Neural Computing and Applications
https://doi.org/10.1007/s00521-024-09684-0

ORIGINAL ARTICLE

Evolution inspired binary flower pollination for the uncapacitated facility location problem

Fehmi Burcin Ozsoydan[1] · Ali Erel Kasırga[2,3]

Received: 16 September 2023 / Accepted: 25 March 2024

Abstract

The present paper introduces a modified flower pollination algorithm (FPA) enhanced by evolutionary operators to solve the uncapacitated facility location problem (UFLP), which is one of the well-known location science problems. The aim in UFLP is to select some locations to open facilities among a certain number of candidate locations so as to minimize the total cost, which is the sum of facility opening costs and transportation costs. Since UFLP is a binary optimization problem, FPA, which is introduced to solve real-valued optimization problems, is redesigned to be able to conduct search in binary domains. This constitutes one of the contributions of the present study. In this context, some evolutionary operators such as crossover and mutation are adopted by the proposed FPA. Next, the mutation operator is further enhanced by making use of an adaptive procedure that introduces greater level of diversity at earlier iterations and encourages intensification toward the end of search. Thus, while premature convergence and local optima problems at earlier iterations are avoided, a more intensified search around the found promising regions is performed. Secondarily, as demonstrated in this study, by making use of the reported evolutionary procedures, FPA is able to run in binary spaces without employing any additional auxiliary procedures such as transfer functions. All available benchmarking instances are solved by the proposed approach. As demonstrated by the comprehensive experimental study that includes statistically verified results, the developed approach is found as a promising algorithm that can be extended to numerous binary optimization problems.

Keywords Metaheuristics · Flower pollination algorithm · Uncapacitated facility location problem · Binary optimization

1 Introduction

Today, there emerges a very competitive environment in various sectors. Enterprises in both public and private sectors need to keep up with this competitive environment in order to be able to maintain their operations efficiently. Accordingly, to keep up with such a competitive environment, enterprises inevitably need to aim to reduce costs and increase customer satisfaction. This brings about a necessity for sustainable supply chains that are established to satisfy customer needs at the minimum cost and by not violating production constraints. As a result, this situation draws researchers' attention to location science problems, which still grab attention of numerous researchers and practitioners from various disciplines. Due to their mentioned strategic importance, location science problems can be considered as one of the commonly studied problems for decades. What is more, most of these problems are known to be in category of NP-hard problems. Therefore, employing metaheuristic algorithms are very common among researchers and practitioners particularly in solving large-scaled instances of location science problems.

As one of the emerging and attention-grabbing branch of metaheuristic algorithms, bio-inspired algorithms have become popular in the last decade. Bio-inspired algorithms are metaheuristics, which are based on metaphors in the simulated environment and they mimic the nature of living

✉ Fehmi Burcin Ozsoydan
burcin.ozsoydan@deu.edu.tr

Ali Erel Kasırga
erelkasirga@gmail.com

[1] Industrial Engineering Department, Faculty of Engineering, Dokuz Eylül University, İzmir, Türkiye

[2] Groupe Atlantic İzmir, İzmir, Türkiye

[3] Graduate School of Natural and Applied Sciences, Dokuz Eylül University, İzmir, Türkiye

Published online: 19 April 2024

beings in solving optimization problems. Although they do not guarantee optimality as metaheuristics; these algorithms are commonly used due to the capability of offering solutions of high-quality in acceptable time and being open to various improvements.

Unfortunately, most of the bio-inspired algorithms are devised for real-valued continuous domains, while majority of real-life problems have discrete and combinatorial spaces. To overcome this issue, researchers commonly adopt transfer functions. These functions transform a continuous vector into a binary vector. However, this results in various additional problems such as spatial distance, since, a continuous vector may not always result in the exact same binary vector on every occasion it is transformed. This undesirable circumstance brings about conflicts throughout search. Consequently, a promising vector may be underestimated by the algorithm, while some other vectors might unnecessarily or luckily be overestimated.

Such issues constitute the main motivation of the present study. The aim here is to address a simple yet effective approach to deal with spatial distance problems in bio-inspired algorithms. Accordingly, a procedure that borrows some evolutionary operators, which can run in binary spaces, is adopted first. The mentioned procedure is used along with the notion of flower pollination algorithm (FPA) [1], which can be considered as a relatively recent bio-inspired algorithm that has been shown to be promising in various problems. The motivation in choosing this algorithm stems from its simplicity in terms of both complexity and intelligibility and being open to various improvements. In addition to these enhancements in FPA, secondarily, an adaptive mutation procedure that introduces greater level of diversity at earlier iterations and encourages intensification toward the end of search is adopted. Thus, while premature convergence and local optima problems at earlier iterations are avoided, a more intensified search around the found promising regions is conducted. Accordingly, the distinguishing features of the proposed FPA can be brought together in twofold: (i) the proposed modification is capable of handling binary spaces without any auxiliary procedures such as transfer functions, (ii) it has a further convergence capability that also enables avoiding local optima. The performance of the proposed approach is tested on the uncapacitated facility location problem (UFLP), which is one of the well-known location science optimization problems and has numerous potential real-life applications in practice. A comprehensive experimental analysis including statistical demonstrations, convergence analysis and complexity analysis is conducted. According to the statistically verified results, the proposed approach is found as a promising algorithm in solving UFLP.

The rest of this paper is organized as follows: Related literature review for UFLP, research gaps and potential applications are given in Sect. 2. The proposed solution approach and developed procedures are presented in Sect. 3. Experimental study and concluding remark are reported in Sects. 4 and 5, respectively.

2 UFLP and related studies

2.1 A formal introduction for UFLP

UFLP [2] is one of the most commonly studied location science problems, and it forms a basis for a variety of related models used in supply chain design. As discussed before, the aim in UFLP is to determine a set of facilities among a certain candidate locations so as to minimize the total cost, which is the sum of facility opening costs and transportation costs between the opened facilities and demand points (customers). More formally, UFLP can be formulated as follows:

$$\text{Minimize} \sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij}x_{ij} + \sum_{i=1}^{N} f_i y_i \tag{1}$$

$$\sum_{i=1}^{N} x_{ij} = 1 j = 1, \ldots, M \tag{2}$$

$$x_{ij} \leq y_i \quad i = 1, \ldots, N j = 1, \ldots, M \tag{3}$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \ldots, N j = 1, \ldots, M \tag{4}$$

$$y_i \in \{0, 1\} \quad i = 1, \ldots, N j = 1, \ldots, M \tag{5}$$

In these formulations, Eq. (1) represents the objective function to be minimized. In Eq. (1), N is the number of potential facility locations, M is the number of demand points, f_i is the facility opening cost associated to ith point, c_{ij} is the cost occurring between facility point i and the demand point j. Equation (2) ensures that all demands are satisfied. Equation (3) provides that customers can be assigned only to opened facilities. Accordingly, Eqs. (4) and (5) impose sign restrictions on decision variables. The binary variable y_i denotes whether the facility i is opened or not. If the candidate facility at location i is opened, the decision variable y_i takes value of 1; otherwise, it takes value of 0. The other binary variable x_{ij} takes value of 1 if the jth demand point is assigned to facility i; otherwise, it takes value of 0.

2.2 Related studies

To the best of our knowledge, Balinski [3] first introduces a mathematical model for UFLP. Later, Efroymson and Ray [4] report a modification on this model and develop a branch and bound algorithm. Next, Khumawala [5] proposes a more efficient branch and bound algorithm. By using ascent and adjustment procedures, Erlenkotter [6] introduces the dual based branch and bound algorithm. Subsequently, by rearranging the primal–dual version of the algorithm developed by Erlenkotter [6], Korkel [7] proposes a more efficient algorithm in terms of required time.

Since UFLP is shown to be an NP-hard problem [8], various metaheuristic solution methods are reported as well. Al-Sultan and Al-Fawzan [9] present an approach that is based on Tabu search (TS) algorithm to solve the UFLP. Ghosh [10] studies the performance of generic local search, TS and complete local search with memory. Ghosh [10] proves that both of these approaches offer effective ways of generating high-quality solutions for large-scaled instances. In 2006, Şevkli and Güner [11] develop a particle swarm optimization (PSO) algorithm to solve UFLP. Two years later, Wang et al. [12] and Güner and Şevkli [13] propose a PSO to solve UFLP. Next, Kiran [14] implements a bio-inspired algorithm for UFLP. A constructive algorithm is employed by Montoya-Torres et al. [15]. According to the reported results, the proposed algorithm can be considered as an efficient approach in solving UFLP. Tsuya et al. [16] use another bio-inspired algorithm to solve UFLP. Later, de Armas et al. [17] solve the stochastic extension of UFLP. Subsequently, Baykasoglu et al. [18] present a binary weighted superposition attraction algorithm to solve UFLP. Ozsoydan [19] proposes a modified swarm-based intelligence algorithm to solve a set of binary optimization problems including UFLP. Ozsoydan [20] and Golcuk and Ozsoydan [21] implement modifications of a bio-inspired algorithm, namely grey wolf optimization algorithm to solve UFLP.

One year later, Sonuç [22] proposes another bio-inspired metaheuristic algorithm that is further extended so as to be capable of handling binary optimization problems. Authors adopt the UFLP as the benchmarking suite. Zhang et al. [23] propose an evolutionary algorithm to solve UFLP. In one of the recent studies, Alidaee and Wang [24] propose a genetic algorithm (GA) hybridized by TS to solve uncapacitated location problem, which is a natural generalization of the UFLP. The authors report promising results. In another recent study, Kaya [25] introduces a binary modification of a swarm-based optimizer that is driven by a bio-inspired algorithm. The proposed algorithm exhibits a favorable performance in solving UFLP. Zhou et al. [26] propose an improved GA and the authors address a real-life application in oil and gas fields. Lately, Soltanpour et al. [27] report efficient and promising algorithms for UFLP. The authors carry out the same analysis also for uncertain problem environment that includes uncertain data. Jiang and Zhang [28] propose an improved adaptive differential evolution (DE) algorithm for solving UFLP. It is shown by the computational results that the proposed approach outperforms other compared swarm-based algorithms. In a recent study, Baş and Yildizdan [29] propose a novel binary arithmetic optimization algorithm for UFLP. The authors adopt logic gates to deal with continuous variables, since, as reported, the canonical form the adopted algorithms is proposed for continuous spaces. Finally, in another recent work, Aslan and Pavone [30] propose another binary extension of a metaheuristic algorithm, namely vortex search algorithm for solving UFLP. As reported in that study, authors adopt several versions of transfer functions.

There exist numerous FPA studies in the literature. Due to space limitation, only closely related ones are reported here. In 2015, Dubey et al. [31] propose a hybrid FPA with a time-varying fuzzy selection methodology for wind in integrated multi-objective dispatch. In the same year, Rodrigues et al. [32] propose a binary modification for FPA by using a sigmoid transfer function. The authors test their approach on a binary optimization problem, namely feature selection problem, which is a well-known problem in machine learning domain. Dahi et al. [33] report a systematic evaluation study that addresses the efficiency of mapping techniques. The authors propose binary variants of FPA so as to evaluate the principal mapping techniques existing in the literature. Next, Nabil [34] presents a modified FPA, which is called opposition-based learning FPA. This algorithm qualitatively and quantitatively analyzes FPA with the proposition of various extensions. Abdel-Baset et al. [35] propose a hybridized FPA to solve constrained optimization problems. In same year, Draa [36] proposes some new extensions for FPA by employing opposition-based learning, modified global search and local search procedures explicitly. Zhou et al. [37] report discrete greedy FPA that is based on the integration between pollen discarding behavior and order-based crossover to get global pollination process. Ozsoydan and Baykasoglu [38] analyze the effects of various switching probability characteristics that directly affects the success of FPA. In one of the recent studies, Song et al. [39] hybridize DE with FPA to solve global optimization problems. The reported results by the authors demonstrate the efficiency of the proposed approach which points out the potential of FPA. Another similar approach is adopted by Mellal et al. [40] to deal with combined heat and power economic dispatch problem, which is shown to be a constrained optimization problem. Finally, in a recent study, Feng et al. [41] propose a binary hybrid artificial hummingbird with FPA for feature selection in Parkinson's disease diagnosis. The authors adopt a sigmoid transfer function to deal with binary decision variables.

2.3 Research gaps, motivation and potential extensions

As stated above, the related literature for UFLP includes both swarm-based methods and trajectory-based (vector-based) algorithms. These methods are further hybridized in some studies. Swarm-based algorithms include both bio-inspired algorithms and well-known metaheuristics such as GA and DE. In this regard, to the best of our knowledge, the performance of FPA, which has been shown to be promising in numerous optimization problems, has not been tested on UFLP yet. Moreover, it is apparent from the related literature that UFLP and its extensions are quite open to real-life application studies. Thus, the proposed modified FPA along with all reported extensions and procedures is tested on UFLP in this study.

Secondarily, it is obvious from the reported related literature that, all metaheuristic algorithms adopt either various transfer functions or some modifications of these functions particularly while using swarm-based and bio-inspired metaheuristic algorithms. As mentioned earlier, this approach results in spatial distance problem, which is not a desirable circumstance for an optimization algorithm. Moreover, it is worth stressing that this undesirable issue applies also for various binary optimization problems. In other words, the mentioned spatial distance problem does not apply only for UFLP. Accordingly, spatial distance issue can be considered as a common drawback while solving binary optimization problems by using swarm-based algorithms that work on real-valued vectors. Therefore, the proposed approach in the present study has numerous potential extensions and applications for various binary optimization problems such as variants of the famous knapsack problems, feature selection or different facility location problems. Moreover, it is clear the proposed approach can be extended to numerous other metaheuristic algorithms that make use of real-valued vectors such as FPA. Accordingly, the next section is devoted to developed solution approaches.

3 Proposed solution approach

3.1 The canonical FPA

FPA [1] is a metaheuristic algorithm that is inspired by the pollination process of flowers. There are numerous flowering plants in nature, and there exist many different means of pollination to achieve the purpose of reproduction. In general, pollination methods can take two forms, namely self-pollination and cross-pollination. Cross-pollination (biotic) depends on living beings such as insects and birds at long distances. This type of pollination represents global search in FPA. Self-pollination (abiotic) on the other hand takes place at a shorter distance and this inspires local search process of FPA. To sum up, as global search and local search, FPA has two main and simple mechanisms. According to a switching probability, algorithm either switches from global to local or from local to global search.

Before diving into details of the developed procedures, the canonical FPA is presented first. In the global search procedure of FPA, solution vectors implement Lévy flight, following the best-found position. Thus, the movement pattern resembles living beings like bees and insects. As the second procedure of FPA, self-pollination takes place as random changes on solution vectors. One can put control on these search patterns via a parameter referred to as switch probability that is denoted by $p \in (0, 1)$. While greater values of p encourages global search, smaller values yield to more frequent use of local search. More formally, all movements of FPA are formulated in the following equations:

$$x_i^{t+1} = x_i^t + L(g^* - x_i^t) \tag{6}$$

$$L \sim \frac{\lambda\Gamma(\lambda)\sin(\pi\lambda/2)}{\pi s^{1+\lambda}}, (s \gg s_0 > 0) \tag{7}$$

$$x_i^{t+1} = x_i^t + r(x_j^t - x_k^t)k \neq j \tag{8}$$

where g^*, x_i^t, L and r represent the best solution found by the algorithm, the ith solution vector at the tth iteration, the Lévy distribution and a random variable, respectively. Equation (6) here represents the global pollination, while Eq. (8) is the local pollination. Equation (7) is employed along with Eq. (6) and $\Gamma(\lambda)$ represents the standard gamma function. For a better understanding of the mechanisms of the canonical FPA, a pseudocode is presented by Algorithm 1.

Algorithm 1. A pseudocode for the canonical FPA

```
1:  initialize FPA population of $popSize$ solution vectors
2:  evaluate fitness values $f(x_i)$, $i = 1, \ldots, popSize$
3:  while $(t < maxIt)$ // $maxIt$ is the maximum number of generations
4:      for $i = 1{:}popSize$
5:          if $rand < p$
6:              generate a vector L from Lévy distribution (Eq. 7)
7:              apply global pollination (Eq. 6)
8:          else
9:              generate a random variable r
10:             apply local pollination (Eq. 8)
11:         end if
12:         evaluate the new solution
13:         update it if necessary
14:     end for
15:     update the global best solution g* if necessary
16: end while
```

3.2 Proposed modifications

It is clear that the canonical FPA is proposed for real-valued optimization problems. Thus, adopting this algorithm to binary domains might require some additional mapping procedures such as employing transfer functions and most of these procedures make use of probability distributions. It further means that two binary vectors derived from the same real-valued vector do not necessarily have to be same. This undesirable circumstance brings about spatial distance between these vectors. Therefore, although such transfer functions are commonly used in numerous studies, their performances tend to be poor in most cases. In this regard, the present paper adopts some evolutionary operators in order to use them in the notion on FPA so as to overcome spatial distance issue.

3.2.1 Solution encoding

The proposed modification adopts the binary encoding technique. Solution vectors in this technique are of size *n*, which represents the number of candidate locations to open facilities for the corresponding problem instance. If a bit takes a value of one, then it means that the corresponding facility is opened. Otherwise, the corresponding locus of the solution vector points out that the regarding facility is not opened. All bits (loci) of a solution vector take only binary values and are never changed to a real values throughout iterations. Finally, it is worth stressing that the initial population that includes only binary solution vectors is generated at random. The rest of the details are presented in the following.

3.2.2 Uniform-based xover for inheritance

As mentioned earlier, one of the notable advantages of FPA is that it has a small number of parameters. One of the parameters used in FPA is the switching probability that is denoted by p (Algorithm 1, line: 5), and this parameter defines the search characteristic by putting control on the transitions between global and local search procedures. If p is increased, FPA tends to exploit around the best-found solution. Lower values for p on the other hand yield to making use of randomly selected solution vectors from the population more frequently. It is clear that according to Eqs. (6–8), these procedures cannot be directly used in binary domains, since, a vector-based operation that uses Eqs. (6–8) very likely results in a continuous vector regardless of the input vectors. This further means that solution vectors that are generated as random binary vectors in the initial population are not guaranteed to remain as binary vectors throughout search. Although transfer functions might come in handy at this point, performance of the optimizer might suffer from the spatial distance issue that is previously mentioned. Therefore, in the present study, a uniform-based xover, which exhibits loci exchanges, is adopted to inherit information between individuals [19].

In this procedure, any *i*th solution vector in population undergoes a uniform-based xover with a parent vector that is either a random vector or the best-found solution vector.

For each dimension of solution vectors, a random number $rand \in (0,1)$ is generated and it is compared to a user-supplied parameter $xOverProb \in (0,1)$. If $rand \leq xOverProb$, allele of the corresponding locus is inherited from the *i*th vector. Otherwise, it is inherited from the parent vector, which probably yields to change of the corresponding bit. Thus, one can conclude that lower values of *xOverProb* results in dampening the changes on the bits. In other words, inertia is encouraged by using lower values of *xOverProb*. Greater values on the other hand yield to changes for the related loci.

An illustration of this procedure is presented by Fig. 1. As one can see from this figure, solution vectors are of size 10 bits. Thus, 10 *rand* values are to be generated. Let's assume that these random numbers appear to be 0.50, 0.10, 0.80, 0.70, 0.05, 0.90, 0.15, 0.45, 0.75, and 0.80, for each dimension, respectively. Given that *xOverProb* is set to be equal to 0.60, new solution vector, denoted by offspring, is generated as given in this figure.

It should be noted that the values of *p* and *xOverProb* should be carefully chosen. Since, greater values of *p* and lower values of *xOverProb* yield to a greater exploitation of the best-found solution. It is clear that overuse of this solution vector might yield to premature convergence and local optima issues. On the contrary, lower values of *p* and greater values of *xOverProb* yield a more random and slower search that might also result in poor convergence. Thus, it can be concluded that both of these parameters simultaneously affect the search characteristic of the proposed FPA modification.

3.2.3 Bit-flip mutations to avoid local optima

Subsequent to xover operations discussed above, a bit-flip mutation is performed. The aim here is to introduce diversity so as to avoid local optima problems. In this procedure, randomly selected bits of a solution vector are flipped. It is clear that greater number of flipped bits introduce greater level of diversity. This becomes useful whether the population loses diversity. Lower level of mutation on the other hand yields to a better exploitation of the found regions; however, continuous use of lower level of mutation might result in loss of diversity and a slower search, where local optima issues might arise. Therefore, an adaptive mutation that encourages the number of mutations at earlier iterations and discouraging mutation gradually over iterations is adopted [20]. In other words, this procedure [18, 19, 21] provides a greater level of diversity at earlier iterations along with a decreased level of mutation toward the end of the search so as to encourage exploitation. Thus, local optima and premature convergence problems are aimed to be avoided while providing a more intensified search over iterations.

This procedure is formulated by Eq. (9), where $stepSize^t \in [1,n]$ and $\phi \in (0,1)$ denote the step size at the *t*th iteration and the severity of the decrement, respectively. Step size in this formulation represents a variable in order to evaluate the number of bits ($mutSize^t$) that undergo mutation operation at the *t*th iteration. Obviously, Eq. (9) yields to a continuous variable; however, the number of bits to mutate is a discrete value variable. Therefore, a ceiling function (Eq. 10) is used to determine the number of bits to mutate at the *t*th iteration. Subsequently, a number of randomly selected $mutSize^t$ bits of a solution vector at the *t*th iteration is flipped, which means that it is flipped to 1 if it is 0 or it is flipped 0 if it is 1.

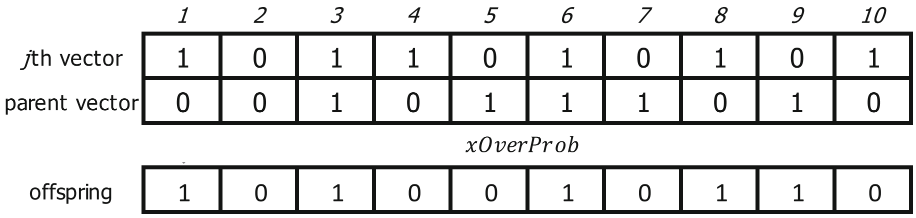
Both $stepSize^1$ and ϕ are user-supplied parameters. Greater values of ϕ induce steeper decrements in step size, whereas smaller values yield to a smoother decrement. A pseudo code and an illustration for the mentioned mutation operator for $stepSize^1$=30 are presented by Algorithm 2 and Fig. 2, respectively. Moreover, a tabular form of some $mutSize^{t+1}$ values with respect to various $stepSize^1$ and ϕ parameters are presented in Table 1. Finally, bringing all modifications together, a pseudocode for the proposed binary FPA (bFPA) is presented by Algorithm 3.

$$\mathrm{stepSize}^{t+1} = \mathrm{stepSize}^{t} - e^{-t/(t+1)} \times \phi \times \mathrm{stepSize}^{t} \qquad (9)$$

$$\mathrm{mutSize}^{t+1} = \lceil \mathrm{stepSize}^{t+1} \rceil \qquad (10)$$

Algorithm 2. A pseudocode for the mutation operator

```
for $m = 1{:}mutSize^t$
    select a random locus $\omega$
    while locus $\omega$ is already chosen
        select a random locus $\omega$
    end while
    $offSpring_\omega = 1 - offSpring_\omega$
end for
```



	1	2	3	4	5	6	7	8	9	10
*j*th vector	1	0	1	1	0	1	0	1	0	1
parent vector	0	0	1	0	1	1	1	0	1	0
offspring	1	0	1	0	0	1	0	1	1	0

Fig. 1 Uniform-based xover

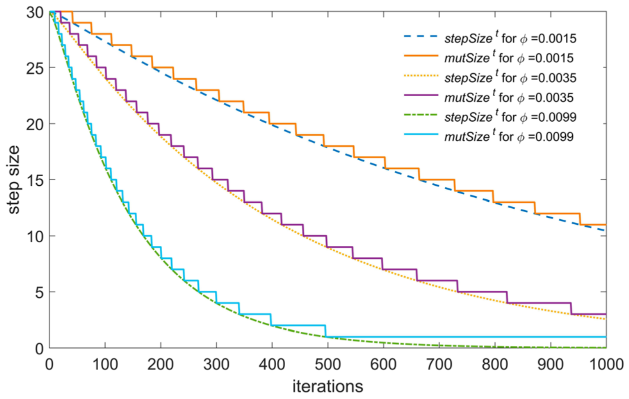


Fig. 2 An illustration of various step size settings

Algorithm 3. A pseudocode for the proposed bFPA

```
initialize FPA population of $popSize$ solution vectors
evaluate fitness values $f(x_i)$, $i = 1,\ldots,popSize$
while ($t < maxIt$)  // $maxIt$ is the maximum number of generations
    calculate $stepSize^t$ and $mutSize^t$
    for $i = 1{:}popSize$
        if $rand < p$
            $offSpring$ = xOver($pop_i$, $g^*$)
        else
            $offSpring$ = xOver($pop_i$, $pop_k$)  // $i \neq k$
        end if
        $offSpring$ = mutation($offspring$, $mutSize^t$)
        evaluate $offSpring$
        update it if necessary
    end for
    update the global best solution $g^*$ if necessary
end while
// $pop_k$ is any solution vector randomly selected from the population.
```

Table 2 Attributes of the used UFLP instances

Problem ID	Size (facilities × customers)	Optimum solution value
cap71	16 × 50	932,615.750
cap72	16 × 50	977,799.400
cap73	16 × 50	1,010,641.450
cap74	16 × 50	1,034,976.975
cap101	25 × 50	796,648.437
cap102	25 × 50	854,704.200
cap103	25 × 50	893,782.112
cap104	25 × 50	928,941.750
cap131	50 × 50	793,439.562
cap132	50 × 50	851,495.325
cap133	50 × 50	893,076.712
cap134	50 × 50	928,941.750

It is clear that the proposed FPA modification denoted by bFPA can be distinguished from the standard FPA in two ways. First, the proposed modification is capable of handling binary spaces without any auxiliary procedures such as transfer functions. By making use of the evolutionary operators in FPA notion, this issue is overcome intrinsically. Secondarily, by using the discussed mutation procedure, bFPA has a further convergence capability that also enables avoiding local optima. Moreover, binary solution vectors that undergo xover operations are guaranteed to remain as binary vectors also subsequent to this mutation procedure, since, this procedure exhibits only bit-flip operations on loci, which already have binary values.

4 Experimental study

4.1 Calibration of parameters

All tests are performed on a computer with an i5-4590 CPU and 32 GB of RAM. Maximum number of iterations (*maxIt*) is used as the termination criterion. In accordance with the related literature, the parameters *maxIt* and population size (*popSize*) are fixed to 1000 and to the number

Table 1 A tabular illustration of $mutSize^t$ values for various mutation parameters

	$stepSize^1$=50			$stepSize^1$=20		
iterations (t)	ϕ=0.0025 $mutSize^t$	ϕ=0.0050	ϕ=0.0075	ϕ=0.0010 $mutSize^t$	ϕ=0.0025	ϕ=0.0050
1	50	50	50	20	20	20
50	47	44	41	20	19	18
100	43	37	32	19	18	15
250	33	22	15	17	14	9
500	22	9	4	15	9	4
750	14	4	1	12	6	2
900	11	3	1	11	5	1
950	10	2	1	11	4	1
1000	9	2	1	10	4	1

of candidate facility locations (n) for the related instance, respectively.

As another parameter, *xOverProb* represents inertia. According to the preliminary tests, it is observed that better results can be obtained when *xOverProb* is fixed to 0.20. Therefore, this parameter is used as 0.20 for all tests. As discussed before, there exist two additional parameters that define the mutation characteristic in the proposed approach. They are denoted by ϕ and $stepSize^1$, which represent the speed of decrement in step-sizing function and the initial step size, respectively. With respect to the preliminary tests again, it is observed that $\phi = 0.0025$ and $stepSize^1 = 0.10 \times n$ yield to more promising results. Therefore, the parameters ϕ and $stepSize^1$ are fixed to 0.0025 and $0.10 \times n$, respectively, for all tests. The final parameter to be calibrated is the switching probability of FPA that is denoted by p. This parameter is fixed to 0.75 for all tests, which is close to the value proposed by Yang [1].

4.2 Computational results

The performance of the bFPA is tested by using UFLP benchmarking instances obtained from OR-Lib (http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/). The related attributes of these instances are presented in Table 2. Moreover, as presented by the same table, optimum solutions for the instances in this benchmark suite are known and available.

Instances here can be categorized into three categories as small-scaled, medium-scaled and large-scaled problems that include 16, 25 and 50 facilities, respectively. All obtained results are given by Table 3, and the values in this table are calculated over 30 independent replications. The rows *best*, *mean*, *worst*, *std*. and *hit* in this table represent the best-found solution by the proposed algorithm, mean over all replications, worst-found solution, standard deviance of the best-found solutions over all replications and the number of replications that the best-known solution is found, respectively. Moreover, in order to analyze the effects of the adopted mutation procedure, bFPA is additionally tested without mutation operation. This modification is referred to as bFPAnm in the rest of the paper.

The first section of Table 3 includes small-scaled instances of size 16 × 50. According to these results, most of the algorithms exhibit competitive performances. It is clear that particularly more recently published algorithms such as ABCbin, bWSA, PSO, prioGWO, learnGWO, prLeGWO and the proposed algorithm bFPA are capable of finding the most promising *hit values* in these instances. It is also clear that GWO occasionally misses to find the most promising hit values in some instances such as cap72 and cap73. CPSO exhibits a similar performance with GWO in terms of finding the most promising hit values. Finally, a notable finding in this section of Table 3 is that bFPAnm achieves the most naïve results for the small-scaled instances of UFLP. This finding also points out the efficiency of the adopted mutation procedure, which is employed by bFPA.

Results of the medium-scaled instances of size 25 × 50 point out that the difference in performances of all compared algorithms become more apparent as the problem scale increases. It is apparent that the proposed algorithm bFPA achieves the best-known solutions in all replications for all instances of the medium-scaled benchmarks. bFPA is followed by prLeGWO, which is capable of achieving a hit value of 26 in cap103. The algorithm prLeGWO is followed by learnGWO, which shows a better performance in comparison with the rest of the compared algorithms. CPSO shows an adequate performance in some instances such as cap102 and cap104. However, it cannot achieve the optimum solution in any replications of the remaining medium-scaled instances. Moreover, bFPAnm is also not capable of finding the mentioned optimum solutions in any replications of the medium-scaled problems as well. Additionally, its standard deviance value is found far greater than that of the CPSO's. This finding points out some possible local optima issues. Therefore, one can conclude that the adopted mutation operator contributes to the performance of bFPA also in medium-scaled instances.

Difference between the compared algorithms become even more apparent in the large-scaled instances of size 50 × 50. Except for the instance cap133, where the proposed algorithm achieves a *hit value* of 26, bFPA is able to find the best-known solutions in all replications in the remaining instances of the same size. This can be considered as a remarkable performance since the closest algorithm in terms of *hit values*, prLeGWO is clearly outperformed by the proposed algorithm. In parallel, bFPA is found as superior to the rest of the algorithms in the large-scaled instances in regard to the *hit values*. Similarly, the standard deviance values of all compared algorithms for cap133 point out that the proposed algorithm exhibits a more robust performance in this instance. Moreover, the standard deviation values obtained by the proposed algorithm bFPA are often better than those found by other algorithms. Finally, it is clear that bFPAnm exhibits the most naïve performance also in the large-scaled instances of UFLP. Standard deviance values, which are found as the far greatest values among those of all compared algorithms', point out possible local optima issues also in these instances.

Convergence plots of bFPA and bFPAnm for all instances are depicted by Fig. 3. The rows in this figure correspond to the small-scaled, medium-scaled and large-scaled instances, respectively. Data points in these

Table 3 Computational results

Algorithms		Instances (facilities × demand points)											
		16 × 50				25 × 50				50 × 50			
		cap71	cap72	cap73	cap74	cap101	cap102	cap103	cap104	cap131	cap132	cap133	cap134
CPSO	*best*	932,615.75	977,799.40	1,010,641.45	1,034,976.98	796,648.44	854,704.20	893,782.11	928,941.75	795,291.86	851,495.33	893,076.71	928,941.75
	mean	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*
	worst	934,199.14	983,713.81	1,012,643.69	1,045,342.23	802,457.23	857,380.85	899,424.91	944,394.83	804,549.64	865,667.16	909,908.7	951,803.25
	std	562.23	1324.3	702.13	2124.54	1480.72	1015.64	1695.79	3842.64	2429.54	4297.07	4210.93	6619.05
	hit	25	25	22	0.00	0.00	10	0.00	18	0.00	0.00	0.00	7
ABCbin	*best*	932,615.75	977,799.40	1,010,641.45	1,034,976.98	796,648.44	854,704.20	893,782.11	928,941.75	793,439.56	851,495.33	893,076.71	928,941.75
	mean	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*
	worst	932,615.75	977,799.40	1,010,641.45	1,034,976.98	796,648.44	854,704.20	894,008.14	928,941.75	794,910.64	851,636.70	895,407.93	928,941.75
	std	0.00	0.00	0.00	0.00	0.00	0.00	85.67	0.00	1065.73	213.28	561.34	0.00
	hit	**30**	**30**	**30**	**30**	**30**	**30**	25	**30**	6	14	5	**30**
bWSA	*best*	932,615.75	977,799.40	1,010,641.45	1,034,976.97	796,648.43	854,704.20	893,782.11	928,941.75	793,439.56	851,495.32	893,076.71	928,941.75
	mean	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*
	worst	932,615.75	977,799.40	1,010,641.45	1,034,976.97	797,508.72	854,704.20	895,027.18	928,941.75	798,449.03	852,257.97	894,801.16	934,586.97
	std	0.00	0.00	0.00	0.00	380.43	0.00	470.95	0.00	1025.78	251.65	501.91	1016.14
	hit	**30**	**30**	**30**	**30**	22	**30**	10	**30**	6	23	7	26
PSO	*best*	932,615.75	977,799.40	1,010,641.45	1,034,976.97	796,648.43	854,704.20	893,782.11	928,941.75	803,826.01	879,785.46	921,608.03	944,335.68
	mean	932,615.75	977,799.40	1,010,641.45	1,034,976.97	799,006.37	857,115.01	896,226.99	933,746.35	817,976.18	889,889.93	943,216.76	1,002,804.33
	worst	932,615.75	977,799.40	1,010,641.45	1,034,976.97	801,769.38	860,573.81	900,406.76	941,871.31	824,751.43	896,935.53	957,975.85	1,035,373.88
	std	0.00	0.00	0.00	0.00	1085.48	1268.63	1763.88	3358.73	4545.68	4697.11	9163.94	19,102.20
	hit	**30**	**30**	**30**	**30**	1	2	3	2	0	0	0	0
GWO	*best*	932,615.75	977,799.40	1,010,641.45	1,034,976.97	796,648.43	854,704.20	893,782.11	928,941.75	793,439.56	851,495.32	893,076.71	928,941.75
	mean	932,615.75	978,056.06	1,010,702.63	1,034,976.97	797,123.96	854,830.95	894,083.99	928,941.75	794,606.70	852,034.68	894,088.39	929,716.82
	worst	932,615.75	981,649.35	1,012,476.97	1,034,976.97	799,144.68	855,971.75	895,027.18	928,941.75	797,570.30	854,704.20	896,529.71	935,122.70
	std	0.00	976.76	335.11	0.00	772.61	386.76	480.63	0.00	1292.51	789.94	1006.16	1783.31
	hit	**30**	28	29	**30**	19	27	20	**30**	11	13	5	20
prioGWO	*best*	932,615.75	977,799.40	1,010,641.45	1,034,976.97	796,648.43	854,704.20	893,782.11	928,941.75	793,439.56	851,495.32	893,076.71	928,941.75
	mean	932,615.75	977,799.40	1,010,641.45	1,034,976.97	796,963.87	854,788.70	894,102.76	928,941.75	794,187.91	851,735.81	893,925.90	929,612.32
	worst	932,615.75	977,799.40	1,010,641.45	1,034,976.97	797,508.72	855,971.75	895,027.18	928,941.75	796,662.92	853,419.13	895,049.66	934,586.97
	std	0.00	0.00	0.00	0.00	421.65	321.58	459.7	0.00	1000.79	446.02	587.93	1699.92
	hit	**30**	**30**	**30**	**30**	19	28	15	**30**	15	17	6	21
learnGWO	*best*	932,615.75	977,799.40	1,010,641.45	1,034,976.97	796,648.43	854,704.2	893,782.11	928,941.75	793,439.56	851,495.32	893,076.71	928,941.75
	mean	932,615.75	977,799.40	1,010,641.45	1,034,976.97	796,849.17	854,704.2	894,016.05	928,941.75	794,749.63	852,229.93	894,390.11	929,834.29
	worst	932,615.75	977,799.40	1,010,641.45	1,034,976.97	797,508.72	854,704.2	894,801.16	928,941.75	798,338.45	855,005.57	897,548.76	934,586.97
	std	0.00	0.00	0.00	0.00	370.08	0.00	406.54	0.00	1476.05	1212.19	1214.52	1786.97
	hit	**30**	**30**	**30**	**30**	23	**30**	20	**30**	10	16	5	20
prLeGWO	*best*	932,615.75	977,799.40	1,010,641.45	1,034,976.97	796,648.43	854,704.20	893,782.11	928,941.75	793,439.56	851,495.32	893,076.71	928,941.75
	mean	932,615.75	977,799.40	1,010,641.45	1,034,976.97	796,648.43	854,704.20	893,782.11	928,941.75	793,781.44	851,616.90	893,717.00	928,941.75
	worst	932,615.75	977,799.40	1,010,641.45	1,034,976.97	796,648.43	854,704.20	894,801.16	928,941.75	794,373.41	853,525.52	894,801.16	928,941.75
	std	0.00	0.00	0.00	0.00	0.00	0.00	260.79	0.00	426.88	429.35	571.45	0.00
	hit	**30**	**30**	**30**	**30**	**30**	**30**	26	**30**	18	26	10	**30**
bFPAnm	*best*	932,615.75	977,799.40	1,010,641.45	1,034,976.975	797,582.28	859,326.91	895,275.68	932,985.32	804,016.88	890,796.27	927,926.47	1,007,301.17
	mean	936,843.21	982,617.05	1,014,524.75	1,044,652.87	805,005.67	866,313.30	908,813.74	952,354.92	826,843.15	908,193.54	972,196.37	1,070,102.14
	worst	952,211	989,636.73	1,021,927.04	1,061,140.75	814,537.96	884,490.87	935,684.76	985,138.10	840,377.46	931,258.31	1,014,984.75	1,118,548.92

Table 3 (continued)

Algorithms		Instances (facilities × demand points)											
		16 × 50				25 × 50				50 × 50			
		cap71	cap72	cap73	cap74	cap101	cap102	cap103	cap104	cap131	cap132	cap133	cap134
	std	4259.88	2808.78	3449.03	7445.91	4428.54	5403.44	8222.90	11,949.76	6750.57	12,408.11	20,496.44	32,134.54
	hit	5	5	2	3	0	0	0	0	0	0	0	0
bFPA	*best*	932,615.75	977,799.40	1,010,641.45	1,034,976.975	796,648.438	854,704.20	893,782.113	928,941.75	793,439.563	851,495.325	893,076.713	928,941.75
	mean	932,615.75	977,799.40	1,010,641.45	1,034,976.975	796,648.438	854,704.20	893,782.113	928,941.75	793,439.563	851,495.325	893,170.766	928,941.75
	worst	932,615.75	977,799.40	1,010,641.45	1,034,976.975	796,648.438	854,704.20	893,782.113	928,941.75	793,439.563	851,495.325	893,782.113	928,941.75
	std	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	243.89	0.00
	hit	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	26	**30**

plots represent mean results of the related iteration over all replications. As one can see from this figure, bFPA exhibits a better convergence particularly in small-scaled and medium-scaled instances in comparison with bFPAnm. It is clear from these plots that results given by Table 3 can be achieved also by using smaller values for maximum number of iterations. However, one should note that in such a case, the parameters of the step-sizing function (Eqs. 9–10) should be recalibrated according to the used maximum number of iterations parameter. Moreover, one can put forward that bFPAnm suffers from premature convergence and local optima issues, whereas bFPA which uses the proposed mutation operator exhibits a more adequate convergence pattern. This also validates that the mentioned mutation procedure contributes to the performance of the proposed algorithm bFPA.

All findings point out the efficiency of the proposed algorithm bFPA. However, in order to demonstrate whether significant improvements are achieved by this algorithm, a further statistical test is crucially required. Therefore, the next subsection is devoted to the statistical demonstrations.

4.3 Statistical analysis

Generally, either parametric or nonparametric statistical tests can be used when comparing multiple algorithms. However, for safe use parametric tests, the requirements for the independence, normality and homoscedasticity should be met, which is not the case in the present study [42]. In cases, where these assumptions cannot be achieved, nonparametric tests can be used. Accordingly, nonparametric tests are employed in this study. In this regard, Friedman test is applied first to demonstrate whether at least a single pair of algorithms shows significant difference in terms of *hit values*. One should note that since bFPAnm is used only to demonstrate the efficiency of the adopted mutation procedure, the results of bFPAnm are not included in this analysis.

The results of the Friedman test and obtained average ranks, where the best performing algorithm is assigned by the *1*st rank for the related instances, are given by Table 4. As one can see from this table bFPA achieves the best average rank. According to the p-value of this test, which is adjusted for ties, the null hypothesis H_0 that is based on the equivalence of medians should be rejected for the significance level α=0.10. This shows that at least one pair of algorithms is significantly different from each other. Thus, a further analysis to find the significantly different algorithms should be conducted.

Since the aim of this study is to demonstrate the efficiency of the proposed algorithm bFPA, a 1 × N design rather than N × N, is adopted. In other words, bFPA is compared to all remaining algorithms according to the

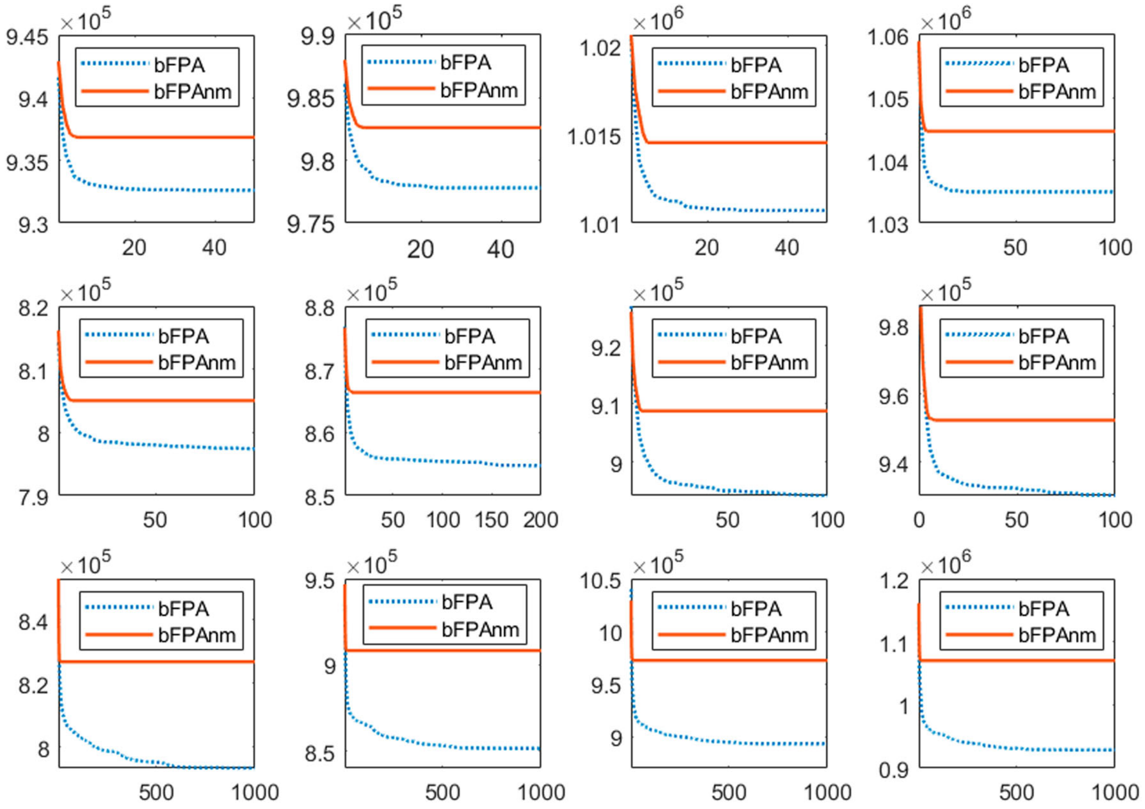


Fig. 3 Convergence plots of bFPA and bFPAnm

Table 4 Results of Friedman test

Algorithms	Overall avg. ranks
CPSO	8.625
PSO	7.125
GWO	5.875
prioGWO	4.625
learnGWO	4.583
bWSA	4.375
ABCbin	4.125
prLeGWO	3.000
bFPA	2.667
p-value (adjusted for ties)	0.000
Decision on H_0	Reject H_0

Table 5 Pairwise comparisons and post hoc test results

bFPA vs	*z*-score	Unadjusted *p*-values	Adjusted *p*-values
CPSO	5.329	0.000 (+)	0.000 (+)
PSO	3.988	0.000 (+)	0.000 (+)
GWO	2.870	0.004 (+)	0.033 (+)
prioGWO	1.752	0.079 (+)	0.639 (~)
learnGWO	1.714	0.086 (+)	0.692 (~)
bWSA	1.528	0.127 (~)	1.000 (~)
ABCbin	1.304	0.192 (~)	1.000 (~)
prLeGWO	0.298	0.766 (~)	1.000 (~)

average ranks over all instances. One crucial issue here is the possibility of losing control on the family-wise error rate while comparing the algorithms [42]. To avoid this, the Bonferroni procedure is employed as a post-hoc test. In this context, the z-scores, unadjusted *p*-values and adjusted *p*-values according to the Bonferroni procedure are given in Table 5.

According to the unadjusted *p*-values of Table 5, the proposed algorithm is found as significantly better than CPSO, PSO, GWO, prioGWO and learnGWO in terms of *hit values* for the significance level $\alpha = 0.10$. According to the adjusted *p*-values on the other hand, bFPA is found as significantly better than CPSO, PSO and GWO. In other words, although the proposed algorithm shows the most promising performance according to Tables 3 and 4, Table 5 points out that there is not enough statistical evidence to demonstrate the significant difference between bFPA and the remaining algorithms prioGWO, learnGWO, bWSA, ABCbin and prLeGWO.

4.4 Complexity analysis

Computational cost of the proposed algorithm is analyzed in terms of the worst case scenario. As can be seen from the pseudocode given by Algorithm 3, execution time of bFPA is based on the parameters *maxIt*, *popSize* and *n*. Since, the uniform-based crossover loops on the dimensions of the solution vectors of *popSize* and step-sizing function does not require additional inner loops. Thus, execution time complexity of this algorithm can be represented by $\mathcal{O}(maxIt \times popSize \times n)$ [19, 43], which also applies for the standard FPA. It further means that developed procedures do not increase the complexity of the standard FPA.

Finally, brining all things together, one can conclude that the proposed FPA modification denoted by bFPA exhibits a promising performance in solving UFLP, which may have a great variety of potential real-life applications. As reported, this algorithm does not use transfer functions that result in spatial distance. Instead, it makes of evolutionary operators such as xover and mutation. As statistically verified, this clearly contributes to the performance of the proposed algorithm. Moreover, according to the results of Table 2, it is apparent that the adopted mutation operator clearly contributes to the efficiency of bFPA. Otherwise, as demonstrated by the experimental study, potential local optima issues might emerge. Although bFPA brings about apparent advantages in solving UFLP, it has some drawbacks such as additional parameter calibration. Since mutation and xover operators require user-supplied parameters, decision makers need to carefully define these parameter values.

5 Conclusion

The present study introduces a modified flower pollination algorithm (FPA) to solve one of the well-known facility location problems, namely the uncapacitated facility location problem (UFLP). Since UFLP is a binary optimization problem, FPA, which is introduced to solve optimization problems in continuous domains, is modified for binary domains first. In this context, some operators such as crossover and mutation that are borrowed from evolutionary algorithms are adopted in the modified FPA. By using the proposed crossover operator, FPA is able to run in binary domains without employing additional auxiliary procedures such as transfer functions. Next, proposed binary bit-flip mutation is further enhanced by making use of an adaptive step-sizing procedure that introduces greater level of diversity at earlier iterations and encourages intensification toward the end of search. Thus, while premature convergence and local optima problems at earlier iterations are avoided, a more intensified search around the found promising regions is conducted.

All available benchmarking data is solved by the proposed algorithm and the obtained results are compared to those of the promising and competitive algorithms' that are lately reported in the related literature. According to the experimental study, the proposed algorithm is found as a promising algorithm. Moreover, appropriate statistical tests are conducted to demonstrate whether significant improvements are achieved. Results of the statistical tests show that the proposed algorithm brings about significant improvements over some of the existing algorithms. What is more, as reported in this study, any additional computational burden is not required by the proposed FPA modification. Nevertheless, one of the drawbacks of the proposed algorithm is that it makes use of two additional parameters, which needs to be calibrated in the proposed step-sizing function. Thus, it brings about an additional effort in parameter calibration stage.

The simplicity and efficiency of FPA deserves a further research. Since the proposed modification runs on binary populations, evaluating the population diversity by using hamming distance metric is quite practicable. In this regard, analyzing the effects of avoiding loss of diversity procedures along with the employed procedures in various binary optimization problems is scheduled as a future work. Moreover, the proposed algorithm in the present study can easily be extended to various binary optimization problems, of which the solution vectors can be represented as binary strings. Therefore, one can conclude that the proposed binary FPA and the proposed mutation operator can intrinsically be adapted to various binary optimization problems such as capacitated facility location problem, 0–1 knapsack problem, feature selection, assignment problems and their numerous extensions. Nevertheless, one should note that parameter calibration for these algorithms should be precisely carried out.

Funding Open access funding provided by the Scientific and Technological Research Council of Türkiye (TÜBİTAK). No funds, grants, or other support was received.

Data availability All data generated or analyzed during this study are included in this hyper link: http://people.brunel.ac.uk/mastjjb/jeb/info.html

Declarations

Conflict of interest The authors declare they have no financial interests. The authors have no relevant financial or non-financial interests to disclose.

Human participants and/or animals None.

Informed consent None.

References

1. Yang XS (2012) Flower pollination algorithm for global optimization. In: Durand-Lose J, Jonoska N (eds) Lecture notes in computer science, vol 7445, Springer, Berlin, Heidelberg, pp 240–249 https://doi.org/10.1007/978-3-642-32894-7_27
2. Cournuejols G, Nemhauser GL, Wolsey LA (1990) The uncapacitated facility location problem, discrete location theory. Lect Note Artif Int 1865:119–171
3. Balinski ML (1965) Integer programming: methods, uses, computations. Manage Sci 12(3):253–313. https://doi.org/10.1287/mnsc.12.3.253
4. Efroymson M, Ray TL (1966) A branch-bound algorithm for plant location. Oper Res 14(3):361–368. https://doi.org/10.1287/opre.14.3.361
5. Khumawala BM (1972) An efficient branch and bound algorithm for the warehouse location problem. Manage Sci 18(12):B-718. https://doi.org/10.1287/mnsc.18.12.B718
6. Erlenkotter D (1978) A dual-based procedure for uncapacitated facility location. Oper Res 26(6):992–1009. https://doi.org/10.1287/opre.26.6.992
7. Korkel M (1989) On the exact solution of large-scale simple plant location problems. Eur J Oper Res 39(2):157–173. https://doi.org/10.1016/0377-2217(89)90189-6
8. Johnson DS, Garey MR (1979) *Computers and intractability: A guide to the theory of NP-completeness*. WH Freeman.
9. Al-Sultan KS, Al-Fawzan MA (1999) A tabu search approach to the uncapacitated facility location problem. Ann Oper Res 86:91–103. https://doi.org/10.1023/A:1018956213524
10. Ghosh D (2003) Neighborhood search heuristics for the uncapacitated facility location problem. Eur J Oper Res 150(1):150–162. https://doi.org/10.1016/S0377-2217(02)00504-0
11. Sevkli M, Guner AR (2006) A continuous particle swarm optimization algorithm for uncapacitated facility location problem. In *International workshop on ant colony optimization and swarm intelligence*, Springer, Berlin, Heidelberg, pp 316–323. https://doi.org/10.1007/11839088_28
12. Wang D, Wu CH, Ip A, Wang D, Yan Y (2008) Parallel multi-population particle swarm optimization algorithm for the uncapacitated facility location problem using openMP. In 2008 IEEE congress on evolutionary computation pp 1214–1218. https://doi.org/10.1109/CEC.2008.4630951
13. Guner AR, Sevkli M (2008) A discrete particle swarm optimization algorithm for uncapacitated facility location problem. J Artif Evolut Appl. https://doi.org/10.1155/2008/861512
14. Kiran MS (2015) The continuous artificial bee colony algorithm for binary optimization. Appl Soft Comput 33:15–23. https://doi.org/10.1016/j.asoc.2015.04.007
15. Montoya-Torres JR, Aponte A, Rosas P (2011) Applying GRASP to solve the multi-item three-echelon uncapacitated facility location problem. J Oper ResSoc 62(2):397–406
16. Tsuya K, Takaya M, Yamamura A (2017) Application of the firefly algorithm to the uncapacitated facility location problem. J Intell Fuzzy Syst 32(4):3201–3208. https://doi.org/10.3233/JIFS-169263
17. de Armas J, Juan AA, Marquès JM, Pedroso JP (2017) Solving the deterministic and stochastic uncapacitated facility location problem: from a heuristic to a simheuristic. J Oper Res Soc 68(10):1161–1176. https://doi.org/10.1057/s41274-016-0155-6
18. Baykasoglu A, Ozsoydan FB, Senol ME (2018). Weighted superposition attraction algorithm for binary optimization problems. Oper Res: 1–27. https://doi.org/10.1007/s12351-018-0427-9
19. Ozsoydan FB (2019) Artificial search agents with cognitive intelligence for binary optimization problems. Comput Ind Eng 136:18–30. https://doi.org/10.1016/j.cie.2019.07.007
20. Ozsoydan FB (2019) Effects of dominant wolves in grey wolf optimization algorithm. Appl Soft Comput 83:105658. https://doi.org/10.1016/j.asoc.2019.105658
21. Golcuk İ, Ozsoydan FB (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
22. Sonuç E (2021) Binary crow search algorithm for the uncapacitated facility location problem. Neural Comput Appl 33(21):14669–14685. https://doi.org/10.1007/s00521-021-06107-2
23. Zhang F, He Y, Ouyang H, Li W (2023) A fast and efficient discrete evolutionary algorithm for the uncapacitated facility location problem. Expert Syst Appl 213:118978. https://doi.org/10.1016/j.eswa.2022.118978
24. Alidaee B, Wang H (2022) Uncapacitated (Facility) location problem: a hybrid genetic-tabu search approach. IFAC-PapersOnLine 55(10):1619–1624. https://doi.org/10.1016/j.ifacol.2022.09.622
25. Kaya E (2022) BinGSO: galactic swarm optimization powered by binary artificial algae algorithm for solving uncapacitated facility location problems. Neural Comput Appl 34(13):11063–11082. https://doi.org/10.1007/s00521-022-07058-y
26. Zhou J, Wang X, Zhang L, Zhou X, Jing S, & Liang G (2022). An improved genetic algorithm for the uncapacitated facility location problem and applications in oil and gas fields. In Journal of physics: conference series (Vol. 2224, No. 1, p. 012134). IOP Publishing. https://doi.org/10.1088/1742-6596/2224/1/012134
27. Soltanpour A, Alizadeh B, Baroughi F (2023) Efficient algorithms for uncapacitated facility location problem on uncertain environments. Iran J Oper Res 14(1):118–132
28. Jiang N, Zhang H (2023) Improved adaptive differential evolution algorithm for the un-capacitated facility location problem. Open J Appl Sci 13(5):685–695. https://doi.org/10.4236/ojapps.2023.135054
29. Baş E, Yildizdan G (2024) A new binary arithmetic optimization algorithm for uncapacitated facility location problem. Neural Comput Appl 36(8):4151–4177. https://doi.org/10.1007/s00521-023-09261-x
30. Aslan M, Pavone M (2024) MBVS: a modified binary vortex search algorithm for solving uncapacitated facility location problem. Neural Comput Appl 36(5):2573–2595. https://doi.org/10.1007/s00521-023-09190-9
31. Dubey HM, Pandit M, Panigrahi BK (2015) A biologically inspired modified flower pollination algorithm for solving

economic dispatch problems in modern power systems. Cogn Comput 7(5):594–608. https://doi.org/10.1007/s12559-015-9324-1

32. Rodrigues D, Yang XS, de Souza AN, Papa JP (2015) Binary flower pollination algorithm and its application to feature selection. In: Yang XS (eds) Recent advances in swarm intelligence and evolutionary computation, Studies in computational intelligence, vol 585. Springer, Cham pp 85–100. https://doi.org/10.1007/978-3-319-13826-8_5
33. Dahi ZAEM, Mezioud C, Draa A (2016) On the efficiency of the binary flower pollination algorithm: application on the antenna positioning problem. Appl Soft Comput 47:395–414. https://doi.org/10.1016/j.asoc.2016.05.051
34. Nabil E (2016) A modified flower pollination algorithm for global optimization. Expert Syst Appl 57:192–203. https://doi.org/10.1016/j.eswa.2016.03.047
35. Abdel-Baset M, Hezam IM (2015) An effective hybrid flower pollination and genetic algorithm for constrained optimization problems. Adv Eng Technol Appl Int J 4:27–27
36. Draa A (2015) On the performances of the flower pollination algorithm-Qualitative and quantitative analyses. Appl Soft Comput 34:349–371. https://doi.org/10.1016/j.asoc.2015.05.015
37. Zhou Y, Wang R, Zhao C, Luo Q, Metwally MA (2019) Discrete greedy flower pollination algorithm for spherical traveling salesman problem. Neural Comput Appl 31(7):2155–2170. https://doi.org/10.1007/s00521-017-3176-4
38. Ozsoydan FB, Baykasoglu A (2019) Analysing the effects of various switching probability characteristics in flower pollination algorithm for solving unconstrained function minimization problems. Neural Comput Appl 31(11):7805–7819. https://doi.org/10.1007/s00521-018-3602-2
39. Song H, Bei J, Zhang H, Wang J, Zhang P (2024) Hybrid algorithm of differential evolution and flower pollination for global optimization problems. Expert Syst Appl 237:121402. https://doi.org/10.1016/j.eswa.2023.121402
40. Mellal MA, Khitous M, Zemmouri M (2023) Combined heat and power economic dispatch problem with binary method using flower pollination algorithm and differential evolution. Electrc Eng 105(4):2161–2168. https://doi.org/10.1007/s00202-023-01801-x
41. Feng L, Zhou Y, Luo Q (2024) Binary hybrid artificial hummingbird with flower pollination algorithm for feature selection in parkinson’s disease diagnosis. J Bionic Eng. https://doi.org/10.1007/s42235-023-00478-z
42. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evolut Comput 1(1):3–18. https://doi.org/10.1016/j.swevo.2011.02.002
43. Al-Betar MA, Awadallah MA, Faris H, Aljarah I, Hammouri AI (2018) Natural selection methods for grey wolf optimizer. Expert Syst Appl 113:481–498. https://doi.org/10.1016/j.eswa.2018.07.022