Neural Computing and Applications (2022) 34:11063–11082
https://doi.org/10.1007/s00521-022-07058-y

ORIGINAL ARTICLE

BinGSO: galactic swarm optimization powered by binary artificial algae algorithm for solving uncapacitated facility location problems

Ersin Kaya[1]

Received: 3 September 2021 / Accepted: 5 February 2022 / Published online: 3 March 2022

Abstract

Population-based optimization methods are frequently used in solving real-world problems because they can solve complex problems in a reasonable time and at an acceptable level of accuracy. Many optimization methods in the literature are either directly used or their binary versions are adapted to solve binary optimization problems. One of the biggest challenges faced by both binary and continuous optimization methods is the balance of exploration and exploitation. This balance should be well established to reach the optimum solution. At this point, the galactic swarm optimization (GSO) framework, which uses traditional optimization methods, stands out. In this study, the binary galactic swarm optimization (BinGSO) approach using binary artificial algae algorithm as the main search algorithm in GSO is proposed. The performance of the proposed binary approach has been performed on uncapacitated facility location problems (UFLPs), which is a complex problem due to its NP-hard structure. The parameter analysis of the BinGSO method was performed using the 15 Cap problems. Then, the BinGSO method was compared with both traditional binary optimization methods and the state-of-the-art methods which are used on Cap problems. Finally, the performance of the BinGSO method on the M^* problems was examined. The results of the proposed approach on the M^* problem set were compared with the results of the state-of-the-art methods. The results of the evaluation process showed that the BinGSO method is more successful than other methods through its ability to establish the balance between exploration and exploitation in UFLPs.

Keywords Galactic swarm optimization · Binary optimization · Uncapacitated facility location problems · Binary artificial algae algorithm

1 Introduction

"Optimization is the art of making good decisions" [1]. Optimization methods realize the process of searching for the optimal solution in the defined solution space of optimization problems. These problems have different characteristics, such as constrained [2], discrete [3], and continuous [4]. Optimization problems can be expensive in terms of computation or hard to solve with classical methods due to the numerous possible solutions. At this point, evolutionary-based or swarm-based methods emerge to obtain an acceptable solution in a reasonable time with limited resources. Binary optimization methods are kinds of discrete optimization in which possible solutions are expressed by 0 and 1 elements. Many binary optimization methods have been introduced that are used to solve real-world problems, such as bin packing [5], feature selection, knapsack [6] and facility location problems [7].

Various modifications have been made to existing optimization methods for solving binary optimization problems. Simplified Binary Harmony Search Algorithm (SBHS) [8], Binary Whale Optimization Algorithm (bWOA) [9], Binary Learning Differential Evolution Algorithm (BLDE) [10], Binary Hybrid Topology Particle Swarm Optimization Algorithm (BHTPSO-QI) [11], Binary Emperor Penguin Optimizer (BEPO) [12], Binary Hybrid Particle Swarm Optimization with Wavelet Mutation (HPSOWM) [13], Binary Fruit Fly Optimization Algorithm (bFOA) [14], Genetic Operators Based Artificial Bee Colony Algorithm (GBABC) [15], Binary Gray Wolf

Ersin Kaya
ekaya@ktun.edu.tr

[1] Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Konya Technical University, G Block G-339, 42250 Konya, Turkey

Optimization (bGWO) [16, 17], Binary Artificial Bee Colony [15, 18, 19], Binary Social Spider Algorithm (BinSSA) [20], Improved Scatter Search algorithm (ISS) [21] and Binary Quantum-Inspired Gravitational Search Algorithm (BQIGSA) [22] are well known modified methods to solve binary optimization problems in the literature.

Location problems are one of the most frequently encountered problems in the field of operational research. The basic components of the location problems are the areas where facilities can be established and the customers whose requirements must be satisfied by these facilities. Location problems in which facility capacities are considered unlimited are called uncapacitated facility location problem (UFLP). There are two types of costs in UFLPs; the cost of establishing a facility and the cost of transportation of the customer requirement. The main aim of UFLPs is to determine the facility areas to satisfy customer requirements with minimum cost [23]. Since UFLPs are NP-Hard problems and the opening and non-opening states of the facility can be expressed as 1 and 0, respectively, many population-based binary optimization methods can be easily applied for solving the UFLPs, and so presented in the literature. There are 2^n possible solutions for n facility. The increase on the number of areas causes an exponential increase on the number of possible solutions.

There are problem-dependent classical methods for solving binary optimization problems in the literature (Enumeration Scheme [24], Enumeration methods [25] and Branch and Bound method [26]). In addition, the increase in the dimension of the problem exponentially increases the calculation and memory costs of the methods. Researchers have trended toward population-based methods in recent years to get rid of these disadvantages. Genetic algorithms are frequently used for binary optimization problems due to the compatibility of both representations of candidate solutions and the generation of new candidate solutions by their operators [27]. Particle Swarm Optimization (PSO) method is designed to solve continuous optimization problems. However, the ability to solve binary problems has been gained by using transfer functions [28]. Greistorfer et al. proposed a method to solve the facility location problems with the filter-and-fan approach [29]. The Scatter Search algorithm, modified with different crossover methods, presents effective results in the UFLPs [21]. Cinar and Kiran proposed an approach for solving binary optimization problems using logic gates and various similarity techniques in the TSA algorithm, which they proposed for continuous problems [30]. Gunduz and Kiran proposed a binary version of the Artificial Bee Colony (ABC) Optimization method [19]. Korkmaz et al. proposed a method that can solve binary optimization problems, and the proposed method has been evaluated on facility location problems [31]. Artificial Algae Algorithm with stigmergic behavior (binAAA) has been successfully implemented to the UFLPs [32]. Genetic algorithms have also been used for solving UFLPs [33]. Binary Social Spider Algorithm (BinSSA), a binary version of the Social Spider Algorithm (SSA), has been proposed by Bas et al. [20], and the algorithm was evaluated for solving binary problems.

There are various algorithms in different disciplines inspired by the biological behavior of living things, called bio-inspired (Bio-inspired P2P Information Systems [34, 35], Bio-inspired networking [36], and Bio-inspired materials chemistry [37]). Bio-inspired methods are also widely used in solving optimization problems such as Particle Swarm Optimization [38], Ant Colony Optimization [39], Artificial Bee Colony [40], and Firefly Algorithm [41]. Artificial Algae Algorithm (AAA) inspired by the behavior of the algae is a promising optimization method introduced in recent years. In particular, AAA performs an effective search in the solution space through its Helical Movement, Reproduction, and Adaptation phases [42]. Due to the success of AAA in continuous optimization problems, several binary AAA versions have been introduced in literature (Artificial Algae Algorithm [31], Artificial Algae Algorithm with stigmergic behavior [32], and binary artificial algae algorithm [43]), and these studies demonstrate successful results.

One of the main challenges in optimization methods is the balance between exploration and exploitation. While exploration refers to the capability of the optimization method on investigating the solution space, exploitation refers to the capability of the optimization method on improving the best available solution [44]. Galactic swarm optimization (GSO) is a framework that successfully performs exploration at the first phase and exploitation at the second phase. GSO is a framework that solves optimization problems using basic optimization algorithms (GA, ABC, PSO, etc.). In the base GSO version, PSO is used as the basic optimization algorithm [45]. In the literature, there are some studies in which different basic optimization algorithms are used in the GSO framework [46, 47]. In these studies, the GSO framework achieves successful results in optimization problems thanks to its exploration and exploitation balance.

This study presents the proposed method using binary AAA on both phases of the GSO framework (BinGSO). The proposed BinGSO method has been evaluated by using two different UFLP sets. These sets include Cap problems consisting of 15 problems and $M*$ problems consisting of 20 problems. A parameter analysis was performed on the evaluation process to determine the optimal parameters of the BinGSO method, which is utilized for solving UFLPs. Afterward, the results obtained from the evaluations of both Cap and M^* problems were compared with the results of the state-of-the-art methods used in solving UFL Problems in the literature. When the results were analyzed by employing statistical tests, it was observed that the BinGSO method achieved more successful results than the results of the state-of-the-art and traditional binary methods in both Cap and M^* problems.

The rest of this paper is organized as follows: Sect. 2 briefly introduces the AAA algorithm, GSO framework, and UFLPs. Section 3 provides details of the proposed approach. Experimental results are presented and analyzed in Sect. 4. The study is finalized in Sect. 5 by providing results and future work.

1.1 Main contribution of the study

As in continuous optimization problems, the major challenge in binary optimization problems is to balance exploration and exploitation. At this point, the GSO method is presented as a two-phase framework by providing a better balance of the exploration and exploitation in its phases. It has been shown in studies [46, 47] that the GSO framework improves the performance of traditional optimization algorithms that solve continuous optimization problems when used as search algorithms in the GSO framework. When the literature was reviewed in detail, it has been determined that the GSO framework was not used in the solution of binary optimization problems. The main motivation of the study is to transfer the advantages of the GSO framework in continuous optimization problems to binary optimization problems. In this context, an approach that effectively solves binary optimization problems by hybridizing the GSO framework and binary AAA algorithm is presented. The proposed method has the ability to search and improve solutions thanks to its effective exploration and exploitation balance. The performance of the proposed binary optimization approach was analyzed using 35 different UFL problems (15 Cap and 20M^*).

2 Preliminaries

2.1 Artificial algae algorithm

Artificial Algae Algorithm (AAA) is a bio-inspired optimization method inspired by the living behaviors of microalgae proposed by Uymaz et al. [42] in 2015. Algae's basic behavioral patterns are the movement toward the light source for photosynthesis, adaptation to the environment, and multiply mitosis. In the basic AAA, artificial algae emulate these behaviors in helical movement, adaptation, and evolutionary processes. In the solution space, a possible solution is represented by the corresponding algae colony. The population is composed of many algal colonies. A population of D-dimensional N algal colony is given in Eq. 1, and the initial population of AAA is calculated by using Eq. 2.

$$\text{Population} = \begin{bmatrix} x_1^1 & \cdots & x_1^D \\ \vdots & \ddots & \vdots \\ x_N^1 & \cdots & x_N^D \end{bmatrix} \tag{1}$$

$$\text{Population}_i^j = x_{\min}^j + r_i^j\left(x_{\max}^j - x_{\min}^j\right)\ i = 1,2,\ldots,N \text{ and } j = 1,2,\ldots,D \tag{2}$$

where x_i^j is the algal cell in the jth dimension of the ith algal colony. $x_{\max}^j$ and $x_{\min}^j$ denotes the upper and lower bounds of the jth dimension, respectively.

The algal colony grows under suitable conditions in the evolutionary process. The colonies which are not found in proper conditions cannot survive. The growth of an algal colony is calculated by the Monod model given in Eq. 3.

$$\mu = \frac{\mu_{\max} S}{K_s + S} \tag{3}$$

where $\mu_{\max}$ is the maximum specific growth rate and assumed as 1, μ is the specific growth rate, S is the nutrient concentration, and its value is taken as the fitness value. K is computed as being the growth rate at half nutrient conditions of the algal colony in time t $(G_i^t/2)$. The size of the algal colony in time $t+1$ is calculated using Eq. 4.

$$G_i^{t+1} = G_i^t + \mu_i^t G_i^t \tag{4}$$

where G_i^t is the size of the ith algal colony at time step t. Algal colonies are sorted from large to small sizes according to their size in the last evolutionary process stage. The evolutionary process is completed by copying a randomly selected algal cell of the largest algal colony to the smallest algal colony. This phase is performed using Eqs. 5–7, respectively.

$$\text{biggest}^t = \max G_i^t,\ \ i = 1,2,\ldots,N \tag{5}$$

$$\text{smallest}^t = \min G_i^t,\ \ i = 1,2,\ldots,N \tag{6}$$

$$\text{smallest}_m^t = \text{biggest}_m^t\ \ m = 1,2,\ldots,D \tag{7}$$

where m is a randomly selected algal cell (dimension of the problem), biggest and smallest represents the biggest algal colony and the smallest one, respectively.

The basic idea of the adaptation process is that the algal colony that does not grow sufficiently can adapt to the circumference. The algal colony that is not adequately developed tries to resemble itself to the largest algal colony. The adaptation process is applied to the starving (highest starvation valued) algae colony. The starvation value is taken as zero for all algal colonies at the initial stage. The starvation value is increased if algal colony movements cannot reach a better position. The adaptation parameter (A_p) determines whether the algal colony with the highest starvation will undergo adaptation or not. The adaptation parameter takes a value between 0 and 1. The algal colony with the highest starvation undergoes adaptation if the generated random number is smaller than the adaptation parameter. The adaptation process is realized using Eqs. 8, 9.

$$\text{starving}^t = \max A_i^t,\ i = 1, 2, \ldots, N \tag{8}$$

$$\text{starving}^{t+1} = \text{starving}^t + (\text{biggest}^t - \text{starving}^t) \times \text{rand} \tag{9}$$

where starving^t is the algal colony with the highest starvation value in time t and A_i^t is the starvation value of ith algal colony in time t. biggest^t is the biggest colony in population in time t. rand is a random number in the range (0, 1).

Algal colony moves helically in the living space to achieve better conditions. Each algal colony moves depending on the energy it has. The algal colony loses some energy due to the loss of energy (e) parameter in each movement. If the algal colony cannot move in a better position, it is again subject to energy loss due to its metabolism. The motion frequencies of the algal colonies, which are more energetic, are more significant. This process increases the local search capability of the method. Besides, friction is another factor that affects movement. The smaller algal colonies have a lower friction surface so that the motion distances are more considerable. This process also increases the global search capability of the method. The friction surface of an algal colony is calculated using Eq. 10.

$$\tau(x_i) = 2\pi\left(\sqrt[3]{\frac{3G_i}{4\pi}}\right) \tag{10}$$

where $\tau(x_i)$ is the friction surface.

In AAA, each algal colony performs a 3-dimensional helical movement. The helical movement is realized using Eqs. 11–13.

$$x_{\text{im}}^{t+1} = x_{\text{im}}^t + \left(x_{\text{jm}}^t - x_{\text{im}}^t\right)(\Delta - \tau^t(x_i)) \times p \tag{11}$$

$$x_{\text{ik}}^{t+1} = x_{\text{ik}}^t + \left(x_{\text{jk}}^t - x_{\text{ik}}^t\right)(\Delta - \tau^t(x_i)) \times \cos\alpha \tag{12}$$

$$x_{\text{il}}^{t+1} = x_{\text{il}}^t + \left(x_{\text{jl}}^t - x_{\text{il}}^t\right)(\Delta - \tau^t(x_i)) \times \sin\beta \tag{13}$$

where α and β are randomly generated angle values in the range (0, 2π), p is randomly generated value in the range (− 1, 1), Δ is shear force parameter, $\tau^t(x_i)$ is the friction surface area of the ith algal cell at time step t, m, k and l are randomly determined dimension indexes different from each other, and j represents the index of randomly selected neighbor different from i. x_j is randomly selected by the tournament method and candidate solution x_i^{t+1} makes a helical movement toward x_j.

In the AAA, initial algal colonies are randomly generated, and adaptation parameter (A_p), shear force (Δ), and loss of energy (e) are defined. Each algal colony performs helical movement until its energy is exhausted. Then, the size of algal colonies is calculated, and the evolution process is carried out. Finally, the adaptation process is realized using the adaptation parameter, and a new population is obtained. These steps are repeated depending on the specified stopping criterion. The best solution is stored by comparing the solutions obtained in each step with the best solution.

2.2 Galactic swarm optimization

The GSO algorithm was introduced by Muthiah-Nakarajan and Noel in 2016 [45]. The GSO algorithm searches for the optimum solution by simulating the movements of the stars, galaxies, and super galaxy clusters in space. GSO is a two-phase optimization method. In the first phase, independent sub-population groups try to improve their own best solutions using the determined search method. Each independent sub-population is run by the number of iterations determined using the search algorithm. After that, a new population called a super-population is formed by obtaining the best individual of each sub-population. In the second phase, an optimal solution is searched by using the determined search method. The first and second phases are repeated with the number of epoch parameters ($EP_{\max}$) and try to find the best solution. The PSO algorithm is used as a search method in both the first and second phases of the base GSO algorithm.

PSO is an optimization method inspired by the social behavior of bird flocks and fish schools. The best results are obtained by moving the randomly generated initial population in the search space [38, 48]. For a d-dimensional optimization problem $X_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$ is the position vector, $V_i = [v_{i1}, v_{i2}, \ldots, v_{id}]$ is the velocity vector, $P_i = [p_{i1}, p_{i2}, \ldots, p_{id}]$ is the best position vector of the ith particle and is called pbest. The motion of the particles in the solution space is done according to Eqs. 14, 15.

$$v_i(t+1) = wv_i(t) + c_1r_1(p_i - x_i) + c_2r_2(g_i - x_i) \tag{14}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{15}$$

where w called inertia weight and used to control global and local searchability. r_1 and r_2 are the random numbers in the range of (0,1). c_1 and c_2 are the acceleration coefficients. g_i is the best solution found so far.

The two-phase structure of the GSO algorithm is expressed in Eq. 16:

$$x1_j^i \in X1_i : j = 1, 2, \ldots, N,\ i = 1, 2, \ldots, M$$

$$g1_i \in X1_i : g1_i = \text{best}(X1_i)$$

$$X2 = \bigcup_{i=1}^{M} g1_i \tag{16}$$

In the original GSO algorithm, the initial M sub-populations ($X1$) are randomly generated. Each sub-population contains N solutions. $x1_j^i$ expresses the jth solution of the ith sub-population. $X1_i$ states the ith sub-population. $x2_i$ ($\text{best}(X1_i)$) represents the best solution of the sub-population $X1_i$. $X2$ states the super-population that involves the best solutions of the sub-populations. The pseudo-code of the original GSO is given in Fig. 1.

2.3 Uncapacitated facility location problem

UFLP is a frequently encountered location problem in the field of operational research. The main objective of UFLP is to find a subset of potential active facilities, ensuring the cost of this subset at minimal. There are two different costs in this problem; one of them is the facility establishment cost which is a fixed cost, and the other is the transshipment cost between the customer and the facility. While the facility establishment fixed costs and transshipment costs are known in the problem, the number of facilities that should be opened is not known where the primary aim is obtaining minimum total cost. Additionally, all demands of the customers should be corresponded by the facilities with the lowest transshipment cost. Let J, I, and C be the set of customers, the set of potential active facilities, and the transshipment cost matrix, respectively. The mathematical objective function of UFLP is given as follows:

Fig. 1 The pseudo-code of the base GSO

```
Objective function f(x), x = (x_1, x_2, ..., x_d,)
Initialize phase 1 populations (X1) in [x_min, x_max]^D
Initialize phase 1 variables (v1, p1, g1) in [x_min, x_max]^D
Initialize phase 2 population (X2) in [x_min, x_max]^D
Initialize phase 2 variables (v2, p2, g2) in [x_min, x_max]^D
Initialize parameter EPmax, M, N
Initial Phase
For i = 1 to M
  X1_i = Generate randomly initial population for ith sub-population
end For
For EP = 1 to EPmax
  Phase 1
  For I = 1 to M
    For k = 1 to the number of iteration of Phase 1
      For j = 1 to N
        Update the velocity of jth solution of the ith sub-population (v1_j^i) by using Eq.14
        Update the position of jth solution of the ith sub-population (x1_j^i) by using Eq.15
        Calculate the fitness value of the jth solution of the ith sub-population (f(x1_j^i))
        Update the personal best of jth solution of the ith sub-population (p1_j^i)
      end For
      Update the global best of the ith sub-population (g1_i)
    end For
  end For
  Phase 2
  X2 = Combine the global best of each sub-population for the initial population of Phase 2
  For k = 1 to the number of iteration of Phase 2
    For i = 1 to M
      Update the velocity of ith solution of the super-population (v2_i) by using Eq.14
      Update the position of ith solution of the super-population (x2_i) by using Eq.15
      Calculate the fitness value of the ith solution of the super-population (f(x2_i))
      Update the personal best of ith solution of the super-population (p2_i)
    end For
    Update the global best of the super-population (g2)
  end For
end For
Return the global best of the super-population (g2)
```

$$F(S) = \sum_{i \in S} f_i + \sum_{j \in J} \min\{C_{i,j} | i \in S\} \tag{17}$$

where f_i is the opening cost of the ith facility and S is a nonempty subset of I. The main goal of UFLP is to obtain the S set that meets the minimum cost requirement.

The proposed method in this study was evaluated on Cap and M^* problems, which are frequently used problems among the UFL problems in the literature. The names, size, and optimum cost of the 15 problems within the scope of the Cap problems are given in Table 1. These problems are divided into 4 groups in terms of size as small (Cap71, Cap72, Cap73, and Cap74), medium (Cap101, Cap102, Cap103, and Cap104), large (Cap131, Cap132, Cap133, and Cap134), and huge (CapA, CapB, and CapC,). Small-size problems (16 × 50) include 16 facilities and 50 customers. Medium-size problems (25 × 50) include 25 facilities and 50 customers. Large-size problems (50 × 50) include 50 facilities and 50 customers. Huge-size problems (100 × 1000) also include 100 facilities and 1000 customers. Small-size and medium-size problems are easy problems to solve. Large-size problems are relatively more difficult to solve. Huge-size problems are the most difficult problems to solve within the scope of Cap problems due to a large number of possible solutions.

The other set of UFL problems used in this study are M^* problems. There are 20 different problems in this set, and the properties of the problems are presented in Table 2. M^* problems are divided into 3 groups according to their sizes [low-scaled (MO1–MO5), middle-scaled (MP1–MP5 and MQ1–MQ5) and large-scaled (MR1–MR5)] [20].

Table 1 The properties of Cap problems

Problem name	Problem size	Optimal cost
Cap71	16 × 50	932615.75
Cap72	16 × 50	977799.40
Cap73	16 × 50	1010641.45
Cap74	16 × 50	1034976.98
Cap101	25 × 50	796648.44
Cap102	25 × 50	854704.20
Cap103	25 × 50	893782.11
Cap104	25 × 50	928941.75
Cap131	50 × 50	793439.56
Cap132	50 × 50	851495.33
Cap133	50 × 50	893076.71
Cap134	50 × 50	928941.75
CapA	100 × 1000	17156454.48
CapB	100 × 1000	12979071.58
CapC	100 × 1000	11505594.33

Table 2 The properties of M^* problems

Problem name	Problem size	Optimal cost
MO1	100 × 100	1305.95
MO2	100 × 100	1432.36
MO3	100 × 100	1516.77
MO4	100 × 100	1442.24
MO5	100 × 100	1408.77
MP1	200 × 200	2686.48
MP2	200 × 200	2904.86
MP3	200 × 200	2623.71
MP4	200 × 200	2938.75
MP5	200 × 200	2932.33
MQ1	300 × 300	4091.01
MQ2	300 × 300	4028.33
MQ3	300 × 300	4275.43
MQ4	300 × 300	4235.15
MQ5	300 × 300	4080.74
MR1	500 × 500	2608.15
MR2	500 × 500	2654.73
MR3	500 × 500	2788.25
MR4	500 × 500	2756.04
MR5	500 × 500	2505.05

3 Binary galactic swarm optimization (BinGSO)

Korkmaz and Kiran presented a binary version of the AAA algorithm (binAAA) for solving binary optimization problems in 2018 [32]. In the presented binAAA method, two update mechanisms are used to perform an effective search in the solution space. These mechanisms are called XOR and Stigmergic mechanisms. In the presented method, the selection of a mechanism among two to update the position of the candidate solution is determined by Eq. 18.

$$\text{Update mechanism} = \begin{cases} \text{UM}-1, & f(r<\text{UMSP}) \text{ and } C_{01}(t) \neq 0 \text{ and } C_{10}(t) \neq 0 \\ \text{UM}-2, & \text{otherwise} \end{cases} \tag{18}$$

where UM–1 (XOR) and UM–2 (stigmergic) are stand for the update mechanism 1 and the update mechanism 2, respectively. *UMSP* is the method-specific parameter called update mechanism selection probability. r is a random number composed in the range of (0,1). C_{10} and C_{01} parameters state the count of changed values 1–0 and 0–1, respectively.

In the original AAA method, the candidate solution makes a helical movement in the solution space. This means a 3-dimensional change of the position of the candidate solution. The XOR update mechanism also updates the position of the candidate solution by making 3-dimensional updates like helical movement. The XOR update mechanism performs the position upte process according to Eqs. 19–21.

Let $V = X_i$

$$V_j = X_{i,j} \oplus \left[\varphi\left(X_{i,j} \oplus X_{n,j}\right)\right] \tag{19}$$

$$V_k = X_{i,k} \oplus \left[\varphi\left(X_{i,k} \oplus X_{n,k}\right)\right] \tag{20}$$

$$V_l = X_{i,l} \oplus \left[\varphi\left(X_{i,l} \oplus X_{n,l}\right)\right] \tag{21}$$

$$i, n \in \{1, 2, \ldots, N\},\ i \neq n,\ j, k, l \in \{1, 2, \ldots, D\},\ j \neq k \,/\!= l$$

where V is the candidate solution, X is the solution in the population, $\oplus$ defines the logical XOR operator, φ defines the logic NOT operator with 50% probability, N is the number of solutions in the population and D is the dimension of the problem. V_j, V_k, and V_l state the jth, the kth, and the lth dimensions of the candidate solution, respectively. n is the index of the neighbor randomly selected from the population using the tournament selection method, different from the i index.

C_{10} and C_{01} values used in the Stigmergic update mechanism are updated after generating the candidate solution using the XOR operator. These parameters are updated by using Eqs. 22, 23.

$$C_{01}(t+1) = \begin{cases} C_{01}(t) + 1, & \text{Obj}(V) < \text{Obj}(X_i) \text{ and } X_{i,d} = 0 \text{ and } V_d = 1, \\ C_{01}(t), & \text{otherwise} \end{cases} \forall d \in P \tag{22}$$

$$C_{10}(t+1) = \begin{cases} C_{10}(t) + 1, & \text{Obj}(V) < \text{Obj}(X_i) \text{ and } X_{i,d} = 0 \text{ and } V_d = 1, \\ C_{10}(t), & \text{otherwise} \end{cases} \forall d \in P \tag{23}$$

$$P = \{j, k, l\}$$

where V is the candidate solution, X is the solution in the population and *Obj* is the objective function of the problem. The adaptation process for binary problems is presented in Eq. 24. Equations 22 and 23 are applied separately for each of the 3 randomly selected dimensions in the P set.

$$X_{s,z} = \begin{cases} X_{b,z}, & if\,(r_z < Ap) \\ X_{s,z}, & \text{otherwise} \end{cases} \quad z \in D,\ z = 1, 2, \ldots, D \tag{24}$$

where $X_{s,z}$ states *the* cell of the most starveling algal colony s, $X_{b,z}$ states the zth cell of the biggest algal colony b, r_z states a random number produced for dimension z and Ap is the adaptation parameter. The pseudo-code of the XOR update mechanism is given in Fig. 2.

The Stigmergic update mechanism uses C_{10} and C_{01} parameters that are updated thanks to the XOR update mechanism. p_{01} and p_{10} values are calculated using C_{10} and C_{01} parameters (Eqs. 25, 26).

$$p_{01}(t+1) = \frac{C_{01}(t)}{C_{01}(t) + C_{10}(t)} \tag{25}$$

$$p_{10}(t+1) = \frac{C_{10}(t)}{C_{01}(t) + C_{10}(t)} \tag{26}$$

where $p_{01}(t+1)$ is the probability rate of C_{01} in time $t + 1$ and $p_{10}(t+1)$ is the probability rate of C_{10} in time $t + 1$.

Let A and B be vectors of the index of ones and zeros in V, respectively. Let a and b be random integers in the range (1, size of A) and (1, size of B), respectively. $V_{A(a)}$ and $V_{B(b)}$ define random dimensions that have a value of 1 and 0, respectively. The candidate solution V is calculated by using Eqs. 27, 28.

$$V_{A(a)} = \begin{cases} 0, & r_1 < p_{10} \\ V_{A(a)}, & \text{otherwise} \end{cases} \tag{27}$$

$$V_{B(b)} = \begin{cases} 1, & r_1 \geq p_{10} \\ V_{B(b)}, & \text{otherwise} \end{cases} \tag{28}$$

where r_1 is a random number in the range of (0,1). If the value of r_1 is less than the probability value of p_{10}, random decision variable with a value of 1 $\left(V_{A(a)}\right)$ is set to 0. Otherwise, the decision variable with a value of 0 $\left(V_{B(b)}\right)$ is set to 1. The working structure of the stigmergic update mechanism is given as pseudo-code in Fig. 3.

After the update mechanism is applied to the candidate solution, the fitness value of the new candidate solution obtained is calculated and compared with the fitness value of the existing solution. If the fitness value of the new candidate solution is better, the position of the existing solution is updated. Otherwise, the position update is not performed. This process is expressed in Eq. 29.

$$X_i(t+1) = \begin{cases} V(t), & \text{Obj}(V(t)) < \text{Obj}(X_i(t)) \\ X_i(t), & \text{otherwise} \end{cases} \tag{29}$$

There are several studies in the literature in which the GSO framework improves the problem-solving performance of traditional metaheuristic methods [45–47]. The main reason for this is that the framework effectively manages the exploration and exploitation balance. In this study, a binary GSO method using binAAA method as the search algorithm due to its effectiveness in solving binary problems is proposed. The flowchart of the suggested BinGSO method is given in Fig. 4.

Fig. 2 The pseudo-code of the XOR update mechanism

```
Let n is index of randomly selected neighbor for X(i) using Obj via tournament selection
Let P be three random dimension indexes between [1,D] and different from each other
Let V is a candidate solution
φ is the logic NOT operator set as 0.5
V = X(i)
For z = 1 to 3
    If rand < φ Then
        V( P(z) ) = XOR ( X(i, P(z) ) , XOR ( X(i, P(z) ), X(n, P(z) ) ) )
    Else
        V( P(z) ) = XOR ( X(i, P(z) ) , ~XOR ( X(i, P(z) ), X(n, P(z) ) ) )
    end If
end For
If Obj(V) < Obj( X(i) ) Then
    For z = 1 to 3
        If X(i, P(z) ) == 0 and V( P(z) ) == 1 Then C_01 = C_01+1 end If
        If X(i, P(z) ) == 1 and V( P(z) ) == 0 Then C_10 = C_10+1 end If
    end For
    X(i) = V
end If
```

```
Let V is candidate solution
For d = 1 to 3
    If rand < DSP Then
        If rand < p_10 Then
            A = find( V==1 )
            a = rand( size of A )
            V( A(a) ) = 0
        else If
            B = find( V==0 )
            b = rand( size of B )
            V( B(b) ) = 1
        end If
    end If
end For
If Obj(V) < Obj( X(i) ) Then
    X(i) = V
end If
```

Fig. 3 The pseudo-code of the Update Mechanism 2

4 Experimental results and discussion

The experimental study consists of three stages. In the first stage, the BinGSO method was used in solving Cap problems taken from OR-Lib [49] with different parameters ($EP_{\max}$, M, N). The parameter analysis was made, and the results were analyzed at this stage. In the second stage, results of the BinGSO method were compared with results of the state-of-the-art studies in the literature using Cap problems. In the third stage, M^* problems, which are different UFL Problems taken from OR-Lib [49], were solved by the BinGSO method, and the results obtained were compared with the results of the state-of-the-art methods. The best results obtained in the comparison tables (Table 3, 5, 6, 7, 8, and 10) are highlighted in bold style.

The gap value was used to evaluate the performance of the methods in the evaluation process. The gap value is calculated as in Eq. 30 depending on the optimum value of the problem and the mean value obtained by the method.

$$\text{gap} = \frac{\text{mean} - \text{optimum}}{\text{optimum}} \times 100 \tag{30}$$

The experimental studies were accomplished with 30 runs of different seeds and a total of 80,000 fitness calculation parameters. The energy loss, the adaptation rate, the update mechanism selection probability (UMSP), and the dimension selection probability (DSP) parameters of the binAAA method were directly taken from [32] as 0.3, 0.5, 0.5 and 0.66, respectively.

4.1 The parameter analysis of BinGSO

GSO parameters ($EP_{\max}$, M, N) were analyzed in 15 UFL Problems (Cap problems) taken from OR-Lib in the parameter analysis process. The experimental study was carried out for a total of 12 different parameter sets with 3, 5 and 8 values for the $EP_{\max}$ parameter and 5 and 10 values for M and N parameters. The gap and rank values obtained from the evaluation process performed for parameter analysis are presented in Table 3. The count of best results and mean rank values obtained by utilizing the methods are given in the last row of the table.

The parameter analysis stage aims to determine the most suitable $EP_{\max}$, M and N parameter values for the BinGSO method in Cap problems. When Table 3 is examined, 3 parameter sets ({$EP_{\max} = 3$, $M = 10$, $N = 5$}, {$EP_{\max} = 5$, $M = 10$, $N = 10$}, {$EP_{\max} = 8$, $M = 10$, $N = 10$})

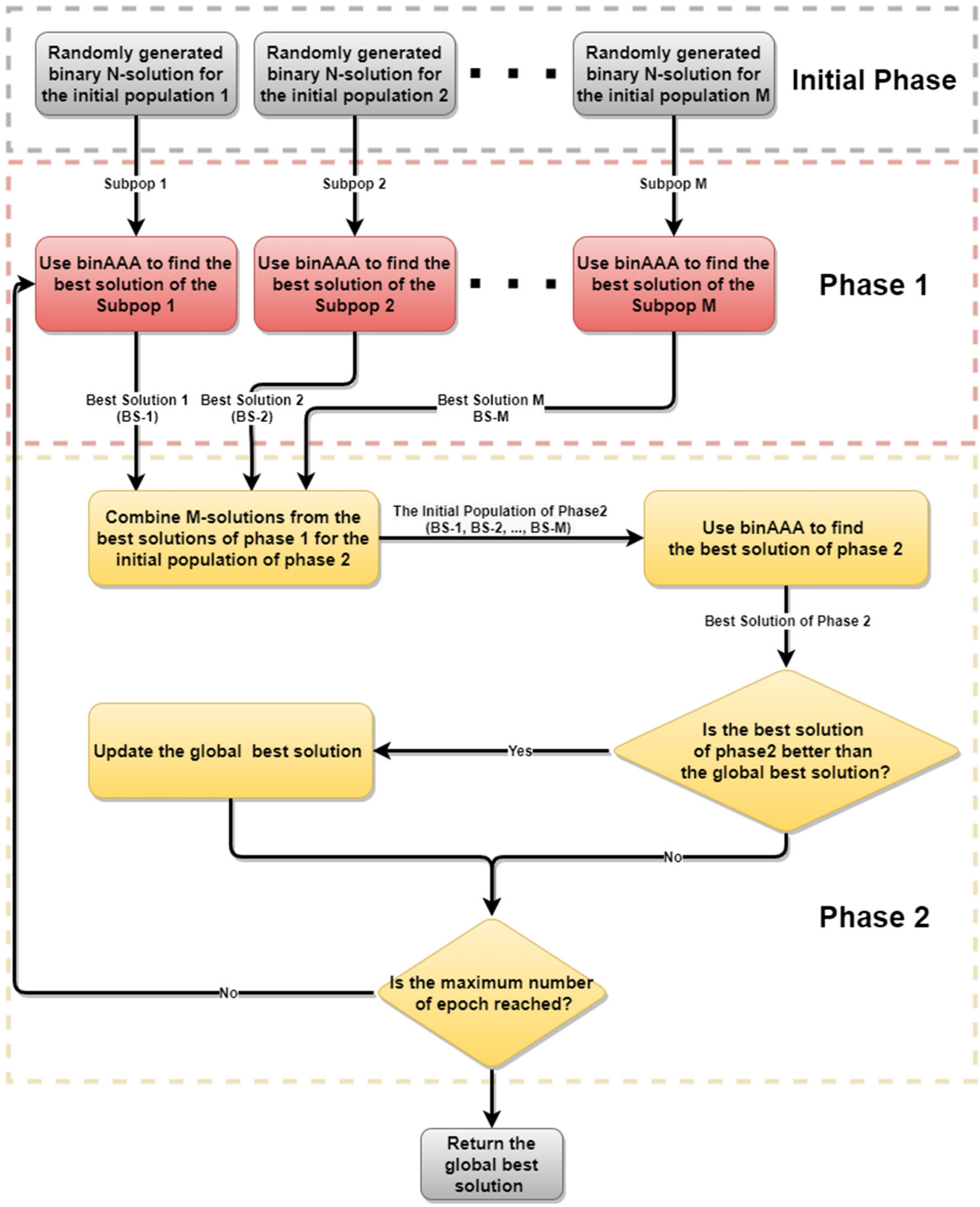

Fig. 4 The flowchart of BinGSO

achieved the best score on 13 problems in terms of the count of best results obtained. These three sets of parameters have reached the optimum solution for small-size, medium-size, large-size, and CapA problems. Therefore, the performances in CapB and CapC problems are decisive in determining which parameter is more appropriate. When the CapB and CapC performances of these 3 parameter sets are examined, it is clearly seen that by using the parameter set $\{EP_{max} = 3, M = 10, N = 5\}$, the method achieved the best results. Although, by using $\{EP_{max} = 3, M = 10, N = 5\}$ parameter set, the method does not get the best result in CapB and CapC problems, it turns out to be the most successful parameter set when the mean rank value is taken into account. As a result, the optimal parameter values for Cap problems in BinGSO method are 3, 10, and 5 for EP_{max}, M, and N, respectively. The detailed results of

Table 3 The experimental results of the analysis of the parameters on BinGSO

	EP_{max} = 3				EP_{max} = 5				EP_{max} = 8			
	$M = 5$ $N = 5$	$M = 5$ $N = 10$	$M = 10$ $N = 10$	$M = 10$ $N = 10$	$M = 5$ $N = 5$	$M = 5$ $N = 10$	$M = 10$ $N = 5$	$M = 10$ $N = 10$	$M = 5$ $N = 5$	$M = 5$ $N = 10$	$M = 10$ $N = 5$	$M = 10$ $N = 10$
Cap71												
Gap	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Cap72												
Gap	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Cap73												
Gap	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Cap74												
Gap	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Cap101												
Gap	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Cap102												
Gap	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Cap103												
Gap	0.0008	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	**2**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Cap104												
Gap	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Cap131												
Gap	0.0108	0.0108	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.0072	**0.0000**	0.0072	**0.0000**	**0.0000**	**0.0000**
Rank	3	3	**1**	**1**	**1**	**1**	2	**1**	2	**1**	**1**	**1**
Cap132												
Gap	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Cap133												
Gap	0.0053	0.0026	**0.0000**	**0.0000**	0.0053	0.0026	0.0007	**0.0000**	0.0132	0.0026	0.0079	**0.0000**
Rank	4	3	**1**	**1**	4	3	2	**1**	6	3	5	**1**
Cap134												
Gap	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
CapA												
Gap	**0.0000**	**0.0000**	**0.0000**	0.0187	**0.0000**	**0.0000**	0.0499	**0.0000**	0.0047	**0.0000**	0.0140	**0.0000**
Rank	**1**	**1**	**1**	3	**1**	**1**	4	**1**	2	**1**	2	**1**
CapB												
Gap	**0.1980**	0.2664	0.2384	0.3095	0.2518	0.3785	0.3390	0.3902	0.4499	0.2823	0.4443	0.3902
Rank	**1**	4	2	6	3	8	7	9	11	5	10	9
CapC												
Gap	0.1845	**0.1600**	0.2095	0.3162	0.2799	0.1928	0.4138	0.3147	0.2072	0.2016	0.4491	0.3147
Rank	2	**1**	6	9	7	3	10	8	5	4	11	8
Winner/total	11/15	12/15	**13**/15	12/15	12/15	12/15	10/15	**13**/15	10/15	12/15	11/15	**13**/15

Table 3 (continued)

	EP_{max} = 3				EP_{max} = 5				EP_{max} = 8			
	M = 5 N = 5	M = 5 N = 10	M = 10 N = 10	M = 10 N = 10	M = 5 N = 5	M = 5 N = 10	M = 10 N = 5	M = 10 N = 10	M = 5 N = 5	M = 5 N = 10	M = 10 N = 5	M = 10 N = 10
Mean rank	1.4667	1.4667	**1.4000**	2.0000	1.7333	1.7333	2.3333	2.0000	2.4000	1.6000	2.6000	2.0000

Table 4 The detailed results of the BinGSO with {EP_{max} = 3, M = 10, N = 5} parameters set on Cap problems

	Best	Worst	Mean	Gap	Std. Dev.
Cap71	932615.750	932615.750	932615.750	0.0000	0.000
Cap72	977799.400	977799.400	977799.400	0.0000	0.000
Cap73	1010641.450	1010641.450	1010641.450	0.0000	0.000
Cap74	1034976.975	1034976.975	1034976.975	0.0000	0.000
Cap101	796648.438	796648.438	796648.438	0.0000	0.000
Cap102	854704.200	854704.200	854704.200	0.0000	0.000
Cap103	893782.113	893782.113	893782.113	0.0000	0.000
Cap104	928941.750	928941.750	928941.750	0.0000	0.000
Cap131	793439.563	793439.563	793439.563	0.0000	0.000
Cap132	851495.325	851495.325	851495.325	0.0000	0.000
Cap133	893076.713	893076.713	893076.713	0.0000	0.000
Cap134	928941.750	928941.750	928941.750	0.0000	0.000
CapA	17156454.478	17156454.478	17156454.478	0.0000	0.000
CapB	12979071.581	13070745.086	13010017.344	0.2384	39900.115
CapC	11505594.329	11577131.301	11529702.002	0.2095	22224.898

the BinGSO method with the optimum parameter set are given in Table 4.

4.2 Comparisons on cap problems

In this stage, the BinGSO method is compared with 7 traditional binary optimization algorithms. Three of these algorithms are GA-based [50] [GA-SP (Genetic Algorithm with Single Point crossover), GA-TP (Genetic Algorithm with Two-Point crossover), and GA-UP (Genetic Algorithm with Uniform crossover)], three are AAA-based [32, 43] [BAAA-Tanh (Binary AAA with Tangent hyperbolic logistic function), BAAA-Sig (Binary AAA with Sigmoid logistic function), binAAA (binary AAA with Stigmergic Behavior)] and one is PSO-based [28] [BPSO (Binary Particle Swarm Optimization)]. These algorithms were conducted with 30 runs and 80,000 maxFES, and the results obtained are presented in Tables 5, 6.

Mean, Gap, Hit, Standard Deviation, and statistical sign test results of 7 traditional binary optimization and BinGSO algorithms are given in these two tables. The hit value indicates how many times the algorithm reaches the optimum value within 30 independent runs. Sign value stands for the results of the Wilcoxon signed-rank test [51] with 0.05 level of p. If the value is +, it can be said that there is a statistically significant difference between the results of the relevant algorithm and the results of BinGSO. Otherwise, there is no statistically significant difference between the results.

When the results in Tables 5 and 6 are examined, it can be seen that AAA-based methods (BAAA-Tanh, BAAA-Sig, and binAAA) and BinGSO method have obtained the optimum solution for all small-size and medium-size problems. BAAA-Sig, binAAA, and BinGSO achieved optimum values in all 30 runs. GA-based methods and the BPSO method have achieved a partial success. When large-size and huge-size problems were analyzed, BAAA-Sig, binAAA, and BinGSO methods achieved optimum value in all 30 runs in Cap131, Cap132, Cap133, Cap134 problems. In CapA, CapB, and CapC problems, which are the most difficult problem groups of the UFLP set, the binAAA, and BinGSO methods showed superior success in the CapA problem. The BinGSO method obtained the optimum value in 17 and 4 of 30 runs, respectively, in CapB and CapC problems, and the proposed method obtained more

Table 5 Experimental results of BinGSO with the binary optimization algorithms on the small-size and medium-size Cap problems

Methods	Metric	Cap71	Cap72	Cap73	Cap74	Cap101	Cap102	Cap103	Cap104
GA-SP	Mean	**932615.750**	**977799.400**	1011314.476	**1034976.975**	797193.286	**854704.200**	894351.782	**928941.750**
	Gap	**0.00000**	**0.00000**	0.06659	**0.00000**	0.06839	**0.00000**	0.06374	**0.00000**
	Hit	**30**	**30**	19	**30**	11	**30**	6	**30**
	Std. Dev.	**0.000**	**0.000**	899.650	**0.000**	421.655	**0.000**	505.036	**0.000**
	Sign	−	−	+	−	+	−	+	−
GA-TP	Mean	**932615.750**	**977799.400**	1,011,130.923	**1034976.975**	797164.610	**854704.200**	894329.179	**928941.750**
	Gap	**0.00000**	**0.00000**	0.04843	**0.00000**	0.06479	**0.00000**	0.06121	**0.00000**
	Hit	**30**	**30**	22	**30**	12	**30**	10	**30**
	Std. Dev.	**0.000**	**0.000**	825.576	**0.000**	428.658	**0.000**	540.160	**0.000**
	Sign	−	−	+	−	+	−	+	−
GA-UP	Mean	**932615.750**	**977799.400**	1011069.739	**1034976.975**	797107.258	**854704.200**	894,427.382	**928941.750**
	Gap	**0.00000**	**0.00000**	0.04238	**0.00000**	0.05759	**0.00000**	0.07220	**0.00000**
	Hit	**30**	**30**	23	**30**	14	**30**	9	**30**
	Std. Dev.	**0.000**	**0.000**	789.612	**0.000**	436.524	**0.000**	522.784	**0.000**
	Sign	−	−	+	−	+	−	+	−
BAAA-Tanh	Mean	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**	796677.114	**854704.200**	**893782.113**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**	0.00360	**0.00000**	**0.00000**	**0.00000**
	Hit	**30**	**30**	**30**	**30**	29	**30**	**30**	**30**
	Std. Dev.	**0.000**	**0.000**	**0.000**	**0.000**	157.066	**0.000**	**0.000**	**0.000**
	Sign	−	−	−	−	−	−	−	−
BAAA-Sig	Mean	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**	**796648.438**	**854704.200**	**893782.113**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Hit	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
	Std. Dev.	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	−	−	−	−	−	−	−	−
BPSO	Mean	**932615.750**	**977799.400**	1010886.187	1035068.312	796992.553	854788.703	894223.572	929318.098
	Gap	**0.00000**	**0.00000**	0.02422	0.00882	0.04320	0.00989	0.04939	0.04051
	Hit	**30**	**30**	26	29	18	28	14	28
	Std. Dev.	**0.000**	**0.000**	634.625	500.272	428.658	321.588	521.237	1432.239
	Sign	−	−	−	−	+	−	+	−
binAAA	Mean	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**	**793439.563**	**851495.325**	**893076.713**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Hit	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
	Std. Dev.	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	−	−	−	−	−	−	−	−
BinGSO	Mean	**932615.750**	**977799.400**	**1010641.450**	**1034976.975**	**793439.563**	**851495.325**	**893076.713**	**928941.750**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Hit	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
	Std. Dev.	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**

successful results than traditional binary optimization methods.

As a nonparametric statistical test, The Friedman test [52] is generally used to evaluate the experimental results. In this study, Gap values of 7 traditional binary optimization methods and the BinGSO method were analyzed using the Friedman test. The level of significance for the Friedman test was set as 0.05. The statistical analysis results made based on Gap values are given in Table 7. When Table 7 is evaluated, it can be said that the obtained p-Value is lower than the level of significance (0.05). This situation indicates that the results obtained in the experimental study have a statistically significant difference. Considering the Final Ranking values, the performance of

Table 6 Experimental results of BinGSO with the binary optimization algorithms on the large-size and huge-size Cap problems

Methods	Metric	Cap131	Cap132	Cap133	Cap134	CapA	CapB	CapC
GA-SP	Mean	793980.104	**851495.325**	893891.911	**928941.750**	17164354.456	13054858.045	11586692.969
	Gap	0.06813	**0.00000**	0.09128	**0.00000**	0.04605	0.58391	0.70486
	Hit	16	**30**	10	**30**	24	9	2
	Std.Dev	720.877	**0.000**	685.076	**0.000**	22,451.206	66,658.649	51,848.248
	Sign	+	−	+	−	+	+	+
GA-TP	Mean	794012.905	**851495.325**	893740.954	**928941.750**	17205089.145	13063527.186	11577797.524
	Gap	0.07226	**0.00000**	0.07438	**0.00000**	0.28348	0.65071	0.62755
	Hit	14	**30**	12	**30**	24	11	0
	Std.Dev	690.560	**0.000**	655.920	**0.000**	139,690.216	89,122.485	46,346.052
	Sign	+	−	+	−	+	+	+
GA-UP	Mean	793865.023	851517.200	893808.891	**928941.750**	17166811.915	13107633.077	11578600.532
	Gap	0.05362	0.00257	0.08198	**0.00000**	0.06037	0.99053	0.63453
	Hit	15	29	9	**30**	24	3	0
	Std.Dev	433.467	119.817	628.654	**0.000**	35181.974	79714.021	57031.219
	Sign	+	−	+	−	+	+	+
BAAA-Tanh	Mean	793525.591	**851495.325**	893333.515	**928941.750**	17471223.794	13153617.764	11676427.752
	Gap	0.01084	**0.00000**	0.02875	**0.00000**	1.83470	1.34483	1.48479
	Hit	27	**30**	16	**30**	3	0	0
	Std.Dev	262.498	**0.000**	324.451	**0.000**	225123.921	73978.543	101438.607
	Sign	−	−	+	−	+	+	+
BAAA-Sig	Mean	**793439.563**	**851495.325**	**893076.713**	**928941.750**	17210900.533	13093705.559	11583462.068
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**	0.31735	0.88322	0.67678
	Hit	**30**	**30**	**30**	**30**	16	1	1
	Std.Dev	**0.000**	**0.000**	**0.000**	**0.000**	90743.456	62168.803	45788.678
	Sign	−	−	−	−	+	+	+
BPSO	Mean	794797.761	851991.551	893816.653	930756.565	17,446,511,870	13,161,205,473	11,692,212,797
	Gap	0.17118	0.05828	0.08285	0.19536	1,69,066	1,40,329	1,62,198
	Hit	10	21	10	18	8	5	1
	Std. Dev.	1505.749	1055.238	690.192	2594.211	319,855,431	135,326,728	115,156,444
	Sign	+	+	+	+	+	+	+
binAAA	Mean	**793439.563**	**851495.325**	**893076.713**	**928941.750**	**17156454.478**	13011234.616	11539496.443
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	0.24781	0.29466
	Hit	**30**	**30**	**30**	**30**	**30**	15	1
	Std. Dev.	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	39224.744	29766.311
	Sign	−	−	−	−	−	+	+
BinGSO	Mean	**793439.563**	**851495.325**	**893076.713**	**928941.750**	**17156454.478**	**13010017.344**	**11529702.002**
	Gap	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.23843**	**0.20953**
	Hit	**30**	**30**	**30**	**30**	**30**	**17**	**4**
	Std .Dev.	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**39900.115**	**22224.898**

the BinGSO method employed for solving Cap problems is better than the performance of other methods.

Convergence curves of the algorithms are given in Figs. 5 and 6. The graphs show that the BinGSO method converged to the optimum solution in small-size problems in the early stages. Except for Cap103, the BinGSO method also reaches the optimum solution in the early stages of medium-size problems. In large-size and huge-size problems, the convergence of the BinGSO method to the optimum solution is generally better than other methods.

The results of the state-of-the-art algorithms studied for solving Cap problems in the literature are compared with the results of the BinGSO algorithm as well as the implemented 7 traditional binary optimization algorithm. These

Table 7 The overall results of BinGSO with the binary optimization algorithms on the Cap problems

	GA-SP	GA-TP	GA-UP	BAAA-Tanh	BAAA-Sig	BPSO	binAAA	BinGSO
Average	0.1129	0.1255	0.1331	0.3138	0.1252	0.3600	0.0362	**0.0299**
Winner/total	7/15	7/15	6/15	9/15	12/15	2/15	13/15	**15/15**
Friedman's test								
Mean rank	5.1667	4.8333	5.1333	4.5000	3.7000	6.8000	3.0000	**2.8667**
Final rank	7	5	6	4	3	8	2	**1**
p-Value	**7.05E − 08**							

algorithms are Improved Binary Particle Swarm Optimization (IBPSO) [53], Dissimilarity Artificial Bee Colony algorithm (DisABC) [54], XOR-based artificial bee colony algorithm for binary optimization (binABC) [19], the continuous artificial bee colony algorithm for binary optimization (ABCbin) [55], differential evolution algorithm for binary optimization (DisDE) [56], Binary Differential Evolution strategies. (binDE) [57], Similarity and Logic Gate-based Tree-Seed Algorithm (SimlogicTSA) [30], Improved Scatter Search algorithm (ISS) [21], and Binary Social Spider Algorithm (BinSSA) [20]. The gap and rank values obtained by the state-of-the-art methods and BinGSO method in 15 Cap problems are given in Table 8. The gap values which are given in the table are taken directly from the related studies. In the last two rows of the table, the results are summarized by giving the best gap number (Winner/Total) and the average rank (Mean Rank) obtained by the methods.

According to Table 8, the IBPSO method could not achieve the best gap value in any Cap Problems. Methods other than IBPSO have reached the optimum solution in small-size problems, which are the easiest group to solve. SimlogicTSA, ISS, BinSSA, and BinGSO methods achieved optimum value in medium-size and large-size problems. When the results of the huge-size problem group in Table 8 are examined, SimlogicTSA, it can be seen that ISS, BinSSA, and BinGSO methods have obtained the optimum solutions for the CapA problem. In the CapB and CapC problems, none of the methods could obtain the average optimum value. Although the DisDE method is more successful in CapB and CapC problems than other methods, it has not been successful in problems that are easier to solve than huge-size problems, such as medium-size and large-size problems. This means that the DisDE method is not stable and robust when using problems of different sizes and types. SimlogicTSA, ISS, BinSSA, and BinGSO methods are more successful than other methods by obtaining the best solution in 13 problems. BinGSO method obtained the best mean rank value when evaluated in terms of mean rank.

Considering the overall performance of the BinGSO method used for solving Cap problems is examined, it is clear that the proposed method presents more successful results than the results of the other methods in terms of the number of best solutions and the average rank value obtained with both traditional binary optimization methods and state-of-the-art methods.

4.3 Comparisons on M* problems

Although not as widely used as UFLP in the literature, *M** problems are used as benchmark problems in comparing algorithms in the area of uncapacitated facility location. *M** problems consist of a total of 20 problems in 3 groups (low-scaled, middle-scaled, and large-scaled) according to their sizes. Details of the problems are given in Table 2 in Sect. 2.3, and the performance of the BinGSO method on *M** problems is presented in Table 9. The table shows the best, worst, mean, gap, and standard deviation values obtained by the BinGSO method.

The results obtained by the BinGSO method on *M** problems have been compared with the state-of-the-art methods that have worked on these problems in the literature. These algorithms are the Binary Social Spider Algorithm (BinSSA) [20], Local Search (LS) [58], and Improved Scatter Search (ISS) [21]. For a fair comparison, the BinGSO method was carried out 100 runs on *M** problems as the other compared methods do. The mean and gap values obtained by the BinSSA, LS, and ISS methods in *M** problems were taken directly from [20, 58], and [21], respectively, and the results of these methods are presented with the results of the BinGSO method.

When Table 10 is analyzed, it is clearly seen that the BinGSO method provides the best solution for low-scaled and middle-scaled problems. LS method is superior in large-scale problems. In summary, the BinGSO method achieved the best solution in 13 out of 20 problems and was more successful than other methods. Considering rank values, BinGSO method obtained the best mean rank value.

Table 8 Experimental results of BinGSO with the state-of-the-art optimization algorithms on the Cap problems

	IBPSO	DisABC	binABC	ABCbin	DisDE	binDE	SimlogicTSA	ISS	BinSSA	BinGSO
Cap71										
Gap	0.0370	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	2	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Cap72										
Gap	0.2750	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	2	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Cap73										
Gap	0.1980	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	2	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Cap74										
Gap	0.4030	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	2	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Cap101										
Gap	0.5970	**0.0000**	**0.0000**	**0.0000**	0.0036	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	3	**1**	**1**	**1**	2	**1**	**1**	**1**	**1**	**1**
Cap102										
Gap	0.7320	**0.0000**	**0.0000**	**0.0000**	0.0049	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	3	**1**	**1**	**1**	2	**1**	**1**	**1**	**1**	**1**
Cap103										
Gap	0.6410	**0.0000**	**0.0000**	0.0050	0.0055	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	4	**1**	**1**	2	3	**1**	**1**	**1**	**1**	**1**
Cap104										
Gap	0.9960	**0.0000**	**0.0000**	0.0000	0.0000	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	2	**1**	**1**	1	1	**1**	**1**	**1**	**1**	**1**
Cap131										
Gap	2.4240	0.6200	**0.0000**	0.1970	0.0036	0.0036	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	5	4	**1**	3	2	2	**1**	**1**	**1**	**1**
Cap132										
Gap	3.6010	0.0950	**0.0000**	0.0200	0.0000	0.0050	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	5	4	**1**	3	1	2	**1**	**1**	**1**	**1**
Cap133										
Gap	5.2630	0.0310	0.1220	0.0750	0.0138	0.0138	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	6	3	5	4	2	2	**1**	**1**	**1**	**1**
Cap134										
Gap	7.6340	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	2	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
CapA										
Gap	137.8860	0.1520	2.5090	3.1720	0.0370	1.3000	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Rank	7	3	5	6	2	4	**1**	**1**	**1**	**1**
CapB										
Gap	55.2700	3.3030	2.5080	2.8150	**0.1890**	1.5200	0.3176	0.2550	0.2547	0.2384
Rank	10	9	7	8	**1**	6	5	4	3	2
CapC										
Gap	45.5560	4.6970	2.5800	2.0370	**0.0909**	1.5500	0.4120	0.1990	0.4337	0.2095
Rank	10	9	8	7	**1**	6	4	2	5	3
Winner/total	0/15	9/15	11/15	7/15	6/15	9/15	**13/15**	**13/15**	**13/15**	**13/15**
Mean rank	4.3333	2.7333	2.4000	2.7333	1.4667	2.0667	1.4667	1.2667	1.4000	**1.2000**

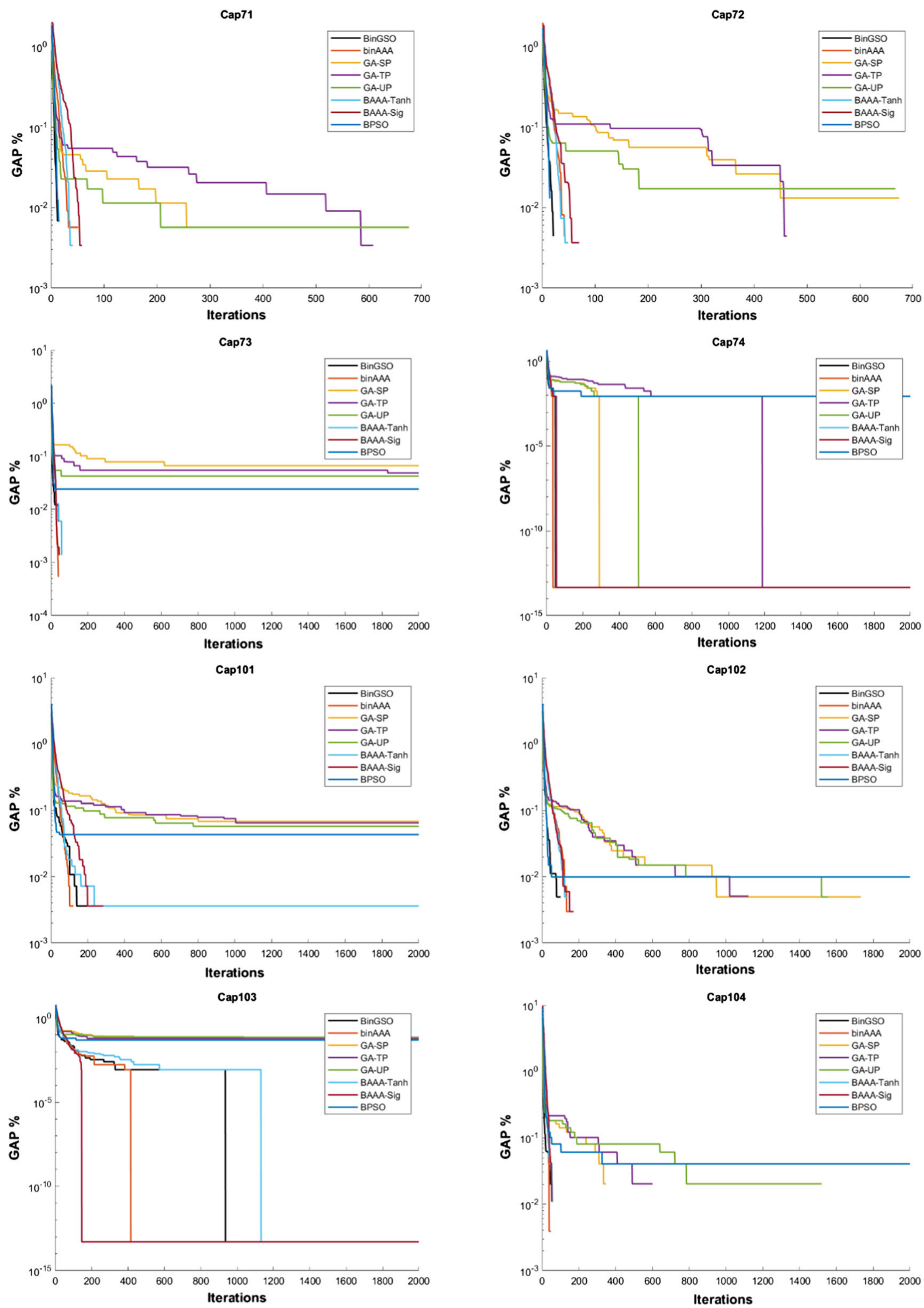

Fig. 5 Convergence curves of the methods on the small-size and medium-size Cap problems

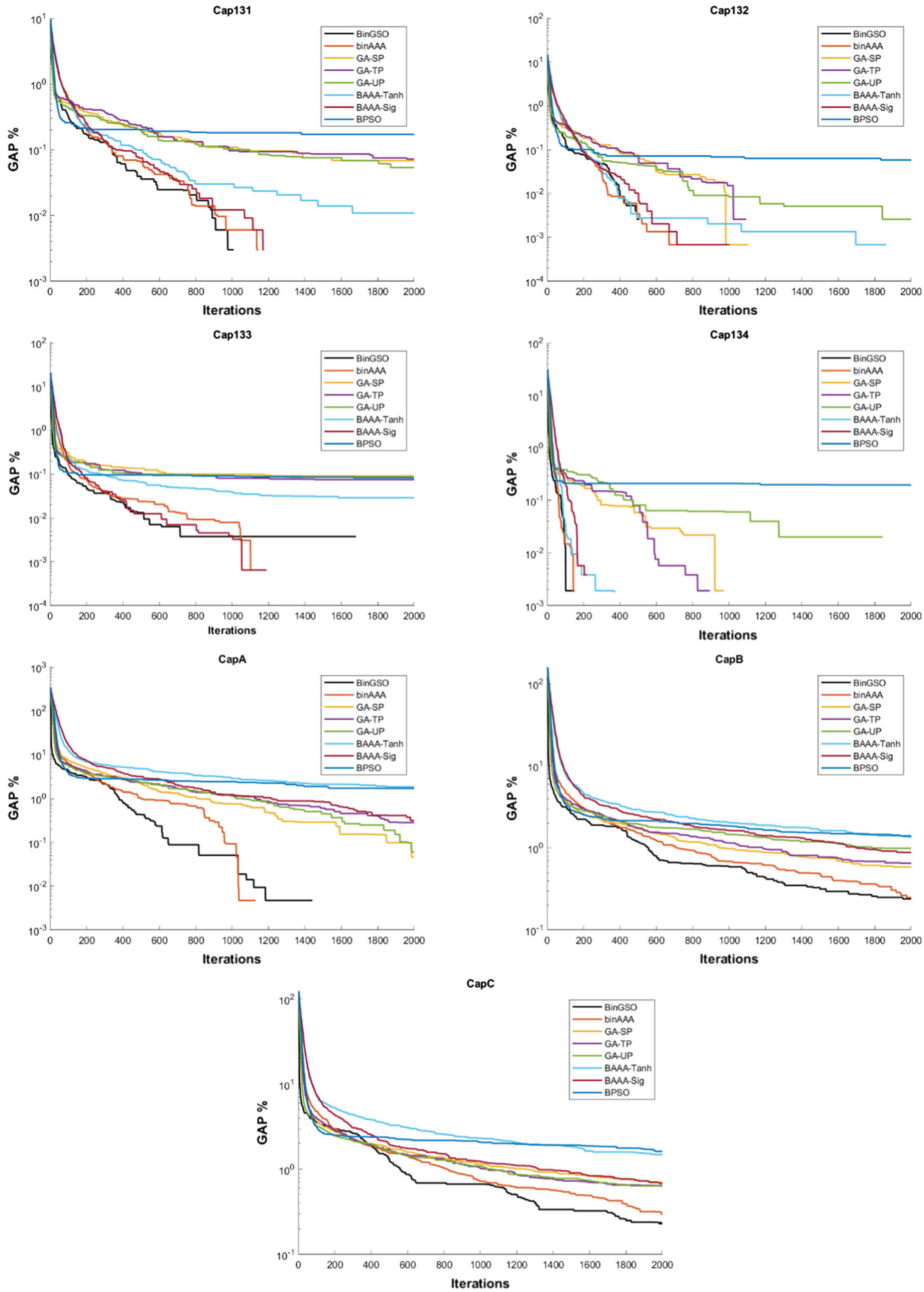

Fig. 6 Convergence curves of the methods on the large-size and huge-size Cap problems

Table 9 The detailed results of the BinGSO method on M^* problems

	Best	Worst	Mean	Gap	Std. Dev.
MO1	1305.95	1305.95	1305.95	0.0000	0.0000
MO2	1432.36	1432.36	1432.36	0.0000	0.0000
MO3	1516.77	1516.77	1516.77	0.0000	0.0000
MO4	1442.24	1442.24	1442.24	0.0000	0.0000
MO5	1408.77	1408.77	1408.77	0.0000	0.0000
MP1	2686.48	2695.72	2687.40	0.0344	2.8202
MP2	2904.86	2904.86	2904.86	0.0000	0.0000
MP3	2623.71	2623.71	2623.71	0.0000	0.0000
MP4	2938.75	2943.82	2939.44	0.0132	1.5724
MP5	2932.33	2932.33	2932.33	0.0000	0.0000
MQ1	4091.01	4091.01	4091.01	0.0000	0.0000
MQ2	4028.33	4028.33	4028.33	0.0000	0.0000
MQ3	4275.43	4275.43	4275.43	0.0000	0.0000
MQ4	4235.15	4239.23	4235.42	0.0064	1.0371
MQ5	4080.74	4104.57	4086.62	0.1439	9.0583
MR1	2608.15	2612.79	2609.30	0.0440	1.3900
MR2	2654.73	2686.83	2661.15	0.2418	13.0563
MR3	2788.25	2807.55	2793.47	0.1871	4.6992
MR4	2756.04	2783.95	2768.34	0.4463	9.6870
MR5	2505.05	2520.20	2510.25	0.2076	4.7972

5 Conclusion and future works

Galactic swarm optimization is a framework inspired by the behavior of stars and galaxies. This framework has a two-phase structure that uses traditional optimization methods as search algorithms. The first stage aims at exploration and the second stage aims at exploitation. GSO increases the performance of existing optimization methods by balancing the exploration and exploitation capability of the search algorithms. PSO was used as the search algorithm in the original GSO method, and continuous optimization problems were tried to be solved. In this effective framework, no binary optimization method has been used as a search algorithm. In this study, the binary GSO method that solves binary location problems for the first time using the Binary Artificial Alge Algorithm as a search algorithm is presented. The proposed binary GSO method has been tested in widely used two different uncapacitated facility location problem sets. The results of BinGSO were compared with both traditional and state-of-the-art binary methods. The proposed BinGSO method has performed more successful results than both traditional and state-of-the-art methods.

Table 10 Experimental results of the BinSSA, LS, ISS, and BinGSO method on the M^* Problems

	BinSSA			LS			ISS			BinGSO		
	Mean	Gap	Rank	Mean	Gap	Rank	Mean	Gap	Rank	Mean	Gap	Rank
MO1	1305.95	**0.0000**	**1**	1305.95	**0.0000**	**1**	1305.95	**0.0000**	**1**	1305.95	**0.0000**	**1**
MO2	1432.36	**0.0000**	**1**	1432.70	0.0090	2	1432.36	**0.0000**	**1**	1432.36	**0.0000**	**1**
MO3	1516.77	**0.0000**	**1**	1520.27	0.2300	2	1516.77	**0.0000**	**1**	1516.77	**0.0000**	**1**
MO4	1442.24	**0.0000**	**1**	1442.24	**0.0000**	**1**	1442.24	**0.0000**	**1**	1442.24	**0.0000**	**1**
MO5	1408.77	**0.0000**	**1**	1409.17	0.0290	2	1408.77	**0.0000**	**1**	1408.77	**0.0000**	**1**
MP1	2687.66	0.0440	4	2688.50	0.0750	3	2686.66	**0.0060**	**1**	2687.40	0.0344	2
MP2	2904.86	**0.0000**	**1**	2904.86	**0.0000**	**1**	2904.85	**0.0000**	**1**	2904.86	**0.0000**	**1**
MP3	2624.00	0.0111	2	2624.77	0.0400	4	2624.34	0.0240	3	2623.71	**0.0000**	**1**
MP4	2940.50	0.0600	3	2939.53	0.0260	2	2940.80	0.0690	4	2939.44	**0.0132**	**1**
MP5	2932.50	0.0058	2	2933.46	0.0380	4	2932.60	0.0090	3	2932.33	**0.0000**	**1**
MQ1	4091.01	**0.0000**	**1**	4091.01	**0.0000**	**1**	4091.01	**0.0000**	**1**	4091.01	**0.0000**	**1**
MQ2	4028.33	**0.0000**	**1**	4028.33	**0.0000**	**1**	4030.08	0.0430	2	4028.33	**0.0000**	**1**
MQ3	4275.43	**0.0000**	**1**	4275.43	**0.0000**	**1**	4275.43	**0.0000**	**1**	4275.43	**0.0000**	**1**
MQ4	4236.46	0.0310	3	4235.47	0.0070	2	4236.46	0.0310	3	4235.42	**0.0064**	**1**
MQ5	4086.45	**0.1400**	**1**	4086.53	0.1410	2	4095.46	0.3600	4	4086.62	0.1439	3
MR1	2610.24	0.0800	3	2608.24	**0.0030**	**1**	2647.03	1.4900	4	2609.30	0.0440	2
MR2	2655.73	0.0380	2	2654.73	**0.0030**	**1**	2691.54	1.3860	4	2661.15	0.2418	3
MR3	2790.14	0.0680	2	2789.04	**0.0280**	**1**	2832.33	1.5810	4	2793.47	0.1871	3
MR4	2756.04	**0.0000**	**1**	2756.04	**0.0000**	**1**	2807.90	1.8810	3	2768.34	0.4463	2
MR5	2505.40	**0.0140**	**1**	2505.48	0.0170	2	2549.97	1.7930	4	2510.25	0.2076	3
Winner/total		12/20			10/20			9/20			**13/20**	
Mean rank		1.6500			1.7500			2.3500			**1.5500**	

In future studies, the performance of the proposed method can be tested in different binary problems such as knapsack problems, unit commitment problems, and feature selection. In addition, the performance of different binary optimization methods in the GSO framework can also be analyzed. The performance of the GSO framework can be increased by developing different interaction strategies, such as implementing the local search phase or feedback mechanism from the second phase to the first phase in each epoch.

Declarations

Conflict of interest The author of the article have no relationship (either financial or personal) with any people or organizations that can affect or bias the paper's contents.

References

1. Islam M (2020) Optimization of the critical production process in a textile factory using AHP. Department of Mechanical and Production Engineering, Islamic University of Technology
2. Kotary J et al (2021). End-to-end constrained optimization learning: a survey. arXiv preprint arXiv: https://arxiv.org/abs/2103.16378
3. Krause J et al (2013) A survey of swarm algorithms applied to discrete optimization problems. Swarm intelligence and bio-inspired computation. Elsevier, pp 169–191
4. Zhan Z-H et al (2021) A survey on evolutionary computation for complex continuous optimization. Artif Intell Rev. https://doi.org/10.1007/s10462-021-10042-y
5. Christensen HI et al (2016) Multidimensional bin packing and other related problems: a survey
6. Salkin HM, De Kluyver CA (1975) The knapsack problem: a survey. Nav Res Logist Q 22(1):127–144
7. Turkoglu D C, Genevois M E (2020) A comparative survey of service facility location problems. Ann Oper Res 292(1):399–468. https://doi.org/10.1007/s10479-019-03385-x
8. Kong XY et al (2015) A simplified binary harmony search algorithm for large scale 0–1 knapsack problems. Expert Syst Appl 42(12):5337–5355
9. Hussien AG et al (2020) New binary whale optimization algorithm for discrete optimization problems. Eng Optim 52(6):945–959
10. Chen Y, Xie WC, Zou XF (2015) A binary differential evolution algorithm learning from explored solutions. Neurocomputing 149:1038–1047
11. Beheshti Z, Shamsuddin SM, Hasan S (2015) Memetic binary particle swarm optimization for discrete optimization problems. Inf Sci 299:58–84
12. Dhiman G, Oliva D, Kaur A, Singh K K, Vimal S, Sharma A, Cengiz K (2021) BEPO: a novel binary emperor penguin optimizer for automatic feature selection. Knowl Based Syst 211:106560. https://doi.org/10.1016/j.knosys.2020.106560
13. Jiang F et al (2017) A new binary hybrid particle swarm optimization with wavelet mutation. Knowl-Based Syst 130:90–101
14. Wang L, Zheng XL, Wang SY (2013) A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem. Knowl-Based Syst 48:17–23
15. Ozturk C, Hancer E, Karaboga D (2015) A novel binary artificial bee colony algorithm based on genetic operators. Inf Sci 297:154–170
16. Gölcük İ, Ozsoydan F B (2020) Evolutionary and adaptive inheritance enhanced grey wolf Optimization algorithm for binary domains. Knowl-Based Syst 194:105586. https://doi.org/10.1016/j.knosys.2020.105586
17. Luo K, Zhao Q (2019) A binary grey wolf optimizer for the multidimensional knapsack problem. Appl Soft Comput 83:105645. https://doi.org/10.1016/j.asoc.2019.105645
18. Kaya E, Kiran MS (2017) An improved binary artificial bee colony algorithm. In: 2017 15th international conference on ict and knowledge engineering (Ict&Ke) pp 29–34
19. Kiran MS, Gunduz M (2013) XOR-based artificial bee colony algorithm for binary optimization. Turk J Electr Eng Comput Sci 21:2307–2328
20. Bas E, Ulker E (2020) A binary social spider algorithm for continuous optimization task. Soft Comput 24(17):12953–12979
21. Hakli H, Ortacay Z (2019) An improved scatter search algorithm for the uncapacitated facility location problem. Comput Ind Eng 135:855–867
22. Nezamabadi-pour H (2015) A quantum-inspired gravitational search algorithm for binary encoded optimization problems. Eng Appl Artif Intell 40:62–75
23. Ghosh D (2003) Neighborhood search heuristics for the uncapacitated facility location problem. Eur J Oper Res 150(1):150–162
24. Yanasse HH, Soma NY (1987) A new enumeration scheme for the knapsack-problem. Discret Appl Math 18(2):235–245
25. James RJW, Nakagawa Y (2005) Enumeration methods for repeatedly solving multidimensional knapsack sub-problems. IEICE Trans Inf Syst 88(10):2329–2340
26. Lalami ME, El-Baz D (2012) GPU implementation of the branch and bound method for knapsack problems. In: 2012 IEEE 26th International parallel and distributed processing symposium workshops and phd forum. pp 1769–1777
27. Tohyama H, Ida K, Matsueda J (2011) A genetic algorithm for the uncapacitated facility location problem. Electron Commun Jpn 94(5):47–54
28. Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. Smc '97 conference proceedings. In: 1997 IEEE international conference on systems, man, and cybernetics. 1–5: 4104–4108
29. Greistorfer P, Rego C (2006) A simple filter-and-fan approach to the facility location problem. Comput Oper Res 33(9):2590–2601
30. Cinar AC, Kiran MS (2018) Similarity and logic gate-based tree-seed algorithms for binary optimization. Comput Ind Eng 115:631–646
31. Korkmaz S, Babalik A, Kiran MS (2018) An artificial algae algorithm for solving binary optimization problems. Int J Mach Learn Cybern 9(7):1233–1247
32. Korkmaz S, Kiran MS (2018) An artificial algae algorithm with stigmergic behavior for binary optimization. Appl Soft Comput 64:627–640
33. Jaramillo JH, Bhadury J, Batta R (2002) On the use of genetic algorithms to solve location problems. Comput Oper Res 29(6):761–779
34. Forestiero A, Mastroianni C, Spezzano G (2005) A Multi-agent approach for the construction of a peer-to-peer information system in grids. Self-Organization Auton Inform 135:220–236
35. Forestiero A, Mastroianni C, Spezzano G (2008) Building a peer-to-peer information system in grids via self-organizing agents. J Grid Comput 6(2):125–140
36. Dressler F, Akan OB (2010) A survey on bio-inspired networking. Comput Netw 54(6):881–900

37. Dujardin E, Mann S (2002) Bio-inspired materials chemistry. Adv Eng Mater 4(7):461–474
38. Eberhart RC, Shi YH (2001) Particle swarm optimization: developments, applications and resources. In: Proceedings of the 2001 congress on evolutionary computation.1, 2: 81–86
39. Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. IEEE Comput Intell Mag 1(4):28–39
40. Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 8(1):687–697
41. Yang X-S (2009) Firefly algorithms for multimodal optimization. In: International symposium on stochastic algorithms. Springer
42. Uymaz SA, Tezel G, Yel E (2015) Artificial algae algorithm (AAA) for nonlinear global optimization. Appl Soft Comput 31:153–171
43. Zhang XD et al (2016) Binary artificial algae algorithm for multidimensional knapsack problems. Appl Soft Comput 43:583–595
44. Chen J et al (2009) Optimal contraction theorem for exploration-exploitation tradeoff in search and optimization. IEEE Trans Syst Man Cybern Part A-Syst Hum 39(3):680–691
45. Muthiah-Nakarajan V, Noel MM (2016) Galactic swarm optimization: a new global optimization metaheuristic inspired by galactic motion. Appl Soft Comput 38:771–787
46. Kaya E, Uymaz SA, Kocer B (2019) Boosting galactic swarm optimization with ABC. Int J Mach Learn Cybern 10(9):2401–2419
47. Nguyen BM et al (2020) Hybridization of galactic swarm and evolution whale optimization for global search problem. IEEE Access 8:74991–75010
48. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: 1995 IEEE International conference on neural networks proceedings. 1–6: 1942–1948
49. Beasley JE (1990) Or-library: distributing test problems by electronic mail. J Oper Res Soc 41(11):1069–1072
50. Holland JH (1992) Genetic Algorithms. Sci Am 267(1):66–72
51. Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 1(6):80–83
52. Friedman M (1940) A comparison of alternative tests of significance for the problem of m rankings. Ann Math Statist 11(1):86–92
53. Yuan XH et al (2009) An improved binary particle swarm optimization for unit commitment problem. Expert Syst Appl 36(4):8049–8055
54. Kashan MH, Nahavandi N, Kashan AH (2012) DisABC: a new artificial bee colony algorithm for binary optimization. Appl Soft Comput 12(1):342–352
55. Kiran MS (2015) The continuous artificial bee colony algorithm for binary optimization. Appl Soft Comput 33:15–23
56. Kashan MH, Kashan AH, Nahavandi N (2013) A novel differential evolution algorithm for binary optimization. Comput Optim Appl 55(2):481–513
57. Engelbrecht AP, Pampara G (2007) Binary differential evolution strategies. In: 2007 IEEE congress on evolutionary computation. 1–10: 1942–1947
58. Cura T (2010) A parallel local search approach to solving the uncapacitated warehouse location problem. Comput Ind Eng 59(4):1000–1009