general parameter that needs to be optimized. Table 4 shows the information about the parameters that should be optimized.

Table 4. Specific parameters of the proposed algorithm.

Description	Sign	Values
Transfer function	*tf*	[1 2 3 4 5 6 7 8]
Chaotic map	*cm*	[1 2 3 4 5 6 7 8]
Maximum fail number for desert bush strategy	*dbMax*	[5 10 15 20 25 30 35 40]
Population size	*N*	[30 40 50 60 70 80 90 100]

The values of the transfer function parameter consist of the options given in Table 2. The list of the chaotic maps was given below in Table 5. The potential values selected for the maximum number of fail parameter were determined as 5 10 15 20 25 under normal conditions. However, due to the use of Taguchi method [95] in parameter analysis, 30 35 40 values have been added to ensure that the level of this parameter is the same as the other two parameters. In most of the studies in the literature, values between 40 and 100 are used as the population number. Therefore, this range was used in this study as well. However, as stated earlier, the value of 30 was added for the population number, since the levels of the parameters must be the same.

Table 5. Chaotic maps.

#	Name
1	Chebyshev map
2	Circle map
3	Gauss/Mouse map
4	Iterative map
5	Logistic map
6	Piecewise map
7	Sine map
8	Singer map

As mentioned above, the number of possible values of *tf*, *cm*, *dbMax* and *N* parameters that need to be optimized is determined as 8, 8, 5 and 7, respectively. The number of all possible combinations for these parameters is 8×8×5×7 = 2240. Applying each combination with 30 runs for 15 problems requires a very long process time. Therefore, a more effective process was followed by using the Taguchi method, which is frequently preferred in parameter analysis and gives successful results [96]. Taguchi's orthogonal array design approach was used in this study. In this approach, the level of each parameter must be equal. That's why; the *dbMax* and *N* parameters have been expanded and made into 8 levels as shown in Table 4. In this case, the total number of combinations is 8×8×8×8 = 4096. With the Taguchi orthogonal array design approach applied in this study, 512 combinations were obtained and tested. Each of these combinations was applied with 30 runs on the 15 problems given in Table 1. In this case, there are 450 (15x30) results for each combination. As the success criterion, the hit value obtained from the total results of the combinations was used. According to the experimental results obtained in the parameter analysis, it was seen that the parameter combination *tf*=S2, *cm*=Piecewise, *dbMax*=5 and *N*=80 achieved the most successful result with 402 total hits. Therefore, the results obtained with this parameter combination were used for the proposed algorithm.

5.1.3 The effect of modifications on the success of the proposed algorithm

In order to analyze the effect of each modification on the success of the proposed algorithm, eight different experimental algorithms were run according to the combinations given in Table 6. The red cross below the modification indicates that the modification was not used in that experimental study, and the green checkmark indicates that it was used. For example, In Exp3, only "random flame selection" modification was applied, "chaotic map" and "desert bush" modifications were not applied. In Exp8, by applying all 3 modifications, the algorithm proposed in this study is obtained.

Eight different experimental algorithms given in Table 6 were run on CAP problems with 30 repetitions for N=80 and 80,000 maximum fitness assessment numbers (*maxFEs*). The average cost and standard deviation values of experimental algorithms for each problem are given in Table 7. In addition, the success rank of the algorithms for each problem and the average success rank achieved over the problem set is also presented in the same table.

Table 6. 8 different experimental algorithms obtained with 3 modifications.

Chaotic map	Random flame selection	Desert bush	
✗	✗	✗	Exp1
✗	✗	✓	Exp2
✗	✓	✗	Exp3
✗	✓	✓	Exp4
✓	✗	✗	Exp5
✓	✗	✓	Exp6
✓	✓	✗	Exp7
✓	✓	✓	Exp8

In Table 7, it is seen that all algorithms can reach optimal solutions for all runs on Cap71-74 and Cap102, Cap104 problems. This can be explained by the fact that these problems are relatively low-dimensional and easy problems. On the Cap101 problem, all algorithms except Exp2 and Exp5 obtain optimal solutions, while algorithms other than Exp1, Exp2 and Exp6 reach optimal solutions on the Cap103 problem. The performances of the algorithms for Cap131-134, which are defined as large type problems, have begun to diverge from each other. While only Exp3 algorithm reach optimal solutions in all runs in Cap131, Exp7 and Exp8 algorithms achieve equal success and share the second place. In Cap132 problem, while Exp3, Exp4, Exp6, Exp7 and Exp8 algorithms reach optimal solutions in all runs; Exp2, Exp5 and Exp1 algorithms ranked as 2nd, 3rd and 4th. places respectively. In Cap133 problem, the Exp8 algorithm takes the first place by obtaining the best mean, while Exp6 and Exp7 share the second place with an equal performance.

Table 7. Mean, standard deviation, and rank values of the experimental algorithms on the CAP problems.

		Exp1	Exp2	Exp3	Exp4	Exp5	Exp6	Exp7	Exp8 (proposed algorithm)
Problem									
	mean	932615.75	932615.75	932615.75	932615.75	932615.75	932615.75	932615.75	932615.75
Cap71	std.	0	0	0	0	0	0	0	0
	rank	1	1	1	1	1	1	1	1
	mean	977799.40	977799.40	977799.40	977799.40	977799.40	977799.40	977799.40	977799.40
Cap72	std.	0	0	0	0	0	0	0	0
	rank	1	1	1	1	1	1	1	1
	mean	1010641.45	1010641.45	1010641.45	1010641.45	1010641.45	1010641.45	1010641.45	1010641.45
Cap73	std.	0	0	0	0	0	0	0	0
	rank	1	1	1	1	1	1	1	1
	mean	1034976.98	1034976.98	1034976.98	1034976.98	1034976.98	1034976.98	1034976.98	1034976.98
Cap74	std.	0	0	0	0	0	0	0	0
	rank	1	1	1	1	1	1	1	1
	mean	796648.44	796705.79	796648.44	796648.44	796677.11	796648.44	796648.44	796648.44
Cap101	std.	0	218.26	0	0	157.07	0	0	0
	rank	1	3	1	1	2	1	1	1
	mean	854704.20	854704.20	854704.20	854704.20	854704.20	854704.20	854704.20	854704.20
Cap102	std.	0	0	0	0	0	0	0	0
	rank	1	1	1	1	1	1	1	1
	mean	893804.72	893884.02	893782.11	893782.11	893782.113	893823.62	893782.113	893782.113
Cap103	std.	68.97	310.94	0	0	0	189.18	0	0
	rank	2	4	1	1	1	3	1	1
	mean	928941.75	928941.75	928941.75	928941.75	928941.75	928941.75	928941.75	928941.75
Cap104	std.	0	0	0	0	0	0	0	0
	rank	1	1	1	1	1	1	1	1
	mean	793692.97	793582.94	793439.56	793496.92	793760.26	793525.59	793468.24	793468.24
Cap131	std.	394.46	326.09	0	218.26	401.97	262.50	157.07	157.07
	rank	6	5	1	3	7	4	2	2
	mean	851560.95	851501.15	851495.33	851495.33	851517.20	851495.33	851495.33	851495.33
Cap132	std.	200.24	31.91	0	0	119.82	0	0	0
	rank	4	2	1	1	3	1	1	1
	mean	893348.90	893249.16	893200.11	893157.71	893322.25	893147.25	893147.25	893134.19
Cap133	std.	450.58	421.56	266.63	251.67	440.48	215.24	215.24	222.59
	rank	7	5	4	3	6	2	2	1
	mean	929129.92	928941.75	928941.75	928941.75	929318.10	928941.75	928941.75	928941.75
Cap134	std.	1030.67	0	0	0	1432.24	0	0	0
	rank	2	1	1	1	3	1	1	1
	mean	17869403.08	17156454.48	17418566.26	17156454.48	17928883.17	17166266.93	17475008.71	17156454.48
CapA	std.	384572.22	0	244881.89	0	386185.06	49391.95	326438.49	0
	rank	5	1	3	1	6	2	4	1
	mean	13327830.12	13034533.46	13244952.36	13026598.76	13361023.17	13045987.73	13261062.51	13024905.67
CapB	std.	160836.07	62284.36	162338.62	49753.05	165772.61	68600.81	129384.60	39614.03
	rank	7	3	5	2	8	4	6	1
	mean	11802434.05	11574746.53	11722427.13	11528700.38	11814749.39	11584708.71	11708865.54	11536635.1
CapC	std.	163366.47	56766.08	150635.73	27875.15	141498.66	82561.52	102017.29	32729.71
	rank	7	3	6	1	8	4	5	2
	Mean rank	3.133	2.200	1.933	1.333	3.333	1.867	1.933	**1.133**

In Cap134 problem, while Exp2, Exp3, Exp4, Exp6, Exp7 and Exp8 algorithms reach optimal solutions in all runs; Exp1 and Exp5 algorithms rank as 2nd and 3rd places respectively. In the CapA problem, while Exp2, Exp4 and Exp8 takes the first place by reaching the optimal solution in all runs; Exp6, Exp3, Exp7, Exp1 and Exp5 achieve 2nd, 3rd, 4th, 5th and 6th places, respectively. In the CapB problem, the Exp8 algorithm takes the first place by achieving the best result, while the Exp4 algorithm performs close to the Exp8 and takes the second place. Finally, in the CapC problem, the Exp4 algorithm takes the first place by obtaining the best result, while the Exp8 algorithm takes the second place with a small difference.

When a general evaluation is made, it is observed that the Exp8 algorithm ranked first in 13 of the 15 problems and achieve the best ranking with a mean success rank of 1.133. While the Exp4 algorithm is the second most successful algorithm with a mean success rank of 1.333; Exp6 takes the 3rd place with a mean success rank of 1.867.

The common point of the 3 algorithms, which are the most successful in order of mean success rank, is the "desert bush" modification. From this point of view, it can be said that the modification that makes the most important contribution to the proposed algorithm is "desert bush". Additionally, Exp5 algorithm with only "chaotic map" modification had a worse result in terms of mean success rank than Exp1 algorithm without any modification. This shows that applying "chaotic map" modification alone does not contribute to the success of the algorithm. The fact that the Exp8 algorithm has a better mean success rank than the Exp4 algorithm shows that the "chaotic map" modification contributes to the algorithm when used together with the "random flame selection" and "desert bush" modifications.

As a result, the values presented in Table 7 clearly show the contribution of all 3 modifications to the proposed algorithm.

5.2 Experimental environment and results

15 different UFL problems taken from OR-Library [80] were used to evaluate the performance of the proposed algorithm. All problems were run for 30 repeats and the *maxFEs* is set as 80,000 for each run to be a fair comparison. Experimental studies were conducted using Matlab 2016 version on Windows 10 64-bit operating system.

The performance comparison was made over the results of studies in the literature that used the same problem set. In the articles compared, it is seen that all or some of the average, standard deviation, best, worst, hit or gap values of the obtained results were used as performance criteria. Here, hit is the number of times the algorithm reaches the optimum solution and gap is the ratio of deviation of the best solution found by the algorithm from the optimum solution. The mathematical expression of the gap is given in Eq. (11).

$$gap = \frac{mean - opt}{opt}\ x\ 100 \tag{11}$$

Here, mean is the average value of all runs and opt is the value of optimum solution for the related problem. Which of these performance criteria are given in the compared study, the same metric values are given for the proposed approach.

5.2.1 Comparison with other algorithms

In Table 8, the proposed algorithm is compared over the best, worst, average and gap values of the SS-EC [47] algorithm. When the results are examined, it is seen that while both algorithms reach optimal results in Cap71, Cap72, Cap73, Cap74, Cap104 and Cap134 problems, the proposed algorithm achieves superior results for other problems. When all the results are compared, the proposed algorithm in 12 of the 15 problems reached the optimal solution in all 30 runs and outperformed the SS-EC method.

In Table 9, the proposed algorithm is compared with the PSO and ABC variants [30] on standard deviation and gap values. Algorithms were ranked according to their performance for each problem. Average achievements are given at the bottom of the table. In addition, *W/D/L* results of the algorithms are also given. Here, win means that the proposed algorithm is more successful, lost means that the compared algorithm is more successful, and draw means that both algorithms perform equally. The Mean-Rank below the table shows the average success rank of each algorithm on the problem set. On the other hand, Final-Rank gives the success order of the algorithms according to the Mean-Rank value. The proposed algorithm achieved the most successful results on the entire problem set and took first place in both the Mean-Rank and Final-Rank rankings. When the obtained results are examined, it is seen that PSO variants suffer from trapping into local-minima even in small size problems. ABC variants, on the other hand, are successful in small and medium-sized problems, while they are stuck in the local minimum for large and huge problems. The proposed *binEMFO-DB* algorithm showed equal or better performance in all problems from the compared algorithms and ranked first in the average ranking.

In Table 10, the proposed algorithm is compared with DE and genetic algorithm (GA) variants [45],[51] on the gap and hit values. When the results are examined, DisDE/rand algorithm has slightly better than the *binEMFO-DB* algorithm by achieving 404 total hits whereas the proposed algorithm has 402 hits. On the other hand, when the gap results are examined, the proposed *binEMFO-DB* method reaches optimal solutions in 12 of the 15 problems in all runs, while the DisDE/rand method reaches optimal solutions in all runs for only 7 problems.

In Table 11, the proposed *binEMFO-DB* algorithm is compared with the recently proposed binAAA and SimLogicTSA algorithms [45],[51] over gap and hit values. All three algorithms showed very successful performances except for capB and capC problems. While SimLogicTSA algorithm achieved 0 hits in capB and capC problems, *binEMFO-DB* achieved 12-2 hits and binAAA achieved 15-1 hits respectively. Looking at the gap metric values, binAAA took the first place with the lowest gap value in the CapB problem, while SimLogicTSA took the second place. In the CapC problem, on the other hand, the proposed approach took the first place with the lowest gap value.

In Table 12, the best, worst, average and gap values obtained by the proposed algorithm and LS approach [97] for each problem are given. When the results are examined, it is seen that the proposed approach and the LS algorithm perform similarly in small-size problems, and that the proposed approach is more successful in medium and large-size problems. Finally, while the proposed approach in the capA problem is more successful, the LS algorithm achieved better results in capB and capC.

Table 8. A Comparison of binEMFO-DB with SS-EC.

	SS-EC				binEMFO-DB			
	Best	Worst	Avg.	Gap	Best	Worst	Avg.	Gap
Cap71	932,615.75	932,615.75	932,615.75	0	932,615.75	932,615.75	932,615.75	0
Cap72	977,799.40	977,799.40	977,799.40	0	977,799.40	977,799.40	977,799.40	0
Cap73	1,010,641.45	1,010,641.45	1,010,641.45	0	1,010,641.45	1,010,641.45	1,010,641.45	0
Cap74	1,034,976.98	1,034,976.98	1,034,976.98	0	1,034,976.97	1,034,976.97	1,034,976.97	0
Cap101	796,648.44	799,593.49	796,746.61	0.012	796,648.43	796,648.43	796,648.43	0
Cap102	854,704.20	855,971.75	854,788.70	0.009	854,704.20	854,704.20	854,704.20	0
Cap103	893,782.11	894,801.16	893,985.92	0.022	893,782.11	893,782.11	893,782.11	0
Cap104	928,941.75	928,941.75	928,941.75	0	928,941.75	928,941.75	928,941.75	0
Cap131	793,439.56	795,883.24	793,787.70	0.043	793,439.56	793,439.56	793,439.56	0
Cap132	851,495.33	851,670.13	851,524.46	0.003	851,495.32	851,495.32	851,495.32	0
Cap133	893,076.71	899,172.51	893,434.25	0.04	893,076.71	894,095.76	893,134.19	0.0064
Cap134	928,941.75	928,941.75	928,941.75	0	928,941.75	928,941.75	928,941.75	0
CapA	17,156,454.48	18,041,168.85	17,215,435.44	0.343	17,156,454.47	17,156,454.47	17,156,454.47	0
CapB	12,979,071.58	13,511,709.68	13,110,151.33	1.01	12,979,071.58	13,081,049.25	13,024,905.67	0.3531
CapC	11,505,594.33	11,867,848.70	11,596,027.44	0.786	11,505,594.33	11,613,592.92	11,536,635.10	0.2698

Table 9. A Comparison of binEMFO-DB with PSO and ABC variants.

Problems		Algorithms					
		BPSO	IBPSO	DisABC	binABC	ABC_{bin}	binEMFO-DB
Cap71	Std. Dev.	0	587.49	0	0	0	0
	GAP(%)	0	0.037	0	0	0	0
	Rank	1	2	1	1	1	1
Cap72	Std. Dev.	0	1,844.64	0	0	0	0
	GAP(%)	0	0.275	0	0	0	0
	Rank	1	2	1	1	1	1
Cap73	Std.Dev.	634.62	1,513.78	0	0	0	0
	GAP(%)	0.024	0.198	0	0	0	0
	Rank	2	3	1	1	1	1
Cap74	Std.Dev.	500.27	4,426.67	0	0	0	0
	GAP(%)	0.009	0.403	0	0	0	0
	Rank	2	3	1	1	1	1
Cap101	Std.Dev.	566.44	3,799.52	0	0	0	0
	GAP(%)	0.046	0.597	0	0	0	0
	Rank	2	3	1	1	1	1
Cap102	Std.Dev.	386.76	3,249.38	0	0	0	0
	GAP(%)	0.015	0.732	0	0	0	0
	Rank	2	3	1	1	1	1
Cap103	Std.Dev.	485.26	4,978.98	0	0	85.67	0
	GAP(%)	0.042	0.641	0	0	0.005	0
	Rank	3	4	1	1	2	1
Cap104	Std.Dev.	1,951.81	10,845.26	0	0	0	0
	GAP(%)	0.081	0.996	0	0	0	0
	Rank	2	3	1	1	1	1
Cap131	Std.Dev.	1,207.63	4,244.29	233,764.00	0	1,065.73	0
	GAP(%)	0.132	2.424	0.62	0	0.197	0
	Rank	2	5	4	1	3	1
Cap132	Std.Dev.	1,196.19	11,569.02	813.37	0	213.28	0
	GAP(%)	0.091	3.601	0.095	0	0.02	0
	Rank	3	5	4	1	2	1
Cap133	Std.Dev.	821.28	14,905.27	359.03	200.24	561.34	222.59
	GAP(%)	0.112	5.263	0.031	0.122	0.075	0.0064
	Rank	4	6	2	5	3	1
Cap134	Std.Dev.	2,285.42	15,788.86	0	0	0	0
	GAP(%)	0.135	7.634	0	0	0	0
	Rank	2	3	1	1	1	1
CapA	Std.Dev.	374,302.81	3,357,138.19	74,782.61	236,833.50	268,685.20	0
	GAP(%)	2.179	137.886	0.152	2.509	3.172	0
	Rank	3	6	2	4	5	1
CapB	Std.Dev.	176,206.07	1,406,575.70	109,738.50	91,430.13	88,452.80	41,926.41
	GAP(%)	1.949	55.27	3.303	2.508	2.815	0.3531
	Rank	2	6	5	3	4	1
CapC	Std.Dev.	92,977.85	1,245,252.20	95,778.78	82,312.70	78,162.20	32,729.70
	GAP(%)	1.487	45.556	4.697	2.58	2.037	0.2698
	Rank	2	6	5	4	3	1
Mean-Rank		2.2	4	2.06	1.8	2	**1**
Final-Rank		5	6	4	2	3	**1**
W/D/L		13/2/0	15/0/0	6/9/0	4/11/0	7/8/0	

Table 10. A Comparison of binEMFO-DB with DE and GA variants.

Problem	DisDE/rand		binDE		GA-SP		GA-TP		GA-UP		GA-EC		binEMFO-DB	
	Gap	Hit	Gap	Hit	Gap	Hit	Gap	Hit	Gap	Hit	Gap	Hit	Gap	Hit
Cap71	0	30	0	30	0	30	0	30	0	30	0	30	0	30
Cap72	0	30	0	30	0	30	0	30	0	30	0	30	0	30
Cap73	0	30	0	30	0.0666	19	0.0484	22	0.0424	23	0	30	0	30
Cap74	0	30	0	30	0	30	0	30	0	30	0	30	0	30
Cap101	0.0036	29	0	30	0.0684	11	0.0648	12	0.0576	14	0.0072	28	0	30
Cap102	0.0049	29	0	30	0	30	0	30	0	30	0	30	0	30
Cap103	0.0055	27	0	30	0.0637	6	0.0612	10	0.0722	9	0.0067	22	0	30
Cap104	0	30	0	30	0	30	0	30	0	30	0	30	0	30
Cap131	0.0036	29	0.0036	29	0.0681	16	0.0723	14	0.0536	15	0.0608	15	0	30
Cap132	0	30	0.005	29	0	30	0	30	0.0026	29	0.0006	29	0	30
Cap133	0.0138	25	0.0138	24	0.0913	10	0.0744	12	0.082	9	0.0406	15	0.0064	28
Cap134	0	30	0	30	0	30	0	30	0	30	0	30	0	30
CapA	0.037	29	1.3	8	0.0461	24	0.2835	24	0.0604	24	0	30	0	30
CapB	0.189	18	1.52	0	0.5839	9	0.6507	11	0.9905	3	0.4092	11	0.3531	12
CapC	0.0909	8	1.55	0	0.7049	2	0.6276	0	0.6345	0	0.1563	5	0.2698	2
Total Hit		404		360		307		315		306		365		402

Table 11. A comparison of binEMFO-DB with SimLogicTSA and binAAA.

	SimLogicTSA		binAAA		binEMFO-DB	
	Gap	Hit	Gap	Hit	Gap	Hit
Cap71	0	30	0	30	0	30
Cap72	0	30	0	30	0	30
Cap73	0	30	0	30	0	30
Cap74	0	30	0	30	0	30
Cap101	0	30	0	30	0	30
Cap102	0	30	0	30	0	30
Cap103	0	30	0	30	0	30
Cap104	0	30	0	30	0	30
Cap131	0	30	0	30	0	30
Cap132	0	30	0	30	0	30
Cap133	0	30	0	30	0.0064	28
Cap134	0	30	0	30	0	30
CapA	0	30	0	30	0	30
CapB	0.3176	0	0.2478	15	0.3531	12
CapC	0.412	0	0.2946	1	0.2698	2
Total Hit		390		406		402

Table 12. A comparison of binEMFO-DB and LS algorithm.

	binEMFO-DB				LS			
	Best	Worst	Avg.	Gap	Best	Worst	Avg.	Gap
Cap71	932,615.75	932,615.75	932,615.75	0	932,615.75	932,615.75	932,615.75	0
Cap72	977,799.40	977,799.40	977,799.40	0	977,799.40	977,799.40	977,799.40	0
Cap73	1,010,641.45	1,010,641.45	1,010,641.45	0	1,010,641.45	1,010,641.45	1,010,641.45	0
Cap74	1,034,976.97	1,034,976.97	1,034,976.97	0	1,034,976.97	1,034,976.98	1,034,976.98	0
Cap101	796,648.43	796,648.43	796,648.43	0	796,648.43	799,144.69	796,733.62	0.01
Cap102	854,704.20	854,704.20	854,704.20	0	854,704.20	855,971.75	854,716.88	0.001
Cap103	893,782.11	893,782.11	893,782.11	0	893,782.11	894,801.16	893,831.92	0.005
Cap104	928,941.75	928,941.75	928,941.75	0	928,941.75	934,586.98	929,111.11	0.018
Cap131	793,439.56	793,439.56	793,439.56	0	793,439.56	795,883.24	793,567.23	0.016
Cap132	851,495.32	851,495.32	851,495.32	0	851,495.32	851,495.33	851,495.33	0
Cap133	893,076.71	893,076.71	893,076.71	0.0064	893,076.71	893,782.11	893,182.03	0.011
Cap134	928,941.75	928,941.75	928,941.75	0	928,941.75	934,586.98	929,506.27	0.06
CapA	17,156,454.47	17,156,454.47	17,156,454.47	0	17,156,454.47	17,665,889.11	17,163,692.65	0.042
CapB	12,979,071.58	13,081,049.25	13,024,905.67	0.3531	12,979,071.58	13,215,550.80	13,014,256.16	0.271
CapC	11,505,594.32	11,613,592.92	11,536,635.10	0.2698	11,505,594.32	11,615,301.64	11,525,439.63	0.172
W/D/L					8/5/2			

6 Conclusions

It is possible to apply the proposed algorithms to solve continuous problems to binary problems with two different options. If the structure of the algorithm is suitable, the algorithm can be directly adapted to the problem in binary. However, algorithms that cannot be used directly in binary, such as the MFO algorithm we used in this study, can be binarized by applying transfer functions. In this case, choosing the right transfer function is an important point to consider. However, one of the weak points of this approach is that the solutions of the members in the population are kept as continuous values and position updates are made over continuous values. In this study, when the MFO algorithm is converted into binary with only the transfer function and used (Table 3), the best result achieved was to catch the 300 best out of 450 cases. However, with the modifications we have applied, the performance of the algorithm has increased significantly, and it has moved to the position of catching the 402 best out of 450. This shows that the applied modifications were quite successful. In this study, the MFO algorithm which is modelled based on the nocturnal flight strategy of moths was binarized and applied on UFLP. Since, the basic MFO is a continuous algorithm, transfer function was used for binarize process. In order to increase the performance of the binary MFO; some modifications were used, such as generating the initial population with a chaotic map, ensuring the diversity of flame selection in the position update phase, and by using the desert bush strategy providing the resurrection of the population that could not progress. The proposed algorithm (*binEMFO-DB*) applied on a problem set that consists of 15 problems with different size types was used. The performance of the proposed algorithm was compared with a set of algorithms which are frequently used in literature by using gap, hit and mean values. The experimental results show that the proposed algorithm is generally successful on UFLP and has better scores when compared with the performance of the other algorithms. On the other hand, when the results of the proposed algorithm are evaluated within itself, it is seen that it is quite successful in small, medium and large sized problems, but it can be improved in huge sized problems such as CapB and CapC.

For the future works, the proposed algorithm can be applied on different binary optimization problems such as knapsack, future selection, job scheduling, resource allocation in cloud computing etc. On the other hand, different search strategies can be applied to improve the performance of the proposed algorithm for binary problems.

7 Author contribution statement

In this study, both Ahmet ÖZKIŞ and Murat KARAKOYUN focused on forming the idea, conducting experimental studies, evaluating the results, contributing to the literature review, spelling, and checking the article's content.

8 Ethics committee approval and conflict of interest statement

There is no need for an ethics committee approval in the prepared article. There is no conflict of interest with any person/institution in the prepared article.

9 References

[1] Babalik A, Cinar AC, Kiran MS. "A modification of tree-seed algorithm using Deb's rules for constrained optimization". *Applied Soft Computing*, 63, 289-305, 2018.

[2] Yuan X, Nie H, Su A, Wang L, Yuan Y. "An improved binary particle swarm optimization for unit commitment problem". *Expert Systems with Applications*, 36(4), 8049-8055, 2009.

[3] Van Beers F, Lindström A, Okafor E,Wiering MA. "Deep Neural Networks with Intersection over Union Loss for Binary Image Segmentation". *ICPRAM. Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*, Prague, Czech Republic, 19-21 February 2019.

[4] Banitalebi A, Aziz MIA, Aziz ZA. "A self-adaptive binary differential evolution algorithm for large scale binary optimization problems". *Information Sciences*, 367, 487-511, 2016.

[5] Ayaz HI, Ozturk ZK. "A mathematical model and a heuristic approach for train seat scheduling to minimize dwell time". *Computers & Industrial Engineering*, 160, 1-7, 2021.

[6] Aslan M, Gunduz M, Kiran MS. "JayaX: Jaya algorithm with xor operator for binary optimization". *Applied Soft Computing*, 82, 1-17, 2019.

[7] Baş E, Ülker E. "A binary social spider algorithm for uncapacitated facility location problem". *Expert Systems with Applications*, 161, 1-27, 2020.

[8] Rizk-Allah RM, Hassanien AE, Elhoseny M, Gunasekaran M. "A new binary salp swarm algorithm: development and application for optimization tasks". *Neural Computing and Applications*, 31(5), 1641-1663, 2019.

[9] Eberhart R, Kennedy J. "A new optimizer using particle swarm theory". *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, 04-06 October 1995.

[10] Mirjalili S, Mirjalili SM,Lewis A. "Grey Wolf Optimizer". *Advances in Engineering Software*, 69, 46-61, 2014.

[11] Karaboga D, Akay B. "A comparative study of artificial bee colony algorithm". *Applied mathematics and computation*, 214(1), 108-132, 2009.

[12] Uymaz SA, Tezel G, Yel E. "Artificial algae algorithm (AAA) for nonlinear global optimization", *Applied Soft Computing*, 31, 153-171, 2015.

[13] Kennedy J, Eberhart RC. "A discrete binary version of the particle swarm algorithm". *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, Orlando, USA, 12-15 October 1997.

[14] Beheshti Z, Shamsuddin SM, Hasan S. "Memetic binary particle swarm optimization for discrete optimization problems". *Information Sciences*, 299, 58-84, 2015.

[15] Mirjalili S, Lewis A. "S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization". *Swarm and Evolutionary Computation*, 9, 1-14, 2013.

[16] Pampara G, Engelbrecht AP, Franken N. "Binary Differential Evolution". *2006 IEEE International Conference on Evolutionary Computation*, Vancouver, Canada, 16-21 July 2006.

[17] Lim SM, Sultan ABM, Sulaiman MN, Mustapha A, Leong KY. "Crossover and mutation operators of genetic algorithms". *International Journal of Machine Learning and Computing*, 7(1), 9-12, 2017.

[18] Braun H. "On solving travelling salesman problems by genetic algorithms". *International Conference on Parallel Problem Solving from Nature*, Dortmund, Germany, 01-03 October 1990.

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

[20] Iwasaki Y, Kusne AG, Takeuchi I. "Comparison of dissimilarity measures for cluster analysis of X-ray diffraction data from combinatorial libraries". *NPJ Computational Materials*, 3(1), 1-9, 2017.

[21] Chen Y, Xie W,Zou X. "A binary differential evolution algorithm learning from explored solutions". *Neurocomputing*, 149, 1038-1047, 2015.

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

[23] Mojtaba Ahmadieh K, Mohammad T, Mahdi Aliyari S. "A novel binary particle swarm optimization". *2007 Mediterranean Conference on Control & Automation*, Athens, Greece, 27-29 June 2007.

[24] Lin JC-W, Yang L, Fournier-Viger P, Hong T-P,Voznak M. "A binary PSO approach to mine high-utility itemsets". *Soft Computing*, 21(17), 5103-5121, 2017.

[25] Nezamabadi-pour H, Rostami-Shahrbabaki M, Maghfoori-Farsangi M. "Binary particle swarm optimization: challenges and new solutions". *CSI Journal on Computer Science and Engineering*, 6(1), 21-32, 2008.

[26] Guner AR, Sevkli M. "A discrete particle swarm optimization algorithm for uncapacitated facility location problem". *Journal of Artificial Evolution and Applications,* 1, 1-9, 2008.

[27] Saha S, Kole A, Dey K. "A modified continuous particle swarm optimization algorithm for uncapacitated facility location problem". *International Conference on Advances in Information Technology and Mobile Communication*, Nagpur, India, 21-22 April 2011.

[28] Karaboga D. "An İdea Based on Honey Bee Swarm for Numerical Optimization". Department of Computer Engineering, Erciyes University, Kayseri, Turkey, Technical Report-TR06, 2005.

[29] Kashan MH, Nahavandi N, Kashan AH. "DisABC: a new artificial bee colony algorithm for binary optimization". *Applied Soft Computing*, 12(1), 342-352, 2012.

[30] Kiran MS, Gündüz M. "XOR-based artificial bee colony algorithm for binary optimization". *Turkish Journal of Electrical Engineering and Computer Sciences*, 21(8), 2307-2328, 2013.

[31] Kiran MS. "A binary artificial bee colony algorithm and its performance assessment". *Expert Systems with Applications*, 175, 1-15, 2021.

[32] Jia D, Duan X, Khan MK. "Binary Artificial Bee Colony optimization using bitwise operation". *Computers & Industrial Engineering*, 76, 360-365, 2014.

[33] Storn R, Price K. "differential evolution-a simple and efficient heuristic for global optimization over continuous spaces". *Journal of Global Optimization*, 11(4), 341-359, 1997.

[34] Engelbrecht AP, Pampara G. "Binary differential evolution strategies". *2007 IEEE Congress on Evolutionary Computation*, Singapore, 25-28 September 2007.

[35] Su H, Yang Y. "Quantum-Inspired Differential Evolution for Binary Optimization". *2008 Fourth International Conference on Natural Computation*, Jinan, China, 18-20 October 2008.
[36] He X, Zhang Q, Sun N, Dong Y. "Feature Selection with Discrete Binary Differential Evolution". *2009 International Conference on Artificial Intelligence and Computational Intelligence*, Shanghai, China, 7-8 November 2009.
[37] Deng C, Zhao B, Yang Y, Deng A. "Novel Binary Differential Evolution Algorithm for Discrete Optimization". *2009 Fifth International Conference on Natural Computation*, Tianjian, China, 14-16 August 2009.
[38] Qingyun Y. "A comparative study of discrete differential evolution on binary constraint satisfaction problems". *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, Hong Kong, 01-06 June 2008.
[39] Wang L, Fu X, Menhas MI, Fei M. "A Modified Binary Differential Evolution Algorithm". *International Conference on Life System Modeling and Simulation*, Wuxi, China, 17-20 September 2010.
[40] Kashan MH, Kashan AH, Nahavandi N. "A novel differential evolution algorithm for binary optimization". *Computational Optimization and Applications,* 55(2), 481-513, 2013.
[41] Rashedi E, Nezamabadi-pour H, Saryazdi S. "GSA: A Gravitational Search Algorithm". *Information Sciences*, 179(13), 2232-2248, 2009.
[42] Rashedi E, Nezamabadi-pour H,Saryazdi S. "BGSA: binary gravitational search algorithm". *Natural Computing*, 9(3), 727-745, 2010.
[43] Khanesar MA, Branson D. "XOR Binary Gravitational Search Algorithm". *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Bari, Italy, 06-09 October 2019.
[44] Rao R. "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems". *International Journal of Industrial Engineering Computations*, 7(1), 19-34, 2016.
[45] Cinar AC, Kiran MS. "Similarity and logic gate-based tree-seed algorithms for binary optimization". *Computers & Industrial Engineering*, 115, 631-646, 2018.
[46] Kiran MS. "TSA: Tree-seed algorithm for continuous optimization". *Expert Systems with Applications*, 42(19), 6686-6698, 2015.
[47] Hakli H, Ortacay Z. "An improved scatter search algorithm for the uncapacitated facility location problem". *Computers & Industrial Engineering*, 135, 855-867, 2019.
[48] Yu JJQ, Li VOK. "A social spider algorithm for global optimization". *Applied Soft Computing*, 30, 614-627, 2015.
[49] Baş E, ÜLker E. "An efficient binary social spider algorithm for feature selection problem". *Expert Systems with Applications*, 146, 1-25, 2020.
[50] Baş E, Ülker E. "A binary social spider algorithm for continuous optimization task". *Soft Computing*, 24(17), 12953-12979, 2020.
[51] Korkmaz S, Babalik A, Kiran MS. "An artificial algae algorithm for solving binary optimization problems". *International Journal of Machine Learning and Cybernetics*, 9(7), 1233-1247, 2018.
[52] Çınar AC. "A Comprehensive Comparison of Binary Archimedes Optimization Algorithms on Uncapacitated Facility Location Problems". *Düzce University Journal of Science and Technology*, 10(1), 27-38, 2022.
[53] Karakoyun M, Ozkis A. "A binary tree seed algorithm with selection-based local search mechanism for huge-sized optimization problems". *Applied Soft Computing*, 129, 1-16, 2022.
[54] Sörensen K. "Metaheuristics—the metaphor exposed". *International Transactions in Operational Research*, 22(1), 3-18, 2015.
[55] Wolpert DH, Macready WG. "No free lunch theorems for optimization". *IEEE Transactions on Evolutionary Computation*, 1(1), 67-82, 1997.
[56] Mirjalili S. "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm". *Knowledge-Based Systems*, 89, 228-249, 2015.
[57] Shehab M, Abualigah L, Al Hamad H, Alabool H, Alshinwan M, Khasawneh AM. "Moth–flame optimization algorithm: variants and applications". *Neural Computing and Applications*, 32(14), 9859-9884, 2020.
[58] Zawbaa HM, Emary E, Parv B, Sharawi M. "Feature selection approach based on moth-flame optimization algorithm". *2016 IEEE Congress on Evolutionary Computation (CEC)*, Vancouver, Canada, 24-29 July 2016.
[59] Aziz MAE, Ewees AA, Hassanien AE. "Whale Optimization Algorithm and Moth-Flame Optimization for multilevel thresholding image segmentation". *Expert Systems with Applications*, 83, 242-256, 2017.
[60] Wang M, Chen H, Yang B, Zhao X, Hu L, Cai Z, Huang H, Tong C. "Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses". *Neurocomputing*, 267, 69-84, 2017.
[61] Yousri DA, AbdelAty AM, Said LA, AboBakr A, Radwan AG. "Biological inspired optimization algorithms for cole-impedance parameters identification". *AEU - International Journal of Electronics and Communications*, 78, 79-89, 2017.
[62] Allam D, Yousri DA, Eteiba MB. "Parameters extraction of the three diode model for the multi-crystalline solar cell/module using Moth-Flame Optimization Algorithm". *Energy Conversion and Management*, 123, 535-548, 2016.
[63] Hazir E, Erdinler ES, Koc KH. "Optimization of CNC cutting parameters using design of experiment (DOE) and desirability function". *Journal of Forestry Research*, 29(5), 1423-1434, 2018.
[64] Elsakaan AA, El-Sehiemy RA, Kaddah SS, Elsaid MI. "An enhanced moth-flame optimizer for solving non-smooth economic dispatch problems with emissions". *Energy*, 157, 1063-1078, 2018.
[65] Jangir N, Pandya MH, Trivedi IN, Bhesdadiya RH, Jangir P, Kumar A. "Moth-Flame optimization Algorithm for solving real challenging constrained engineering optimization problems". *2016 IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, Bhopal, India, 05-06 March 2016.
[66] Trivedi IN, Kumar A, Ranpariya AH, Jangir P. "Economic Load Dispatch problem with ramp rate limits and prohibited operating zones solve using Levy flight Moth-Flame optimizer". *2016 International Conference on Energy Efficient Technologies for Sustainability (ICEETS)*, Nagercoil, India, 07-08 April 2016.

[67] Li WK, Wang WL, Li L. "Optimization of water resources utilization by multi-objective moth-flame algorithm". *Water Resources Management*, 32(10), 3303-3316, 2018.
[68] Savsani V, Tawhid MA. "Non-dominated sorting moth flame optimization (NS-MFO) for multi-objective problems". *Engineering Applications of Artificial Intelligence*, 63, 20-32, 2017.
[69] Nanda SJ. "Multi-objective moth flame optimization". 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, India, 21-24 September 2016.
[70] K SR, Panwar LK, Panigrahi BK, Kumar R. "Solution to unit commitment in power system operation planning using binary coded modified moth flame optimization algorithm (BMMFOA): A flame selection based computational technique". *Journal of Computational Science*, 25, 298-317, 2018.
[71] Abdel-mawgoud H, Kamel S, Ebeed M, Youssef A. "Optimal allocation of renewable dg sources in distribution networks considering load growth". *2017 Nineteenth International Middle East Power Systems Conference (MEPCON)*, Cairo, Egypt, 19-21 December 2017.
[72] Anfal M, Abdelhafid H. "Optimal placement of PMUs in Algerian network using a hybrid particle swarm–moth flame optimizer (PSO-MFO)". *Electrotehnica, Electronica, Automatica (EEA),* 65(3), 191-196, 2017.
[73] Bhesdadiya R, Trivedi IN, Jangir P, Kumar A, Jangir N, Totlani R. "A novel hybrid approach particle swarm optimizer with moth-flame optimizer algorithm". *Advances in Computer and Computational Sciences*, Ajmer, India, 12-13 August 2016.
[74] Jangir P. "Optimal power flow using a hybrid particle swarm optimizer with moth flame optimizer". *Global J Res Eng*, 17, 524-542, 2017.
[75] Kamalapathi K, Priyadarshi N, Padmanaban S, Holm-Nielsen JB, Azam F, Umayal C, Ramachandaramurthy VK. "A hybrid moth-flame fuzzy logic controller based ıntegrated cuk converter fed brushless dc motor for power factor correction". *Electronics,* 7(11), 1-19, 2018.
[76] Khalilpourazari S, Khalilpourazary S. "An efficient hybrid algorithm based on Water Cycle and Moth-Flame Optimization algorithms for solving numerical and constrained engineering optimization problems". *Soft Computing*, 23(5), 1699-1722, 2019.
[77] Sarma A, Bhutani A, Goel L. "Hybridization of moth flame optimization and gravitational search algorithm and its application to detection of food quality". *2017 Intelligent Systems Conference (IntelliSys)*, London, UK, 07-08 September 2017.
[78] Sayed GI, Hassanien AE. "A hybrid SA-MFO algorithm for function optimization and engineering design problems". *Complex & Intelligent Systems*, 4(3), 195-212, 2018.
[79] Yang W, Wang J, Wang R. "Research and application of a novel hybrid model based on data selection and artificial ıntelligence algorithm for short term load forecasting". *Entropy,* 19(2), 1-27, 2017.
[80] Beasley JE. "OR-Library: Distributing test problems by electronic mail". *Journal of the Operational Research Society*, 41(11), 1069-1072, 1990.
[81] Cornuéjols G, Nemhauser G, Wolsey L. "The uncapicitated facility location problem". Cornell University Operations Research and Industrial Engineering, Technical Report, 605, 1983.
[82] Jakob K, Pruzan PM. "The simple plant location problem: Survey and synthesis". *European journal of operational research*, 12, 36-81, 1983.
[83] Monabbati E, Kakhki HT. "On a class of subadditive duals for the uncapacitated facility location problem". *Applied Mathematics and Computation*, 251, 118-131, 2015.
[84] Glover F, Hanafi S, Guemri O, Crevits I. "A simple multi-wave algorithm for the uncapacitated facility location problem". *Frontiers of engineering management*, 5(4), 451-465, 2018.
[85] Kole A, Chakrabarti P, Bhattacharyya S. "An ant colony optimization algorithm for uncapacitated facility location problem". *Proceedings of the 38th International Conference on Computers and Industrial Engineering*, Beijing, China, Oct.-Nov. 2013.
[86] Tuncbilek N, Tasgetiren F, Esnaf S. "Artificial Bee Colony Optimization Algorithm for Uncapacitated Facility Location Problems". *Journal of Economic & Social Research*, 14(1), 1-24, 2012.
[87] Li Y, Zhu X, Liu J. "An improved moth-flame optimization algorithm for engineering problems". *Symmetry*, 12(8), 1-30, 2020.
[88] Pelusi D, Mascella R, Tallini L, Nayak J, Naik B, Deng Y. "An Improved Moth-Flame Optimization algorithm with hybrid search phase". *Knowledge-Based Systems*, 191, 1-14, 2020.
[89] Mafarja M, Aljarah I, Heidari AA, Faris H, Fournier-Viger P, Li X, Mirjalili S. "Binary dragonfly optimization for feature selection using time-varying transfer functions". *Knowledge-Based Systems*, 161, 185-204, 2018.
[90] Omidvar R, Parvin H, Eskandari A. "A clustering approach by SSPCO optimization algorithm based on chaotic initial population". *Journal of Electrical and Computer Engineering Innovations (JECEI)*, 4(1), 31-38, 2016.
[91] Ebrahimzadeh R, Jampour M. "Chaotic genetic algorithm based on lorenz chaotic system for optimization problems". *International Journal of Intelligent Systems and Applications*, 5(5), 19-24, 2013.
[92] Gao W-f, Liu S-y, Huang L-l. "Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique". *Communications in Nonlinear Science and Numerical Simulation*, 17(11), 4316-4327, 2012.
[93] Rafsanjani A, Brulé V, Western TL, Pasini D. "Hydro-responsive curling of the resurrection plant Selaginella lepidophylla". *Scientific Reports*, 5(1), 1-7, 2015.
[94] Mickel JT, Smith AR. *The Pteridophytes of Mexico (Memoirs of the New York Botanical Garden)*. New York, USA, New York Botanical Garden Press, 2004.
[95] Rosa JL, Robin A, Silva M, Baldan CA, Peres MP. "Electrodeposition of copper on titanium wires: Taguchi experimental design approach". *Journal of materials processing technology*, 209(3), 1181-1188, 2009.
[96] Ansari NA, Sharma A, Singh Y. "Performance and emission analysis of a diesel engine implementing polanga biodiesel and optimization using Taguchi method". *Process Safety and Environmental Protection*, 120, 146-154, 2018.
[97] Cura T. "A parallel local search approach to solving the uncapacitated warehouse location problem". *Computers & Industrial Engineering*, 59(4), 1000-1009, 2010.