Expert Systems With Applications 213 (2023) 118978

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A fast and efficient discrete evolutionary algorithm for the uncapacitated facility location problem

Fazhan Zhang [a], Yichao He [a,*], Haibin Ouyang [b], Wenben Li [a]

[a] School of Information Engineering, Hebei GEO University, Shijiazhuang, 050031, China
[b] School of Mechanical and Electric Engineering, Guangzhou University, Guangzhou, 510006, China

ARTICLE INFO

Keywords:
Evolutionary algorithm
Facility location problem
Optimization algorithm
One direction mutation operator
Redundant checking strategy

ABSTRACT

In order to solve the uncapacitated facility location problem (UFLP) quickly and effectively, an enhanced group theory-based optimization algorithm (EGTOA) is proposed in this paper. Firstly, a new local search operator, One Direction Mutation Operator, is proposed, which is suitable for solving UFLP. Secondly, a Redundant Checking Strategy is presented to further optimize the quality of feasible solutions. To verify the performance of EGTOA, 15 benchmark instances of UFLP is selected in OR-Library, the comparison results with the 16 existing algorithms show that the solution obtained by EGTOA is better than other algorithms, moreover its speed is much faster than state-of-the-art algorithms. These demonstrates that EGTOA is a fast and effective algorithm for solving UFLP.

1. Introduction

The uncapacitated facility location problem (UFLP) (Kuehn & Hamburger, 1963) is a combinatorial optimization problem firstly proposed by Kuehn and Hamburger in 1963, which is one of the most important NP-hard problems in location theory (Daskin et al., 2003). The main goal of UFLP is try to find an undetermined number of facilities to minimize the sum of constant setup and serving costs of customers. At present, UFLP has important application value in the fields of resource allocation, capital budget, logistics and transportation, computer network and computer vision (Armas & Juan, 2018; Azad et al., 2013; Cura, 2010; Lazic et al., 2010, 2009), as well as the location of infrastructure construction of schools, factories, warehouses, fire stations and hospitals and so on (Bumb, 2002; Manne, 1964; Stollsteimer & F., 1963).

At present, many approaches for solving UFLP have been proposed. In terms of deterministic algorithms, Efroymson and Ray (1966) were the first men who solved UFLP with branch-and-bound method. Their basic contribution is that the problem is formulated as an integer program in order to effectively optimize the related continuous problems. Khumawala (Akinc & Khumawala, 1977) improved the branch-and-bound algorithm proposed by Efroymson and Ray, and introduced an improved method of solving the linear programming at the nodes which substantially reduces the computations. Galvvao and Raggi (1989) proposed a three-stage exact method for solving general 0-1 formulation of UFLP, which is composed of a primal–dual algorithm, a sub-gradient optimization to solve a Lagrangian dual and a branch-and-bound algorithm. Conn and Cornuejols (1990) presented a new method for solving the UFLP based upon the exact solution of the condensed dual via orthogonal projections. The method is flexible and can handle side constraints. Especially, the linear programming formulation can be strengthened by adding cuts when there is a duality gap.

In terms of non-deterministic algorithm, Charikar and Guha (Moses & Sudipto, 1999) proposed a simple greedy local search algorithm based on two central ideas — cost scaling and greedy improvement, which achieves an approximation ratio of 2.414+ϵ in $O(n^2/\epsilon)$ time. Li (2013) presented an 1.488-approximation algorithm for the metric UFLP. In the approximation algorithm, by randomly selecting the value of parameter γ proposed by Byrka and Aardal (2007), the approximation ratio of the algorithm is improved to 1.488. Kratica et al. (2001) proposed a method of solving UFLP by using genetic algorithm (GA), in which the crossover operation uses a uniform cross-strategy proposed by Syswerda (1989), This is an important early literature for solving UFLP by evolutionary algorithm. Michel and Hentenryck (2004) presented a simple, yet robust and efficient tabu-search algorithm for the UFLP. It can find optimal solutions to all benchmarks very quickly and with very high frequencies. Aydin and Fogarty (2004) proposed a distributed evolutionary simulated annealing algorithm (DESA) to solve UFLP. It is a distributed algorithm that consists of a simulated annealing operator instead of reproduction and a selection operator. Kiran and GuNDuZ (2014) proposed XOR-based modification for the

* Corresponding author.
E-mail addresses: 43103098@qq.com (F. Zhang), heyichao@hgu.edu.cn (Y. He), oyhb1987@163.com (H. Ouyang), 865975991@qq.com (W. Li).

https://doi.org/10.1016/j.eswa.2022.118978
Received 3 March 2021; Received in revised form 3 October 2022; Accepted 3 October 2022
Available online 11 October 2022
0957-4174/

solution-updating equation of the ABC algorithm (binABC) in order to solve UFLP. After that, Kiran and Servet (2015) proposed a new binary artificial bee colony algorithm (ABC_{bin}) and used it to give a method of solving UFLP. According to the calculation results, its advantages in solution quality, robustness and simplicity are verified. (Husseinzadeh Kashan et al., 2013) replaced arithmetic subtraction operators with dissimilarity measures of binary structures and proposed a novel binary DE algorithm (DisDE) for solving UFLP. Takaya (Tsuya et al., 2017) and Atta (Atta et al., 2018) proposed approaches to solve UFLP by using firefly algorithm (FA) and monkey algorithm (MA), respectively. But they did not give the calculation results of large-scale UFLP instances. Korkmaz et al. (2017) proposed a feasible method of solving UFLP by using the artificial algae algorithm (AAA). Subsequently, they made further improvements and proposed a more effective algorithm IAAA (Korkmaz & Kiran, 2018) for solving UFLP. Experimental results show that IAAA is one of the most competitive algorithms at present. Emine and Erkan (2020) proposed a binary social spider algorithm BinSSA for uncapacitated facility location problem. Two new methods (similarity measure and logic gate) are used in candidate solution production schema for increasing the exploration and exploitation capacity of BinSSA. Taymaz et al. (2022) proposed a binary battle royale optimizer algorithm (BinBRO). In BinBRO, a crossover operator is applied on the loser solution, if it has not yet reached the maximum number of losses. This crossover operator is from GA and has used in three different ways: (i) single-point crossover, (ii) two-point crossover, and (iii) uniform crossover.

Obviously, the deterministic algorithms can obtain the optimal solution, but they are not polynomial time complexity and suitable only for solving small-scale UFLP instances. Although non-deterministic algorithms cannot guarantee to obtain the optimal solution, they are polynomial time complexity and can obtain an approximate optimal solution of UFLP problem rapidly. Especially the evolutionary algorithms (EAs), they are not only fast, but also can obtain the optimal solution or approximate optimal solution of UFLP problem. At present, EAs have become the main method to quickly solve UFLP problem in practical application.

In recent years, in addition to classical EAs such as genetic algorithm (GA) (Damgacioglu et al., 2015; Deb, 2000; Mitchell, 1996), particle swarm optimization (PSO) (Ghaderi et al., 2012; Kennedy & Eberhart, 1995; Sevkli & Guner, 2006) and differential evolution (DE) (Meng et al., 2019; Meng & Yang, 2021), scholars have also proposed many new EAs by simulating the behavior of biological communities in nature or using mathematical, physical and other theoretical systems, such as grey wolf optimization (GWO) (Emarya et al., 2016; Mirjalili et al., 2014), fireworks algorithm (FWA) (Tan & Zhu, 2010), Monkey King Evolution(MKE) (Meng & Pan, 2016), pigeon-inspired optimization (PIO) (Duan & Fei, 2017), whale optimization algorithm (WOA) (Mirjalili & Lewis, 2016), group theory-based optimization algorithm (GTOA) (He & Wang, 2018), lion optimization algorithm (LOA) (Yazdani & Jolai, 2016), etc. At present, EAs have successfully applied to numerical optimization, combinatorial optimization, economic scheduling, engineering optimization, machine learning and so on (Cai et al., 2005; Duan & Fei, 2017; Emarya et al., 2016; Meng et al., 2020, 2019, 2016). GTOA is a discrete evolutionary algorithm proposed by He and Wang (2018) based on group theory operation in 2018. It has the advantages of fewer parameters, strong stability and fast convergence, and successfully used in solving knapsack problems such as the set-union knapsack problem, the discounted {0-1} knapsack problem, and the bounded knapsack problem (He & Wang, 2018). In this paper, we study how to use GTOA to solve UFLP rapidly, and propose an enhanced group theory-based optimization algorithm EGTOA for solving UFLP.

The main contributions of this paper are as follows:

(1) A local search operator, One Direction Mutation Operator (ODMO), is proposed, which is suitable for solving UFLP by evolutionary algorithm.

(2) A redundant checking strategy (RCS) is proposed, which can improve the quality of the feasible solutions of UFLP obtained by evolutionary algorithm.

(3) An enhanced group-theory optimization algorithm EGTOA is proposed based on ODMO and RCS, which is more competitive than the state-of-the art algorithms for solving UFLP in computed result and speed.

The rest of the paper is organized as follows: Section 2 introduces the definition and the mathematical model of UFLP. Section 3 describes the principle of GTOA and gives its pseudo-code. In Section 4, a new local search operator ODMO is proposed to enhance the ability of solving UFLP. And a redundancy checking strategy RCS is proposed to improve the quality of feasible solutions. Then, we present an enhanced group theory-based optimization algorithm (EGTOA) based on GTOA. In Section 5, we first use the Kruskal–Wallis test to determine the best value of the parameters in EGTOA. Then, the comparison with the 16 existing algorithms shows that not only the performance of EGTOA is best among all algorithms, but also its speed is much faster than other algorithms. Section 6 summarizes this work.

2. Definition and model of UFLP

The definition of UFLP is generally described as: giving the set $K = \{k_1, k_2, \dots, k_m\}$ of customers, where m is the number of customers and k_i is the ith customer. Giving the set $S = \{s_1, s_2, \dots, s_n\}$ of potential facilities that can be opened, where n is the number of facilities and s_j represents the jth facility. Giving an $m \times n$ matrix $D = [d_{ij}]_{m\times n}$, where d_{ij} represents the service cost when the ith customer receives the service from the jth facility. $G = \{g_1, g_2, \dots, g_n\}$ is the set of fixed open cost of facilities, where g_j represents the opening cost required by the opening of the jth facility. It is worthy to note that the demand of any customer is fulfill by only one facility. The goal of UFLP is to find a set of open facilities and a reasonable allocation scheme between facilities and customers, so that the sum of total service costs and total open facility costs is minimized.

We define two binary decision variables y_{ij} and x_j as follows:

$$y_{ij} = \begin{cases} 1, & if\ customer\ i\ gets\ service\ from\ facility\ j; \\ 0, & otherwise. \end{cases} \tag{1}$$

$$x_j = \begin{cases} 1, & if\ facility\ j\ is\ opened; \\ 0, & otherwise. \end{cases} \tag{2}$$

According to the above definition, the mathematical model of UFLP is described as follows:

$$minimize \quad \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}y_{ij} + \sum_{j=1}^{n} g_i x_j \tag{3}$$

$$s.t. \quad \sum_{j=1}^{n} y_{ij} = 1,\ i = 1,2,\dots,m \tag{4}$$

$$y_{ij} \le x_j,\ i = 1,2,\dots,m,\ and\ j = 1,2,\dots,n \tag{5}$$

$$y_{ij}, x_j \in \{0,1\},\ i = 1,2,\dots,m,\ and\ j = 1,2,\dots,n \tag{6}$$

The first term in the objective function (3) denotes the total service cost, and the second term denotes the total opening cost of the opened facilities. Constraint (4) ensures that every customer is served by exactly one facility. Constraint (5) ensures that a customer can be served from a facility only if a facility is opened. Constraint (6) defines the decision variables in the binary structure.

3. GTOA algorithm

GTOA (He & Wang, 2018) is a discrete evolutionary algorithm recently proposed by He and Wang. The key parts of GTOA are that the feasible solution of the problem is considered as an element of the direct product of groups and that the evolution process is implemented by multiplication and inverse operations of the direct product of groups. GTOA has the advantages of fewer parameters, simple structure, strong stability, fast convergence speed and high convergence accuracy. It has been successfully applied to solve a variety of knapsack problems (He & Wang, 2018). In this section, we introduce the principle of GTOA and give its pseudo-code.

Let $Z_q = \{0, 1, \dots, q\}$, q is a positive integer and $q \geq 1$. We define a binary operation $\oplus$ on Z_q as follows:

$$\forall a, b \in Z_q, a \oplus b = (a + b)(mod\ q) \tag{7}$$

Where + is a common addition operator, $x(mod\ q)$ denotes the remainder when x is divided by q. It is clear that $(Z_q, \oplus)$ is a modulo q integer additive group, and its identity is 0. We use $-a$ to denote the inverse a^{-1} of a in Z_q. Then, $-0 = 0$, and, if $a \neq 0$ then $-a = a^{-1} = q - a$.

Let $Z_{q_i} = \{0, 1, \dots, q_i\}$, where q_i is a positive integer and $q_i \geq 1$, $i = 1, 2, \dots, n$. $Z_{q_1} \times Z_{q_2} \times \cdots \times Z_{q_n}$ is a direct product of groups which denoted by $Z[q_1, q_2, \dots, q_n]$. It is easy to see that $(0, 0, \dots, 0)$ is the identity. The inverse of $(a_1, a_2, \dots, a_n)$ is $-(a_1, a_2, \dots, a_n)$, and $-(a_1, a_2, \dots, a_n) = (-a_1, -a_2, \dots, -a_n)$, where $-a_i$ is the inverse of a_i, $a_i \in Z_{q_i}$. $\forall (a_1, a_2, \dots, a_n)$, $(b_1, b_2, \dots, b_n) \in Z[q_1, q_2, \dots, q_n]$, we have

$$(a_1, a_2, \dots, a_n) \oplus (b_1, b_2, \dots, b_n) = (a_1 \oplus b_1, a_2 \oplus b_2, \dots, a_n \oplus b_n) \tag{8}$$

Suppose that $U = (u_1, u_2, \dots, u_n)$, $V = (v_1, v_2, \dots, v_n)$, $W = (w_1, \dots, w_n)$ are three different elements randomly selected from $Z[q_1, q_2, \dots, q_n]$. A new element $X = (x_1, x_2, \dots, x_n) \in Z[q_1, q_2, \dots, q_n]$ is generated by group operations work on U, V, W according to the following Eq. (9):

$$X = U \oplus (F(V \oplus (-W))) \tag{9}$$

where $x_j = u_j \oplus [f_j(v_j \oplus (q_j + 1 - w_j))]$, $j = 1, 2, \dots, n$. $F = (f_1, f_2, \dots, f_n)$ is an n-dimensional random vector in $\{-1, 0, 1\}^n$, which is called the combinatorial factor vector (He & Wang, 2018). Eq. (9) is called random linear combination operator (RLCO). It is a global random search operator of GTOA.

In order to balance the exploration and exploitation. GTOA proposes two kinds of local search operators. For the combinatorial optimization problem with solution space $Z[q_1, q_2, \dots, q_n]$ and $q_j \geq 1$ ($j = 1, 2, \dots, n$), the local search operator is called Inversion and Random Mutation Operator (IRMO) (He & Wang, 2018); For the combinatorial optimization problem with solution space $Z[2, 2, \dots, 2] = \{0, 1\}^n$, the local search operator is called Switch Mutation Operator (SMO) (He & Wang, 2018). Since UFLP is a binary optimization problem, only SMO can be used. Therefore, SMO is briefly described below: Let $X = (x_1, x_2, \dots, x_n) \in \{0, 1\}^n$ and P_r be the mutation probability of SMO. For each x_j ($j = 1, 2, \dots, n$), if ($rand \leq P_r$) then $x_j \leftarrow x_j^{-1}$; otherwise x_j remains unchanged, where $rand$ is a random number in (0,1), and P_r satisfies $0 \leq P_r \leq 0.5$. It is easy to see that the time complexity of SMO is $O(n)$.

Let $P(t) = \{X_c(t) | 1 \leq c \leq PS\}$ be the tth generation population of GTOA, where $X_c(t) = (x_{c1}(t), x_{c2}(t), \dots, x_{cn}(t)) \in \{0, 1\}^n$ is the cth individual. PS is the size of the population, t is an integer and $t \geq 0$. Let $B = (b_1, b_2, \dots, b_n) \in \{0, 1\}^n$ be the best individual at present, MI be the number of iterations, and $S = (s_1, s_2, \dots, s_n) \in \{0, 1\}^n$ be an n-dimensional vector. $X_{P1}(t)$, $X_{P2}(t)$ and $X_{P3}(t)$ are three different individuals randomly selected from the tth generation population $P(t)$. For the maximum optimization problem $h(X)$, $X \in \{0, 1\}^n$, the pseudo-code of GTOA (He & Wang, 2018) is described in Algorithm 1:

Algorithm 1. GTOA

```
Input:  An instance of h(X), Parameters PS, MI and P_r;
Output: An approximate solution (or optimal solution) B and f(B).
    Generate initial population P(0) = {X_c(0) | 1 ≤ c ≤ PS}
    randomly;
    Compute h(X_c(0)), 1 ≤ c ≤ PS; Determine B; t ← 0;
    while (t ≤ MI) do
      for c ← 1 to PS do
        S ← X_p1(t) ⊕ (F(X_p2(t) ⊕ (−X_p3(t))));
        S ← SMO(S, P_r);   //call Switch Mutation Operator SMO
        if h(S) > h(X_c(t)) then X_c(t+1) ← S; else
        X_c(t+1) ← X_c(t);
      end for
      Generate P(t+1); Determine B; t ← t+1;
    end while
    return(B, h(B))
```

According to literature (He & Wang, 2018), when the time complexity of computing $h(X)$ is $O(n)$, MI and PS are linear function of n, the time complexity of GTOA is $O(n^3)$.

Table 1
The scale and the optimal value of 15 benchmark instances.

Scale	Instance	$m \times n$	Optimal Value
Small-scale (Cap71∼ Cap74)	Cap71	16 × 50	932615.75
	Cap72	16 × 50	977799.40
	Cap73	16 × 50	1010641.45
	Cap74	16 × 50	1034976.98
Medium-scale (Cap101∼ Cap104)	Cap101	25 × 50	796648.44
	Cap102	25 × 50	854704.20
	Cap103	25 × 50	893782.11
	Cap104	25 × 50	928941.75
Medium-scale (Cap131∼ Cap134)	Cap131	50 × 50	793439.56
	Cap132	50 × 50	851495.33
	Cap133	50 × 50	893076.71
	Cap134	50 × 50	928941.75
Large-scale (CapA∼ CapC)	CapA	100 × 1000	17156454.48
	CapB	100 × 1000	12979071.58
	CapC	100 × 1000	11505594.33

Table 2
Kruskal–Wallis test results for 4 instances.

Instance	$m \times n$	P	Res
Cap71	16 × 50	1.0000	N
Cap101	25 × 50	0.9757	N
Cap131	50 × 50	0.0402	Y
CapB	100 × 1000	3.55E−22	Y

Table 3
$\#OPT$ and $Time$ in 30 times independent calculation for 4 instances.

Instance	$Metric$	0.05	0.1	0.2	0.4	0.6
Cap71	$\#OPT$	**30**	**30**	**30**	**30**	**30**
	$Time$	**0.316**	**0.323**	**0.326**	**0.347**	**0.356**
Cap101	$\#OPT$	28	28	**30**	**30**	**30**
	$Time$	0.485	0.477	**0.490**	**0.508**	**0.522**
Cap131	$\#OPT$	18	28	**30**	**30**	24
	$Time$	0.853	0.869	**0.890**	**0.967**	1.007
CapB	$\#OPT$	10	16	**24**	1	1
	$Time$	20.837	20.548	**21.730**	20.876	21.282

4. Enhanced group theory-based optimization algorithm

In order to fast solve UFLP by GTOA, Firstly, a new local search operator ODMO is proposed to enhance the optimization ability of GTOA. In addition, in order to further optimize the quality of feasible solutions, redundant checking strategy RCS is used to modify open

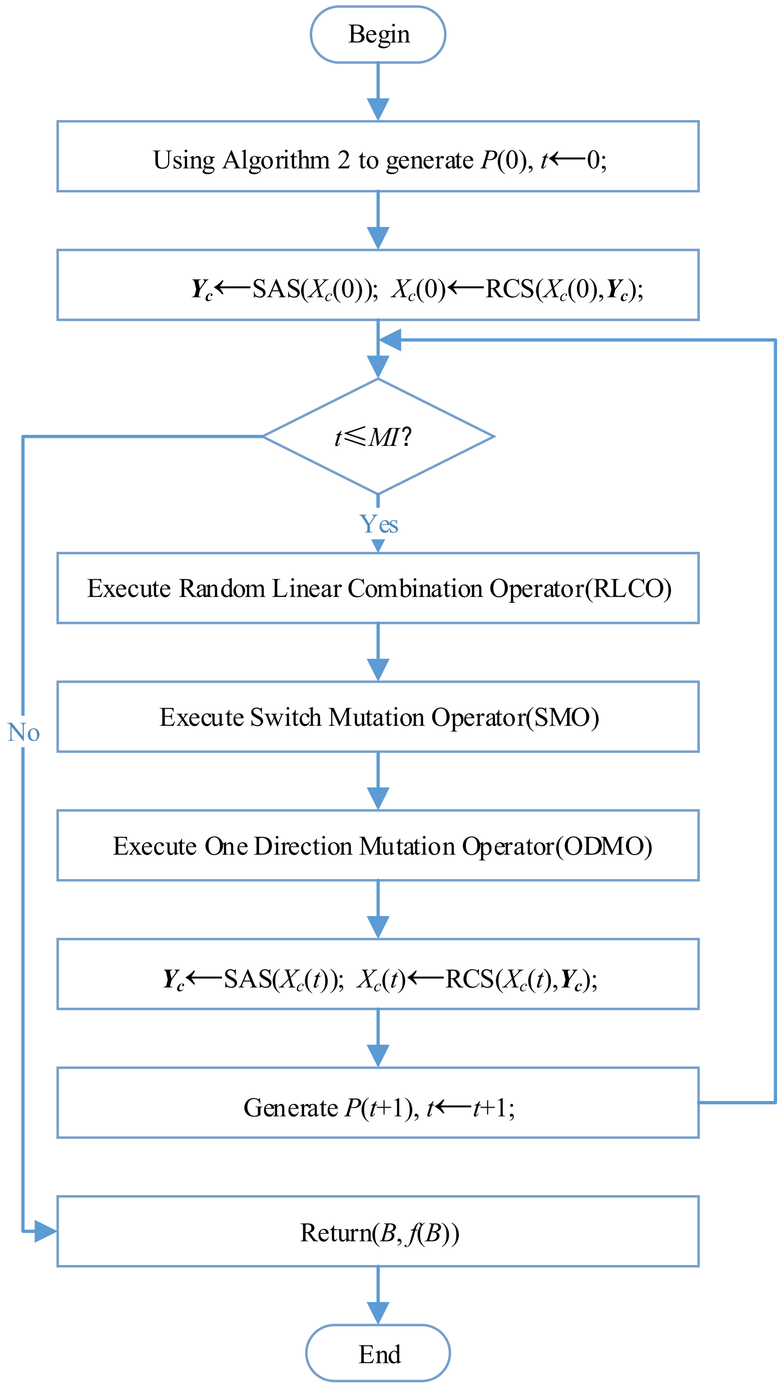


Fig. 1. Flowchart of solving UFLP by EGTOA.

facilities after determining the customers served by facilities. Based on the above improvements, an enhanced group theory-based optimization algorithm EGTOA is proposed for solving UFLP quickly. Next, the principles and implementation of EGTOA are introduced step by step.

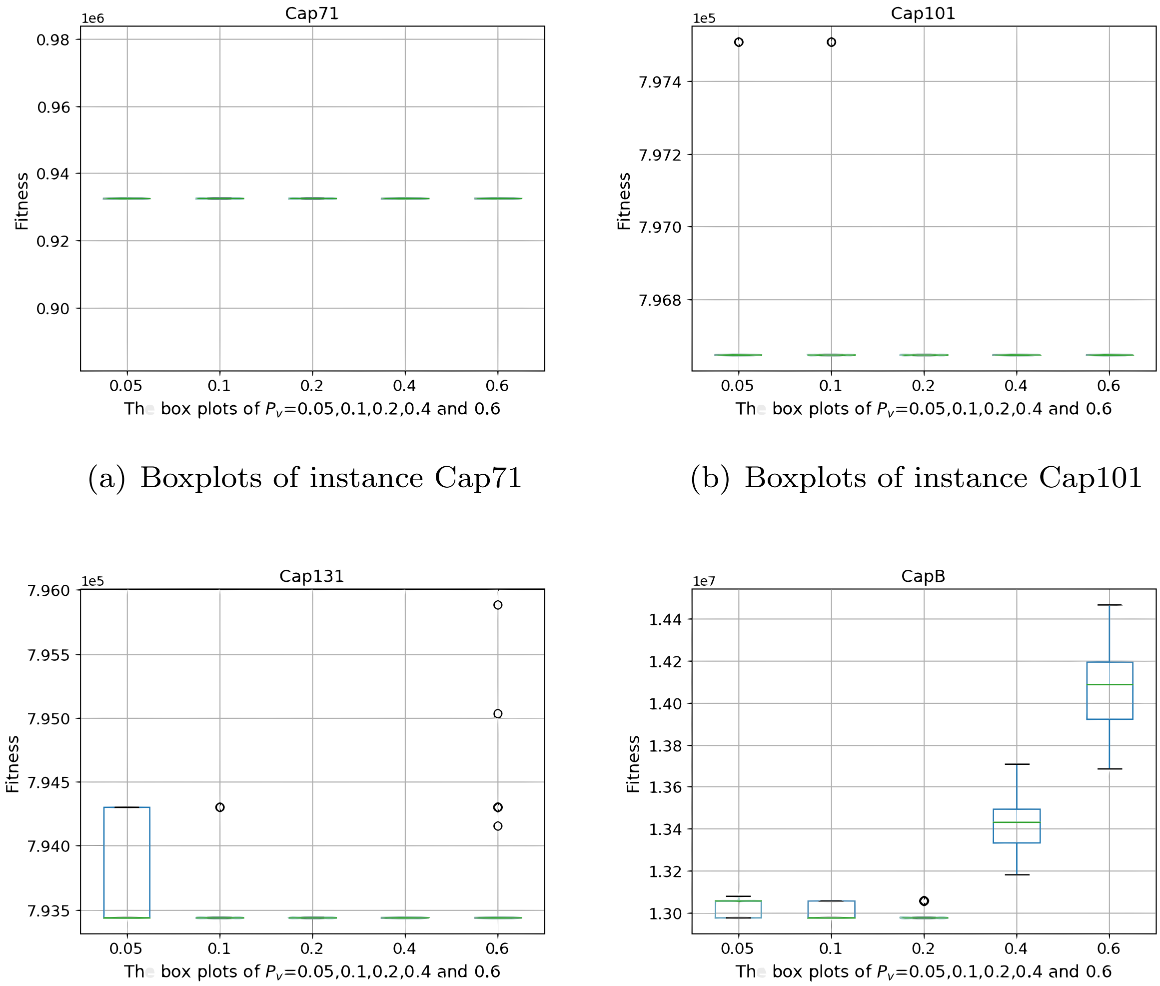
4.1. Population initialization and determining service scheme

In EGTOA, an individual $X = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$, where $x_j \in \{0,1\}$ and $j = 1, 2, \ldots, n$, expresses a solution of the UFLP. If facility j is open, then $x_j = 1$; otherwise $x_j = 0$.

EGTOA first randomly generates an initial population $P(0) = \{X_c(0) | 1 \le c \le PS\}$, where $X_c(0) = (x_{c1}(0), x_{c2}(0), \ldots, x_{cn}(0)) \in \{0,1\}^n$ and PS is the size of the population. The pseudo-code of initializing population is shown in Algorithm 2.

Table 4
Parameter settings of BPSO, GBPSO, HBDE, GHBDE, GTOA, and EGTOA.

Algorithm	BPSO and GBPSO	HBDE and GHBDE	GTOA and EGTOA
Parameters	$PS = 40$	$PS = 40$	$PS = 40$
	$MI = 2000$	$MI = 2000$	$MI = 2000$
	$W = 1.0$	$CR = 0.3$	$P_r = 0.02$
	$C_1 = C_2 = 2.0$	$FS = 0.8$	–
	$[-V, V] = [-5.0, 5.0]^n$	$[-V, V] = [-5.0, 5.0]^n$	–
	$P_v = 0.2$ (GBPSO)	$P_v = 0.1$ (GHBDE)	$P_v = 0.2$(EGTOA)



(a) Boxplots of instance Cap71 (b) Boxplots of instance Cap101

(c) Boxplots of instance Cap131 (d) Boxplots of instance CapB

Fig. 2. Boxplots of instance Cap71, Cap101, Cap131 and CapB.

Algorithm 2. *Population Initialization*

Input: Parameters PS and n (number of facilities);
Output: $P(0) = \{X_c(0) | 1 \leq c \leq PS\}$.
for $c \leftarrow 1$ to PS do
  for $j \leftarrow 1$ to n do
    $if(rand \leq 0.5)$ then $x_{cj} \leftarrow 1$; $else\ x_{cj} \leftarrow 0$;
  end for
end for
return($P(0)$).

It is easy to see that the time complexity of Algorithm 2 is $O(PS * n)$.

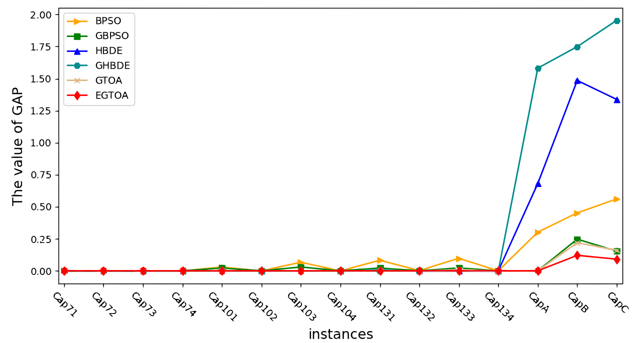
Let $\boldsymbol{Y}_c = [y_{ij}]_{m \times n}$ be an $m \times n$ matrix. After determining the open facility by $X_c = (x_{c1}, x_{c2}, \ldots, x_{cn})$, $c = 1, 2, \ldots, PS$, the method in literature (Husseinzadeh Kashan et al., 2013) is used to determine matrix $\boldsymbol{Y}_c$. That is, the specific method of determining the best service scheme between the open facilities and customers based on X_c is as follows: each customer i ($i = 1, 2, \ldots, m$) is fulfilled by the facility opened at location k whose service cost d_{ik} is minimum ($k = argmin_{j \in S}\{d_{ij}\}, S = \{j \mid x_{cj} = 1 \wedge x_{cj} \in X_c\}$). Then, $y_{ik} = 1$ and $y_{ij} = 0$, $\forall j = 1, 2, \ldots, n$, $j \neq k$. Therefore, this is the location decisions that should be done optimally.

4.2. One direction mutation operator

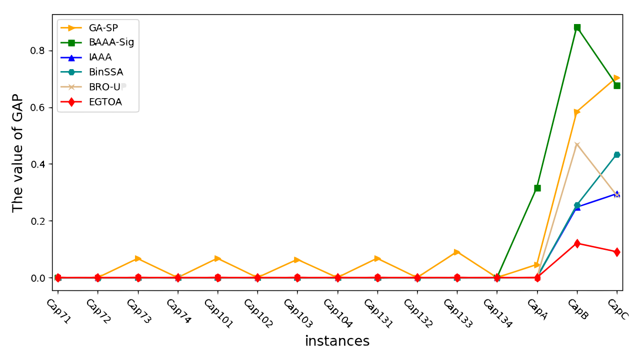
Although the method of determining $\boldsymbol{Y}_c$ by X_c in Section 4.1 is the most effective method at present, because it is based on greedy strategy, there is a defect that cannot be ignored. In order to better illustrate this point, let $\boldsymbol{Y}' = (y'_{ij})_{m \times n}$ be determined first by using $k = argmin_{j=1,\ldots,n}\{d_{ij}\}$, $y'_{ik} = 1$ and $y'_{ij} = 0$, $\forall j = 1, 2, \ldots, n$, $j \neq k$, $i = 1, 2, \ldots, m$. After that $X' = (x'_1, x'_2, \ldots, x'_n) \in \{0, 1\}^n$ is determined according to $x'_j = 1$ if and only if $\sum_{i=1}^{m} y'_{ij} \geq 1, j = 1, 2, \ldots, n$. Thus, a feasible solution $\langle X', \boldsymbol{Y}' \rangle$ of UFLP is determined. Obviously, this is a typical approximation algorithm (Du et al., 2011), and its computational effect has been proved to be very poor.

Let $S' = \{j \mid x'_j = 1 \wedge x'_j \in X'\}$. It is not difficult to see that if there are many components with a value of 1 in X_c, the situation of $S' \subseteq S$ is very easy to occur, resulting in $\boldsymbol{Y}_c = \boldsymbol{Y}'$. Even if there is no $S' \subseteq S$, but the value of $\mid S' \bigcap S \mid$ is very large, which also makes $\boldsymbol{Y}_c$ very close to $\boldsymbol{Y}'$. These two situations will greatly reduce the quality of feasible solutions of UFLP. Furthermore, it can be seen from the mathematical model of UFLP that if the number of open facilities is larger (i.e. $\mid S \mid$ is larger), the value of objective function (see Eq.(3)) is more affected by $\sum_{j=1}^{n} g_i x_{cj}$ and the total cost is higher. Therefore, there is such defect in the above method, that is, the more components with a value of 1 in X_c, the easier it is to reduce the quality of the feasible solution, resulting in a decline in the performance of the algorithm.

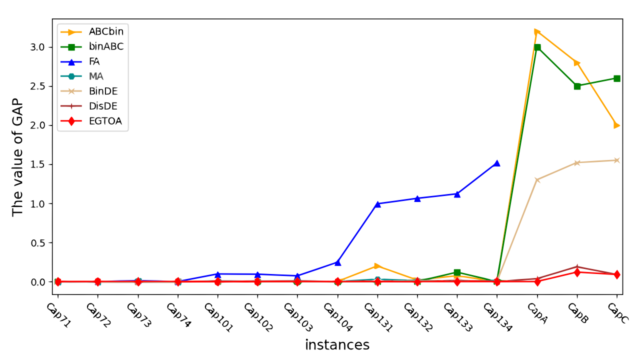
A simple way to overcome the above defects is to reduce open facilities as much as possible while meeting all customer service needs, so as to reduce the value of the objective function, ameliorate the quality of the solution, and improve the performance of the algorithm. To this end, we propose a local search operator by using the idea



(a) GAP curves for EGTOA,GPBSO, GHBDE, GTOA,BPSO,and HBDE



(b) GAP curves for EGTOA,GA-SP,BAAA-Sig, IAAA,BinSSA, and BRO-UP



(c) GAP curves for EGTOA,ABC_{bin},binABC, FA, MA, BinDE,and DisDE

Fig. 3. The GAP curves of 17 algorithms.

of randomized algorithm: For each open facility j, if the randomly generated random number $rand$ in (0, 1) does not exceed the mutation probability P_v $(0 < P_v < 1)$, then facility j is turned off. Since the local search operator based on the strategy is a one direction mutation, it is called the One Direction Mutation Operator (ODMO).

Let $X = (x_1, x_2, \dots, x_n) \in \{0, 1\}^n$ be a scheme of open facilities. The pseudo-code of ODMO is given in Algorithm 3.

Algorithm 3. *ODMO*

Input: $X = (x_1, x_2, \dots, x_n)$, mutation probability P_v;
Output:The mutated $X = (x_1, x_2, \dots, x_n)$.
for $j \leftarrow 1$ to n do
  if $(x_j = 1 \wedge rand \leq P_v)$ then $x_j \leftarrow 0$;
end for
return(X)

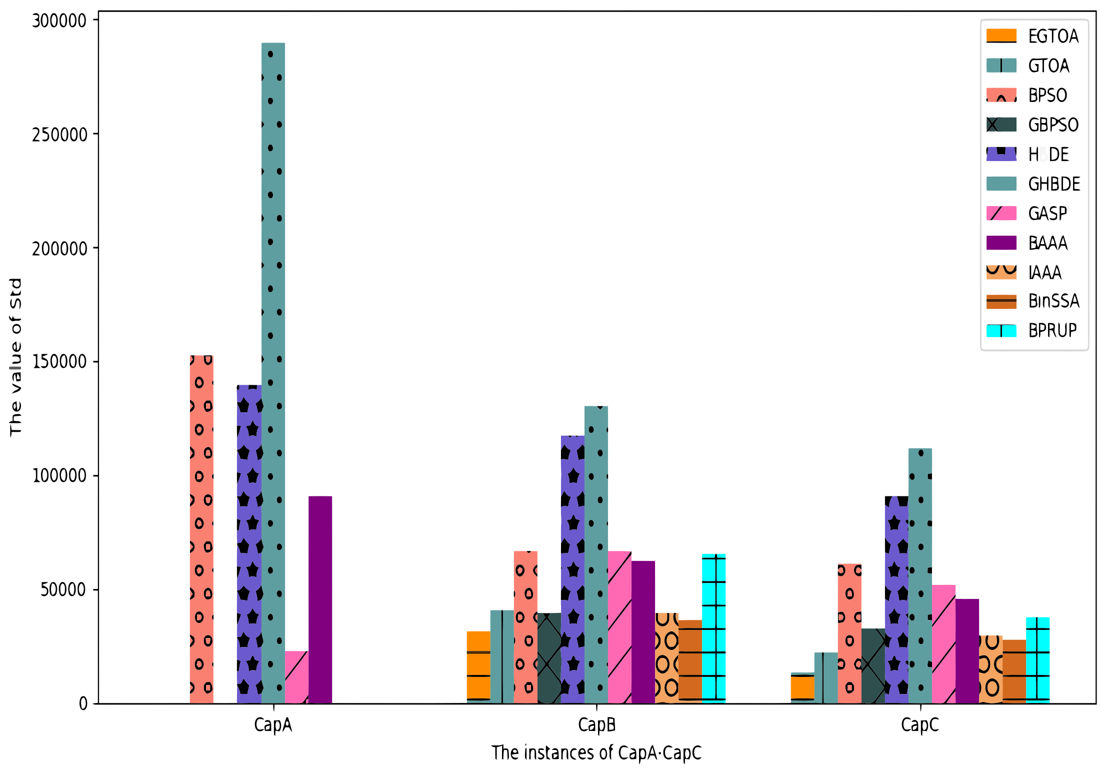


Fig. 4. The Std histograms of 11 algorithms.

In general, $P_v \in (0, 0.5)$ is more appropriate. It is easy to see that the time complexity of ODMO is $O(n)$.

4.3. Redundant checking strategy

After all customers are served by open facilities, that is, $y_{ij} = 1$ or $y_{ij} = 0$ is determined. It is possible that there exist some open facilities which do not provide service to any one of the customers. At this time, if closing these facilities, the total open costs will be reduced. For this reason, a redundant checking strategy RCS is presented, that is, checking whether there are open facilities that do not provide services for any customers, and if so, close them. Specifically, for each $x_j = 1$, judge whether $\sum_{j=1}^{n} y_{ij} = 0$? If yes, then $x_j \leftarrow 0$. It is clear that RCS can improve the quality of solutions and its time complexity is $O(m \times n)$.

4.4. Pseudo-code and flowchart of EGTOA

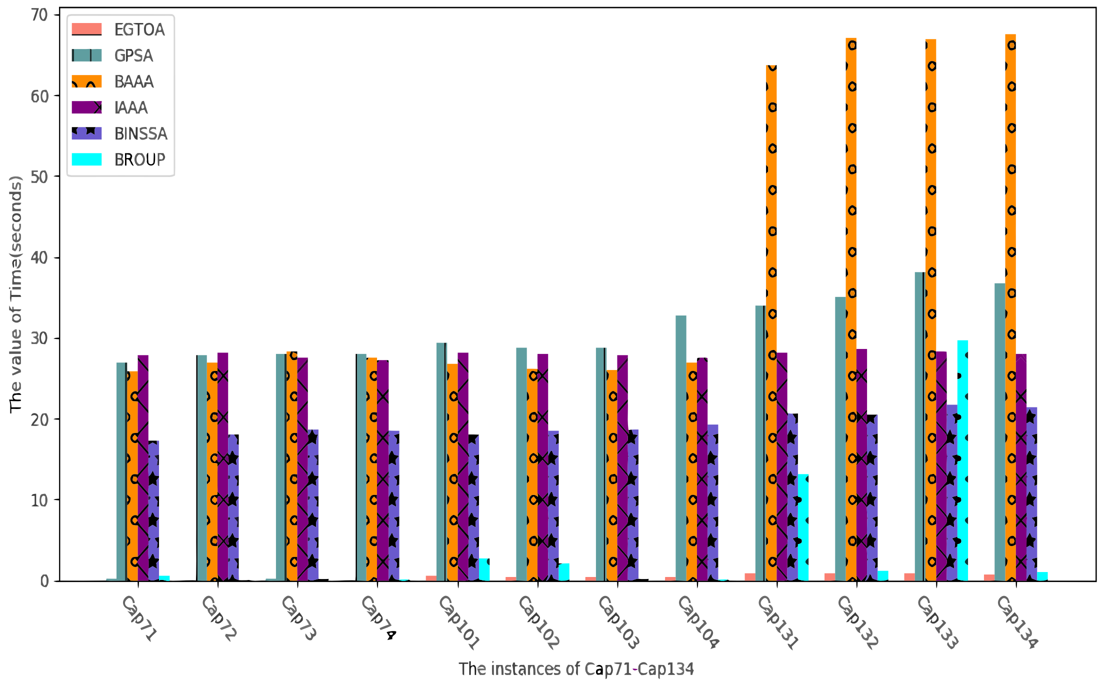
Let $\boldsymbol{Y} \leftarrow SAS(X)$ denote the operation that determines matrix $\boldsymbol{Y} = [y_{ij}]_{m\times n}$ by $X = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$, $X \leftarrow RCS(X, \boldsymbol{Y})$ represent the optimized operation of X by RCS, $X_a = (x_{a1}, x_{a2}, \ldots, x_{an}) \in \{0,1\}^n$ be a temporary vector and $B = (b_1, b_2, \ldots, b_n) \in \{0,1\}^n$ be the best solution at present. Let PS be the size of the population, MI be the number of iterations. Let $f(X) = \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}y_{ij} + \sum_{j=1}^{n} g_i x_j$, where $\boldsymbol{Y} = (y_{ij})$ be obtained by X. The pseudo-code of EGTOA is shown in Algorithm 4:

Algorithm 4. EGTOA

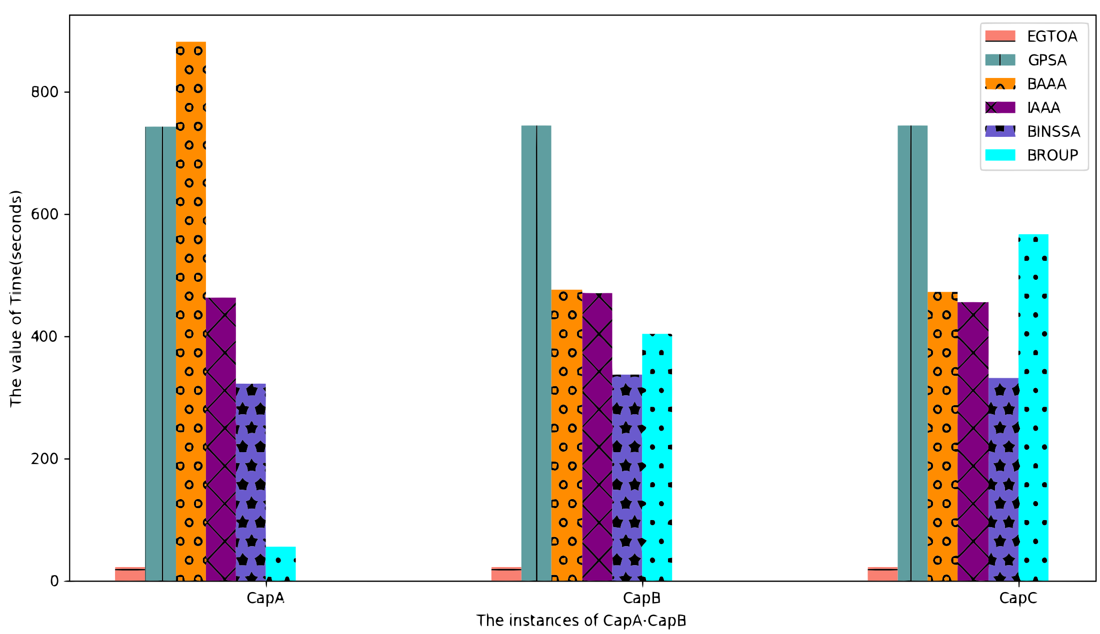
Input: An instance of UFLP, Parameters PS, MI, P_r and P_v;
Output: An approximate solution (or optimal solution) B and $f(B)$.
1 Using Algorithm 2 to generate $P(0) = \{X_c(0) | 1 \le c \le PS\}$;
2 for $c \leftarrow 1$ to PS do
3 $\boldsymbol{Y}_c \leftarrow SAS(Xc(0))$; $X_c(0) \leftarrow RCS(X_c(0), \boldsymbol{Y}_c)$;
4 end for
5 Compute $f(X_c(0))$, $1 \le c \le PS$; Determine B; $t \leftarrow 0$;
6 while $(t \le MI)$ do
7 for $c \leftarrow 1$ to PS do
8 $X_a \leftarrow X_{p1}(t) \oplus (F(X_{p2}(t) \oplus (-X_{p3}(t))))$;
9 $X_a \leftarrow SMO(X_a, P_r)$; $X_a \leftarrow ODMO(X_a, P_v)$;
10 $\boldsymbol{Y}_a \leftarrow SAS(X_a)$; $X_a \leftarrow RCS(X_a, \boldsymbol{Y}_a)$;
11 if $f(X_a) < f(X_c(t))$ then $X_c(t+1) \leftarrow X_a$; else $X_c(t+1) \leftarrow X_c(t)$;
12 end for
13 Generate $P(t+1)$; Determine B; $t \leftarrow t+1$;
14 end while
15 return(B, $f(B)$)

In Algorithm 4, the time complexity of Step 1 is $O(PS \times n)$. The time complexity of Step $2 \sim 4$ is $O(PS \times m \times n)$. The time complexity of Step $6 \sim 14$ is $O(MI \times PS \times m \times n)$. Therefore, the time complexity of EGTOA to solve UFLP algorithm is $O(PS \times n) + O(PS \times m \times n) + O(MI \times PS \times m \times n)$. Noting that both MI and PS are linear function of n, the time complexity of EGTOA is $O(mn^3)$.

Based on the discussion above, the flowchart of solving UFLP by EGTOA is shown in Fig. 1.



(a) The $Time$-histogram of 6 algorithms for small-scale instances



(b) The $Time$-histogram of 6 algorithms for large-scale instances

Fig. 5. The $Time$-histograms of 6 algorithms.

5. Computational experiments

In this section, the 15 benchmark instances of UFLP from OR-Library (Beasley, 1990) are used to validate the performance of EGTOA, the calculation results of EGTOA are compared with the 16 state-of-the-art algorithms including binABC (Kiran & GuNDuZ, 2014), ABC_{bin} (Kiran & Servet, 2015), DisDE (Husseinzadeh Kashan et al., 2013), binary differential evolution (BinDE) (Engelbrecht & Pampara, 2008), FA (Tsuya et al., 2017), MA (Atta et al., 2018), binary AAA with sigmoid logistic function (BAAA-Sig) (Korkmaz & Kiran, 2018), IAAA (Korkmaz & Kiran, 2018), BinSSA (Emine & Erkan, 2020), BinBRO with uniform crossover (BRO-UP) (Taymaz et al., 2022), genetic algorithm with single-point crossover (GA-SP) (Holland, 1992), GTOA (He & Wang, 2018), binary particle swarm optimization (BPSO) (Kennedy & Eberhart, 1997), binary differential evolution algorithm with hybrid encoding (HBDE) (He et al., 2007), GBPSO which is an improved BPSO

Table 5
The calculation results of EGTOA, GTOA, BPSO, GBPSO, HBDE, and GHBDE for Cap71–Cap74.

Algorithm	*Metric*	Cap71	Cap72	Cap73	Cap74
EGTOA	*#OPT*	**30**	**30**	**30**	**30**
	Mean	**932615.75**	**977799.40**	**1010641.45**	**1034976.98**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	**0.314**	**0.286**	**0.238**	**0.235**
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
GTOA	*#OPT*	**30**	**30**	**30**	**30**
	Mean	**932615.75**	**977799.40**	**1010641.45**	**1034976.98**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	**0.241**	**0.234**	**0.229**	**0.231**
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
BPSO	*#OPT*	**30**	**30**	**30**	**30**
	Mean	**932615.75**	**977799.40**	**1010641.45**	**1034976.98**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	**0.273**	**0.269**	**0.264**	**0.268**
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
GBPSO	*#OPT*	**30**	**30**	**30**	**30**
	Mean	**932615.75**	**977799.40**	**1010641.45**	**1034976.98**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	**0.435**	**0.392**	**0.308**	**0.295**
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
HBDE	*#OPT*	**30**	**30**	**30**	**30**
	Mean	**932615.75**	**977799.40**	**1010641.45**	**1034976.98**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	**0.274**	**0.263**	**0.238**	**0.235**
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
GHBDE	*#OPT*	**30**	**30**	**30**	**30**
	Mean	**932615.75**	**977799.40**	**1010641.45**	**1034976.98**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	**0.342**	**0.316**	**0.270**	**0.258**
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–

Table 6
The calculation results of EGTOA, GTOA, BPSO, GBPSO, HBDE, and GHBDE for Cap101–Cap104.

Algorithm	*Metric*	Cap101	Cap102	Cap103	Cap104
EGTOA	*#OPT*	**30**	**30**	**30**	**30**
	Mean	**796648.44**	**854704.20**	**893782.11**	**928941.75**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	**0.485**	**0.444**	**0.433**	**0.396**
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
GTOA	*#OPT*	**30**	**30**	**30**	**30**
	Mean	**796648.44**	**854704.20**	**893782.11**	**928941.75**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	**0.403**	**0.378**	**0.360**	**0.356**
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
BPSO	*#OPT*	21	**30**	16	**30**
	Mean	796906.52	**854704.20**	894227.40	**928941.75**
	Std	394.233	**0.000**	521.231	**0.000**
	Time	0.446	**0.439**	0.424	**0.391**
	GAP	0.030	**0.000**	0.067	**0.000**
	Sign	–	–	–	–
GBPSO	*#OPT*	24	**30**	16	**30**
	Mean	796820.50	**854704.20**	894046.20	**928941.75**
	Std	344.115	**0.000**	389.327	**0.000**
	Time	0.608	**0.553**	0.474	**0.429**
	GAP	0.022	**0.000**	0.030	**0.000**
	Sign	–	–	–	–
HBDE	*#OPT*	**30**	**30**	**30**	**30**
	Mean	**796648.44**	**854704.20**	**893782.11**	**928941.75**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	**0.431**	**0.414**	**0.380**	**0.332**
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
GHBDE	*#OPT*	**30**	**30**	**30**	**30**
	Mean	**796648.44**	**854704.20**	**893782.11**	**928941.75**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	**0.517**	**0.473**	**0.421**	**0.363**
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–

based on ODMO and RCS, and GHBDE which is an improved HBDE based on ODMO and RCS.

5.1. Computing environment and performance criteria

Algorithms EGTOA, GTOA, BPSO, GBPSO, HBDE, and GHBDE are performed on PC with Intel(R) Core(TM) i5-7500 CPU-3.40 GHz, 4 GB DDR3. The operating system is Microsoft Windows 7. The implementation of the algorithm adopts VC++, and the compilation environment is Code:Blocks (Note: the C++ source code of EGTOA can be downloaded from https://www.researchgate.net/publication/342846629_The_source_code_of_EGTOA_for_solving_UFLP_Problem acquisition). The computing environment of other algorithms can be found in the relevant literature and will not be repeated. All figures are drawn by using Python in the compilation environment JetBrains PyCharm.

All instances of UFLP are independently calculated 30 times. Multiple quantitative assessment measures, $\#OPT$, $Mean$, Std, $Time$, GAP, $Sign$, $Rank$, $Mean-R$, and $Fin-R$, are used to evaluate the obtained results. Where, $\#OPT$ is the number of optimal value obtained by algorithm. $Mean$ and Std are the average and standard deviation, respectively. $Time$ is the average running time (in seconds). GAP is the gap between the mean of best values ($f_{\mathcal{A}}$) obtained by algorithm $\mathcal{A}$ and the optimal value (f_{opt}). The formula of GAP is given by Eq. (10), where $f_{\mathcal{A}}(i)$ is the best value obtained by the algorithm $\mathcal{A}$ in the ith execution. $Sign$ stands for the results of Wilcoxon signed rank test (Wilcoxon, 1945) with 0.05 level of p. When $p > 0.05$, $Sign$ is "+ "; otherwise $Sign$ is "−". $Rank$ (Korkmaz & Kiran, 2018) is the sequence number of each algorithm after sorting in non-decreasing order of the GAP of each algorithm. $Mean-R$ (Korkmaz & Kiran, 2018) is the mean of $Rank$, and $Mean-R = (\sum_{i=1}^{15} Rank_i)/15$. $Fin-R$ (Korkmaz & Kiran, 2018) is the sequence number of each algorithm after sorting in non-decreasing order of the $Mean-R$ of each algorithm.

$$GAP = \frac{\frac{\sum_{i=1}^{30} f_{\mathcal{A}}(i)}{30} - f_{opt}}{f_{opt}} \tag{10}$$

It should be noted that the CPU clock frequency of PC used by GA-SP, BAAA-Sig and IAAA is higher than that used in this paper, so their $Time$ is not further processed. Because the CPU clock frequency of PC used by BinSSA and BRO-UP is slightly lower, for the sake of fairness, the $Time$ of BinSSA and BRO-UP is processed by using the formula $\frac{Time\ of\ BinSSA(or\,BRO-UP)}{\mu}$, where μ is the value of 3.4 divided by the CPU clock frequency of PC used by BinSSA (or BRO-UP). It should be noted that the time consumption of BRO-UP is not about the number of iterations. Therefore, such comparison BRO-UP is more advantageous.

5.2. Benchmark instances and algorithm parameters

The 15 benchmark instances of UFLP (Beasley, 1990) are divided into 4 classes depending on the scale of instance. The first class includes 4 small-scale instances with scale 16 × 50 (i.e. 16 facilities and 50 customers), which are named $Cap71 \sim Cap74$ respectively. The second class includes 4 medium-scale instances with scale 25 × 50, which are named $Cap101 \sim Cap104$ respectively. The third class includes 4 medium-scale instances with scale 50 × 50, which are named $Cap131 \sim Cap134$ respectively. The fourth class includes 3 large-scale instances with scale 100 × 1000, which are named $CapA \sim CapC$ respectively. All instances of UFLP are introduced in Table 1 with their scale and optimal value.

In order to guarantee the performance of EGTOA, GTOA, BPSO, GBPSO, HBDE, and GHBDE for solving UFLP, we use the Kruskal–Wallis

Table 7
The calculation results of EGTOA, GTOA, BPSO, GBPSO, HBDE, and GHBDE for Cap131–Cap134.

Algorithm	Metric	Cap131	Cap132	Cap133	Cap134
EGTOA	#OPT	**30**	**30**	**30**	**30**
	Mean	**793439.56**	**851495.33**	**893076.71**	**928941.75**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	**0.916**	**0.779**	**0.885**	**0.687**
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
GTOA	#OPT	**30**	**30**	27	**30**
	Mean	**793439.56**	**851495.33**	893147.25	**928941.75**
	Std	**0.000**	**0.000**	211.620	**0.000**
	Time	**0.799**	**0.753**	0.747	**0.707**
	GAP	**0.000**	**0.000**	0.008	**0.000**
	Sign	–	–	–	–
BPSO	#OPT	17	27	9	**30**
	Mean	794094.97	851512.81	893951.93	**928941.75**
	Std	1140.232	52.44	660.974	**0.000**
	Time	0.894	0.881	0.847	**0.816**
	GAP	0.0823	0.002	0.098	**0.000**
	Sign	–	–	+	–
GBPSO	#OPT	24	**30**	16	**30**
	Mean	796820.50	**851495.33**	894046.20	**928941.75**
	Std	344.115	**0.000**	389.327	**0.000**
	Time	1.083	**1.02**	0.949	**0.887**
	GAP	0.022	**0.000**	0.023	**0.000**
	Sign	–	–	–	–
HBDE	#OPT	**30**	**30**	**30**	**30**
	Mean	**793439.56**	**851495.33**	**893076.71**	**928941.75**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	**0.801**	**0.761**	**0.714**	**0.668**
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
GHBDE	#OPT	28	**30**	29	**30**
	Mean	793492.23	**851495.33**	893082.54	**928941.75**
	Std	197.903	**0.000**	31.378	**0.000**
	Time	0.8824	**0.816**	0.762	**0.708**
	GAP	0.007	**0.000**	0.001	**0.000**
	Sign	–	–	–	–

Table 8
The calculation results of EGTOA, GTOA, BPSO, GBPSO, HBDE, and GHBDE for CapA–CapC.

Algorithm	Metric	CapA	CapB	CapC
EGTOA	#OPT	**30**	**24**	**9**
	Mean	**17156454.48**	**12994726.03**	**11515694.03**
	Std	**0.000**	**31308.897**	**13443.815**
	Time	**22.181**	**21.730**	**21.597**
	GAP	**0.000**	**0.121**	**0.091**
	Sign	–	+	+
GTOA	#OPT	**30**	11	3
	Mean	**17156454.48**	13007494.11	11523972.75
	Std	**0.000**	40368.193	21870.9113
	Time	**21.746**	22.130	22.539
	GAP	**0.000**	0.219	0.160
	Sign	–	+	+
BPSO	#OPT	25	15	1
	Mean	17208192.35	13037773.48	11570131.78
	Std	152309.243	66830.793	60983.729
	Time	21.053	39.644	21.412
	GAP	0.302	0.452	0.561
	Sign	+	+	+
GBPSO	#OPT	**30**	18	5
	Mean	**17156454.48**	13011170.66	11523317.20
	Std	**0.000**	39530.960	32807.287
	Time	**21.207**	42.255	22.978
	GAP	**0.000**	0.247	0.154
	Sign	–	+	+
HBDE	#OPT	11	1	0
	Mean	17273345.86	13171852.67	11689430.10
	Std	139314.062	117325.863	90843.675
	Time	19.579	19.417	19.583
	GAP	0.681	1.485	1.337
	Sign	+	+	+
GHBDE	#OPT	7	0	0
	Mean	17427685.43	13285740.07	11753613.22
	Std	289649.914	130041.071	111617.846
	Time	19.371	21.151	19.658
	GAP	1.581	1.749	1.954
	Sign	+	+	+

test (Derrac et al., 2011; García et al., 2009; Sprent & Smeeton, 2007) to determine the reasonable value of the parameters in all algorithms. For the fairness of algorithm comparison, according to the literatures (Atta et al., 2018; Emine & Erkan, 2020; Korkmaz & Kiran, 2018), the population size of all algorithms in this paper is always set as $PS = 40$, and the number of iterations is always set as $MI = 2000$.

For determining the mutation probability P_v of EGTOA, we select 4 instances Cap71, Cap101, Cap131 and CapB, and set P_v = 0.05, 0.1, 0.2, 0.4, 0.6 to calculate and analysis respectively. Each instance is independently calculated 30 times for the 5 different values of P_v. The results of the Kruskal–Wallis test are given in Table 2, in which Res indicates whether the value of P_v affects the performance of EGTOA. If Res is "Y" then the value of P_v has an influence on the calculation result; otherwise, Res is "N". For 4 instances, Table 3 shows the number $\#OPT$ of the optimal value obtained in 30 independent calculation and the average running time $Time$ (in seconds). The boxplots are shown in Fig. 2, where the X-axis is 5 different values of the parameter P_v and the Y-axis is the best value of the objective function obtained by EGTOA.

It can be seen from Table 2: For small and medium-scales UFLP instances, the different values of the parameter P_v have little effect on the calculation results. But for large-scale UFLP instances, the parameter values have a significant impact on EGTOA. It can be seen from Table 3 and Fig. 2: For the instance Cap71, EGTOA can 100% obtain the optimal value when P_v takes 5 different values. For the instance Cap101, EGTOA can 100% obtain the optimal value when P_v = 0.2, 0.4 and 0.6. For the instance Cap131, EGTOA can 100% obtain the optimal value when P_v = 0.2 and 0.4. For large-scale instance CapB, the number of obtaining the optimal value is the most when P_v=0.2. According to the Kruskal–Wallis test results, it can be determined that the mutation probability P_v=0.2 is suitable in EGTOA.

The parameter P_r of EGTOA and GTOA, and all parameters of BPSO, GBPSO, HBDE, and GHBDE are determined by the same way above. To avoid repetition, omit it. The value of all parameters of BPSO, GBPSO, HBDE, GHBDE, GTOA, and EGTOA are listed in Table 4, respectively. In BPSO and GBPSO, W is the inertia weight; C_1, C_2 are the acceleration constants, and $[-V, V]^n$ is the search range of the particle velocity vector. In HBDE and GHBDE, CR is the crossover factor; FS is the scaling factor, and $[-A, A]^n$ is the search space of the real vector of individual. In GTOA and EGTOA, P_r is the mutation probability of SMO. In EGTOA, GBPSO and GHBDE, P_v is the mutation probability of ODMO. The parameter settings of other algorithms are exactly the same as those in relevant literature and will not be repeated.

5.3. Comparison and analysis

In order to illustrate the effectiveness of ODMO and RCS, Section 5.3.1 first compares GTOA, BPSO and HBDE with their improved algorithms EGTOA, GBPSO and GHBDE based on ODMO and RCS. For pointing out that EGTOA is a more competitive and faster algorithm for solving UFLP, Section 5.3.2 compares EGTOA with the state-of-the-art algorithms GA-SP, BAAA-Sig, IAAA, BinSSA, and BRO-UP. Finally, EGTOA is compared with other representative algorithms binABC, ABC_{bin}, DisDE, BinDE, FA, and MA for solving UFLP.

5.3.1. Comparing EGTOA with GTOA, BPSO, GBPSO, HBDE, and GHBDE

The calculation results of 4 classes of UFLP instances by BPSO, GBPSO, HBDE, GHBDE, GTOA, and EGTOA are given in $Table5 \sim$

Table 9
Comparing EGTOA with the state-of-the-art algorithms for Cap71–Cap74.

Algorithm	Metric	Cap71	Cap72	Cap73	Cap74
EGTOA	#OPT	**30**	**30**	**30**	**30**
	Mean	**932615.75**	**977799.40**	**1010641.45**	**1034976.98**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	**0.314**	**0.286**	**0.238**	**0.235**
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
GA-SP	#OPT	**30**	**30**	19	**30**
	Mean	**932615.75**	**977799.40**	1011314.48	**1034976.98**
	Std	**0.000**	**0.000**	899.650	**0.000**
	Time	26.957	27.893	27.994	27.998
	GAP	**0.000**	**0.000**	0.067	**0.000**
	Sign	–	–	+	–
BAAA-Sig	#OPT	**30**	**30**	**30**	**30**
	Mean	**932615.75**	**977799.40**	**1010641.45**	**1034976.98**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	25.868	26.927	28.219	27.461
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
IAAA	#OPT	**30**	**30**	**30**	**30**
	Mean	**932615.75**	**977799.40**	**1010641.45**	**1034976.98**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	27.901	28.078	27.528	27.239
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
BinSSA	#OPT	**30**	**30**	**30**	**30**
	Mean	**932615.75**	**977799.40**	**1010641.45**	**1034976.98**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	17.26	18.09	18.63	18.42
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
BRO-UP	#OPT	**30**	**30**	**30**	**30**
	Mean	**932615.75**	**977799.40**	**1010641.45**	**1034976.98**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	0.49	0.033	0.18	0.074
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–

Table 10
Comparing EGTOA with the state-of-the-art algorithms for Cap101–Cap104.

Algorithm	Metric	Cap101	Cap102	Cap103	Cap104
EGTOA	#OPT	**30**	**30**	**30**	**30**
	Mean	**796648.44**	**854704.20**	**893782.11**	**928941.75**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	**0.485**	**0.444**	**0.433**	**0.396**
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
GA-SP	#OPT	11	**30**	6	**30**
	Mean	797193.29	**854704.20**	894351.78	**928941.75**
	Std	421.655	**0.000**	505.036	**0.000**
	Time	29.372	28.730	28.689	32.706
	GAP	0.068	**0.000**	0.064	**0.000**
	Sign	+	–	+	–
BAAA-Sig	#OPT	**30**	**30**	**30**	**30**
	Mean	**796648.44**	**854704.20**	**893782.11**	**928941.75**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	26.836	26.215	25.926	26.963
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
IAAA	#OPT	**30**	**30**	**30**	**30**
	Mean	**796648.44**	**854704.20**	**893782.11**	**928941.75**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	28.177	27.923	27.838	27.592
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
BinSSA	#OPT	**30**	**30**	**30**	**30**
	Mean	**932615.75**	**977799.40**	**1010641.45**	**1034976.98**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	18.09	18.41	18.71	19.25
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
BRO-UP	#OPT	**30**	**30**	**30**	**30**
	Mean	**932615.75**	**977799.40**	**1010641.45**	**1034976.98**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	2.72	2.01	0.264	0.148
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–

$Table8$, in which the algorithm with the best performance is marked in bold.

It can be seen from Table 5: All algorithms can 100% obtain the optimal value of Cap71–Cap74, and their time consume is almost equal. Therefore, for the first class of UFLP instances, there are no significant differences in the performance of the 6 algorithms.

As seen from Table 6: EGTOA, GTOA, HBDE, and GHBDE can 100% obtain the optimal value of the second class of UFLP instances. BPSO and GBPSO can only 100% obtain the optimal value of Cap102 and Cap104. GBPSO can 24% obtain the optimal value of Cap101, BPSO can only 21% obtain the optimal value of Cap101. GBPSO and BPSO can 16% obtain the optimal value of Cap103. Moreover, there are no significant differences for the speed of 6 algorithms. Therefore, for the second class of UFLP instances, the performance of EGTOA, GTOA, HBDE, and GHBDE is best, and the performance of GBPSO is slightly better than that of BPSO.

It is easy to see from Table 7: EGTOA and HBDE can 100% obtain optimal value of the third class of UFLP instances. GTOA can 100% obtain the optimal value of instances Cap131, Cap132, and Cap134. GBPSO and GHBDE can 100% obtain the optimal value of instances Cap132 and Cap134. BPSO can only 100% obtain the optimal value of instance Cap134. Furthermore, The value of $Time$ of GTOA is slightly smaller than that of other 5 algorithms, and the value of $Time$ of GBPSO is slightly larger. Therefore, for the third class of UFLP instances, the performance of EGTOA and HBDE is best, and the performances of EGTOA and GBPSO are better than that of GTOA and BPSO respectively.

From the calculation results in Table 8: EGTOA, GTOA, and GBPSO can 100% obtain the optimal value of CapA. BPSO, HBDE, and GHBDE cannot 100% obtain the optimal value of the fourth class all instances. The number of optimal values of CapB and CapC obtained by EGTOA is obviously far more than other algorithms. The number of optimal values of CapB and CapC obtained by GBPSO is also obviously more than BPSO. And as well, there are not much differences between the time consume of 6 algorithms. Therefore, for the fourth class of UFLP instances, the performance of EGTOA is best, and the performances of EGTOA and GBPSO are better than that of GTOA and BPSO respectively.

Through the above comparison, it is not difficult to see that EGTOA is the best among the six algorithms. In addition, according to the fact that the solving effect of EGTOA and GBPSO is obviously better than that of GTOA and BPSO, we can see that ODMO and RCS can improve not only the performance of GTOA for solving UFLP, but also the performance of BPSO for solving UFLP. This shows that ODMO and RCS can indeed improve the ability of many evolutionary algorithm for solving UFLP, and they are very effective improvement strategies.

5.3.2. Comparison with the state-of-the-art algorithms for solving UFLP

To demonstrate the superior performance of EGTOA for solving UFLP quickly, it is compared with the state-of-the-art algorithms GA-SP, BAAA-Sig, IAAA, BinSSA, and BRO-UP. The calculation results of 6 algorithms for 4 classes of UFLP instances are given in Table 9 ∼ Table 12, in which the algorithm with the best performance is marked in bold, the symbol "*" indicates that this content of the algorithm does not exist (see Tables 9–12).

As can be seen from Table 9 ∼ Table 11: EGTOA, BAAA-Sig, IAAA, BinSSA, and BRO-UP can 100% obtain the optimal value of the first three classes of instances. GA-SP can only 100% obtain the optimal value of 7 instances of first three classes of instances. The value of $Time$ of EGTOA is at most 1/31 of GA-SP, BAAA-Sig, and IAAA, and at most 1/22 of BinSSA. Although the value of $Time$ of EGTOA is higher than that of BRO-UP for solving Cap71–Cap74, the value of $Time$ of

Table 11
Comparing EGTOA with the state-of-the-art algorithms for Cap131–Cap134.

Algorithm	*Metric*	Cap131	Cap132	Cap133	Cap134
EGTOA	*#OPT*	**30**	**30**	**30**	**30**
	Mean	**793439.56**	**851495.33**	**893076.71**	**928941.75**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	**0.916**	**0.779**	**0.885**	**0.687**
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
GA-SP	*#OPT*	16	**30**	10	**30**
	Mean	793980.10	**851495.33**	893891.91	**928941.75**
	Std	720.877	**0.000**	685.076	**0.000**
	Time	34.017	35.107	38.143	36.748
	GAP	0.068	**0.000**	0.091	**0.000**
	Sign	+	–	+	–
BAAA-Sig	*#OPT*	**30**	**30**	**30**	**30**
	Mean	**793439.56**	**851495.33**	**893076.71**	**928941.75**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	63.656	67.132	66.943	67.612
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
IAAA	*#OPT*	**30**	**30**	**30**	**30**
	Mean	**793439.56**	**851495.33**	**893076.71**	**928941.75**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	28.080	28.539	28.336	28.011
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
BinSSA	*#OPT*	**30**	**30**	**30**	**30**
	Mean	**932615.75**	**977799.40**	**1010641.45**	**1034976.98**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	20.69	20.48	21.64	21.34
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–
BRO-UP	*#OPT*	**30**	**30**	**30**	**30**
	Mean	**932615.75**	**977799.40**	**1010641.45**	**1034976.98**
	Std	**0.000**	**0.000**	**0.000**	**0.000**
	Time	13.094	1.112	29.71	1.029
	GAP	**0.000**	**0.000**	**0.000**	**0.000**
	Sign	–	–	–	–

Table 12
Comparing EGTOA with the state-of-the-art algorithms for CapA–CapC.

Algorithm	*Metric*	CapA	CapB	CapC
EGTOA	*#OPT*	**30**	**24**	**9**
	Mean	**17156454.48**	**12994726.03**	**11515694.03**
	Std	**0.000**	*31308.897*	**13443.815**
	Time	**22.181**	**21.730**	**21.597**
	GAP	**0.000**	**0.121**	**0.091**
	Sign	–	+	+
GA-SP	*#OPT*	24	9	2
	Mean	17164354.46	13054858.05	11586692.97
	Std	22451.206	66658.649	51848.248
	Time	741.535	743.370	744.455
	GAP	0.046	0.584	0.705
	Sign	–	+	+
BAAA-Sig	*#OPT*	16	1	1
	Mean	17210900.53	13093705.56	11583462.07
	Std	90743.456	62168.803	45788.678
	Time	880.452	475.592	470.911
	GAP	0.317	0.883	0.677
	Sign	+	+	+
IAAA	*#OPT*	**30**	15	1
	Mean	**17156454.48**	13011234.616	11539496.443
	Std	**0.000**	39224.744	29766.311
	Time	461.906	470.094	455.470
	GAP	**0.000**	0.248	0.295
	Sign	–	+	+
BinSSA	*#OPT*	**30**	*	*
	Mean	**17156454.48**	13012123.523	11555489.082
	Std	**0.000**	36156.14	27598.80
	Time	322.17	336.53	330.44
	GAP	**0.000**	0.255	0.434
	Sign	–	+	+
BRO-UP	*#OPT*	**30**	*	*
	Mean	**932615.75**	**977799.40**	**1010641.45**
	Std	**0.000**	65614.422	37520.233
	Time	55.63	403.18	566.94
	GAP	**0.000**	0.470	0.290
	Sign	–	+	+

EGTOA is basically the same as that of BRO-UP for solving Cap101–Cap104, and is significantly lower than that of BRO-UP for solving Cap131–Cap134. The above comparison results indicate that for the first three classes of UFLP instances, except for the poor solution effect of GA-SP, other algorithms can obtain 100% optimal solutions of all instances. Meanwhile, the speed of EGTOA is slightly faster than that of BRO-UP, and far faster than that of GA-SP, BAAA-Sig, IAAA, and BinSSA. Therefore, for small and medium-sized UFLP instances, the performance of EGTOA and BRO-UP is superior to GA-SP, BAAA-Sig, IAAA, and BinSSA.

From the calculation results in Table 12: EGTOA, IAAA, BinSSA, and BRO-UP can 100% obtain the optimal value of instance CapA. For instances CapB and CapC, the number of optimal values obtained by EGTOA is obviously far more than other algorithms, which is almost 1.5 times than that of IAAA. Neither GA-SP nor BAAA-Sig can 100% obtain the optimal value of any instance, and the calculation results of all instances obtained by them are obviously worse than those of EGTOA, IAAA, BinSSA, and BRO-UP. Moreover, the value of $Time$ of EGTOA is at most 1/20 of GA-SP, BAAA-Sig, and IAAA, at most 1/14 of BinSSA. Although the value of $Time$ of EGTOA is only at most 2/5 of BRO-UP for instance CapA, the value of $Time$ of EGTOA is at most 1/17 of BRO-UP for instance CapB and CapC. In addition, the value of Std and GAP of EGTOA is obviously better than other 5 algorithms. This indicates that EGTOA is much faster than the other 5 algorithms for solving large-scale UFLP instances. Therefore, for large-scale UFLP instances, EGTOA not only has a higher success rate of obtaining the optimal value, but also consumes less time.

In summary, the success rate of EGTOA is highest among the six algorithms, and its speed is also much faster than that of other algorithms. Therefore, the performance of EGTOA is more superior to GA-SP, BAAA-Sig, IAAA, BinSSA, and BRO-UP for solving UFLP.

5.3.3. Comparison with other algorithms for solving UFLP

In this section, we compare EGTOA with binABC, ABC_{bin}, DisDE, BinDE, FA, and MA, all of which are proposed in recent years and have good performance for solving UFLP. The parameters setting of the 6 algorithms is exactly the same as that in literature (Atta et al., 2018; Husseinzadeh Kashan et al., 2013; Kiran & GuNDuZ, 2014; Kiran & Servet, 2015; Tsuya et al., 2017). The calculation results of EGTOA, binABC, ABC_{bin}, DisDE, BinDE, FA, and MA for of 4 classes of UFLP instances are given in Table 13. It is easy to see that the smaller the $Rank$, $Mean-R$, and $Fin-R$ of algorithm are, the better the performance of the algorithm has.

It can be seen from Table 13 that the GAP and $Rank$ of EGTOA are better than those of other 6 algorithms for 14 UFLP instances except CapB. Even for instance CapB, EGTOA is only slightly worse than $DisDE$. This shows that the performance of EGTOA is better than that of ABC_{bin}, binABC, FA, MA, BinDE, and DisDE for solving UFLP.

5.3.4. Comparison of robustness, stability and time

Because GAP and Std are the important indicators for evaluating the performance of the algorithms, the GAP curves of EGTOA, GTOA, BPSO, GBPSO, HBDE, and GHBDE are given in Fig. 3(a), the GAP curves of EGTOA, GA-SP, BAAA-Sig, IAAA, BinSSA, and BRO-UP are given in Fig. 3(b), the GAP curves of EGTOA, binABC, ABC_{bin}, FA, MA, BinDE, and DisDE are given in Fig. 3(c). The values on X-axis are the name of UFLP instances and Y-axis are the values of GAP. Since the results of the fourth class of UFLP instances by FA and MA are not given in literatures (Atta et al., 2018; Tsuya et al., 2017), there are only partial GAP curves of the two algorithms in Fig. 3(c). In Fig. 4, the Std histogram of EGTOA, GTOA, BPSO, GBPSO, HBDE, GHBDE, GA-SP, BAAA-Sig, IAAA, BinSSA, and BRO-UP are given for large-scale

Table 13
The comparison of EGTOA with ABC_{bin}, binABC, FA, MA, BinDE, DisDE by using GAP and $Rank$.

Algorithm	EGTOA		ABC_{bin}		binABC		FA		MA		BinDE		DisDE	
	GAP	$Rank$	GAP	$Rank$	GAP	$Rank$	GAP	$Rank$	GAP	$Rank$	GAP	$Rank$	GAP	$Rank$
Cap71	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**
Cap72	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**
Cap73	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	0.0120	2	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**
Cap74	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**
Cap101	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	0.0970	4	0.0057	3	**0.0000**	**1**	0.0036	2
Cap102	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	0.0940	4	0.0003	2	**0.0000**	**1**	0.0049	3
Cap103	**0.0000**	**1**	0.0051	2	**0.0000**	**1**	0.0730	5	0.0057	4	**0.0000**	**1**	0.0055	3
Cap104	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	0.2460	2	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**
Cap131	**0.0000**	**1**	0.2000	4	**0.0000**	**1**	0.9930	5	0.0297	3	0.0036	2	0.0036	2
Cap132	**0.0000**	**1**	0.0200	4	0.0000	1	1.0630	5	0.0121	3	0.0050	2	**0.0000**	**1**
Cap133	**0.0000**	**1**	0.0750	4	0.1200	5	1.1210	6	0.0011	2	0.0138	3	0.0138	3
Cap134	**0.0000**	**1**	**0.0000**	**1**	**0.0000**	**1**	1.5160	3	0.0056	2	**0.0000**	**1**	0.0000	1
CapA	**0.0000**	**1**	3.2000	5	3.0000	4	–	–	–	–	1.3000	3	0.0370	2
CapB	0.2020	2	2.8000	5	2.5000	4	–	–	–	–	1.5200	3	**0.1890**	**1**
CapC	**0.0310**	**1**	2.0000	4	2.6000	5	–	–	–	–	1.5500	3	0.0909	2
$Mean-R$	**1.0670**		2.4667		2.0000		3.5833		2.1667		1.6667		1.6667	
$Fin-R$	1		4		3		5		3		2		2	

UFLP instances. The values on X-axis are the name of UFLP instances and Y-axis are the values of Std.

It can be seen from Fig. 3: with the increase of instance size continuously, the GAP curves of EGTOA almost always coincides with the X-axis, while the GAP curve of other algorithms are gradual upward trend. Therefore, the performance and robustness of EGTOA are better than those of other algorithms.

It can be seen from Fig. 4 that the Std values of EGTOA, GTOA, GBPSO, IAAA, BinSSA, and BRO-UP are zero for CapA, which indicate that those algorithms have the best stability for CapA. For instances CapB and CapC, the Std value of EGTOA is much smaller than that of other algorithms, which shows that EGTOA has best stability for CapB and CapC.

In order to intuitively illustrate the fast solution performance of EGTOA, the $Time$ histograms of EGTOA, GA-SP, BAAA-Sig, IAAA, BinSSA, and BRO-UP are given in Fig. 5. It is clear that the $Time$ of EGTOA is much smaller than that of GA-SP, BAAA-Sig and IAAA for all UFLP instances.

In summary, EGTOA not only has better robustness and stability than other algorithms, but also has lowest running time in all algorithms. Therefore, EGTOA is a faster and more competitive algorithm for solving UFLP.

6. Conclusion

In this study, an enhanced group-based optimization algorithm EGTOA based on one direction mutation operator ODMO and redundant checking strategy RCS is proposed. The calculation of 15 benchmark instances of UFLP from OR-Library (Beasley, 1990) shows that ODMO and RCS can not only greatly improve the performance of EGTOA for solving UFLP, but also apply to improve the performance of algorithms such as BPSO to solve UFLP. The comparison between EGTOA and the 16 state-of-the-art algorithms for solving 15 benchmark instances of UFLP shows that EGTOA not only has better calculation results and stronger stability, but also its speed is much faster than other algorithms, which indicates that EGTOA is a new and more competitive evolutionary algorithm for quickly solving UFLP.

Obviously, the improvement of EGTOA based on ODMO and RCS is very successful and effective, and this improvement of GBPSO is also successful. But the improvement is not applicable for HBDE. This indicates that ODMO and RCS are not suitable for every evolutionary algorithm. Why did this happen? It is obviously a problem worth studying. In addition, although EGTOA has shown excellent performance in solving the uncapacitated facility location problem quickly, whether it can also solve other facility location problems quickly and efficiently (such as capacity constrained facility location problem, multi criteria facility location problem, etc.) is also a worthy of further discussion. Thirdly, EGTOA is very suitable for solving combinatorial optimization problems with integer vector as feasible solution. Although it performs well in solving UFLP, whether it can still maintain good performance for solving other combinatorial problems, such as set covering problem (Punnen & Pandey, 2019), knapsack problems (He & Wang, 2018) and satisfiability problem (Du et al., 2011), is a problem worthy of study. In view of the above problems, we will conduct in-depth research and discussion in the future.

CRediT authorship contribution statement

Fazhan Zhang: Writing – original draft, Software, Investigation. **Yichao He:** Conceptualization, Methodology, Writing – review & editing. **Haibin Ouyang:** Resources, Conceptualization, Writing – review & editing. **Wenben Li:** Software, Formal analysis.

Declaration of competing interest

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

Data availability

No data was used for the research described in the article.

Acknowledgments

We thank Editor-in-Chief and anonymous reviewers whose valuable comments and suggestions help us significantly improve this article. The first author and corresponding authors contributed equally the same to this article which was supported by Natural Science Foundation of Hebei Province, China (F2020403013), and Funded by Science and Technology Project of Hebei Education Department, China (ZD2021016).

References

Akinc, U., & Khumawala, B. M. (1977). An efficient branch and bound algorithm for the capacitated warehouse location problem. *Management Science*, *23*(6), 585–594.

Armas, J. D., & Juan, A. A. (2018). A biased-randomized algorithm for the uncapacitated facility location problem. In *Applied mathematics and computational intelligence* (pp. 287–298).

Atta, S., Mahapatra, P. R. S., & Mukhopadhyay, A. (2018). Solving uncapacitated facility location problem using monkey algorithm. *Intelligent Engineering Informatics*, *695*, 71–78.

Aydin, M. E., & Fogarty, T. C. (2004). A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems. *Journal of Heuristics*, *10*(3), 269–292.

Azad, M., Rocha, A. M. A., & Fernandes, E. M. (2013). A simplified binary artificial fish swarm algorithm for uncapacitated facility location problems. *Lecture Notes in Engineering & Computer Science*, *2204*(1), 31–36.

Beasley, J. E. (1990). OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, *41*(11), 1069–1072.

Bumb, A. (2002). Approximation algorithms for facility location problems. *67*(11), 1–5.

Byrka, J., & Aardal, K. (2007). An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. In M. Charikar, & et al. (Eds.), *Lecture notes in computer science*: *vol. 4627*, *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*. Berlin: Springer.

Cai, Z., Peng, J., & Gao, W. (2005). Improved evolutionary algorithm for the traveling salesman problem. *Journal of Computer Science*, *05*, 823–828.

Conn, A. R., & Cornuejols, G. (1990). A projection method for the uncapacitated facility location problem. *Mathematical Programming*, *46*(1), 273–298.

Cura, T. (2010). A parallel local search approach to solving the uncapacitated warehouse location problem. *Computers & Industrial Engineering*, *59*(4), 1000–1009.

Damgacioglu, H., Dinler, D., Ozdemirel, N. E., & Iyigun, C. (2015). A genetic algorithm for the uncapacitated single allocation planar hub location problem. *Computers & Operations Research*, *62*, 224–236.

Daskin, M., Snyder, L., & Berger, R. (2003). Facility location in supply chain design. In *Logistics systems: design and optimization* (pp. 39–65).

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics Engineering*, *186*(2/4), 311–338.

Derrac, J., Garcia, 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 & Evolutionary Computation*, *1*(1), 3–18.

Du, D., Ge, K., & Hu, X. (2011). *Design and analysis of approximation algorithm*. Beijing: Higher Education Press.

Duan, H., & Fei, Y. E. (2017). Progresses in pigeon-inspired optimization algorithms. *Journal of Beijing University of Technology*, *43*(1), 1–7.

Efroymson, M. A., & Ray, T. L. (1966). A branch-bound algorithm for plant location. *Operations Research*, *14*(3), 361–368.

Emarya, E., Zawbaabc, H. M., & Hassanienab, A. E. (2016). Binary grey wolf optimization approaches for feature selection. *Neurocomputing*, *172*, 371–381.

Emine, B., & Erkan, U. (2020). A binary social spider algorithm for uncapacitated facility location problem. *Expert Systems with Applications*, *161*, Article 113618.

Engelbrecht, A. P., & Pampara, G. (2008). Binary differential evolution strategies. In *IEEE congress on evolutionary computation*.

Galvvao, R. D., & Raggi, L. A. (1989). A method for solving to optimality uncapacitated location problems. *Annals of Operations Research*, *18*(1), 225–244.

García, S., Molina, D., Lozano, M., & Herrera, F. (2009). A study on the use of nonparametric tests for analyzing the evolutionary algorithms behaviour: A case study on the CEC2005 special session on real parameter optimization. *Journal of Heuristics*, *15*(6), 617–644.

Ghaderi, A., Jabalameli, M. S., Barzinpour, F., & Rahmaniani, R. (2012). An efficient hybrid particle swarm optimization algorithm for solving the uncapacitated continuous location-allocation problem. *Networks & Spatial Economics*, *12*(3), 421–439.

He, Y. C., & Wang, X. Z. (2018). Group theory-based optimization algorithm for solving knapsack problems. *Knowledge-Based Systems*.

He, Y., Wang, X., & Kou, Y. (2007). A binary differential evolution algorithm with hybrid encoding. *Journal of Computer Research Development*, *44*(9), 1476–1484.

Holland, J. (1992). Genetic algorithms. *Scientific American*, *267*(1), 66–72.

Husseinzadeh Kashan, M., Husseinzadeh Kashan, A., & Nahavandi, N. (2013). A novel differential evolution algorithm for binary optimization. *Computational Optimization & Applications*, *55*(2), 481–513.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks, vol. 4* (pp. 1942–1948).

Kennedy, J., & Eberhart, R. (1997). A discrete binary version of the particle swarm algorithm. *Systems, Man, and Cybernetics*, *5*, 4104–4108.

Kiran, M. S., & GuNDuZ, M. (2014). XOR-based artificial bee colony algorithm for binary optimization. *Turkish Journal of Electrical Engineering & Computer Sciences*, *21*(suppl.2), 2307–2328.

Kiran, & Servet, M. (2015). The continuous artificial bee colony algorithm for binary optimization. *Applied Soft Computing*, *33*, 15–23.

Korkmaz, S., Babalik, A., & Kiran, M. S. (2017). An artificial algae algorithm for solving binary optimization problems. *International Journal of Machine Learning and Cybernetics*, *9*(7), 1233–1247.

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

Kratica, J., Duvsan Tovsic, V. F., & Ljubic, I. (2001). Solving the simple plant location problem by genetic algorithm. *RAIRO-Operations Research*, *35*(01), 127–142.

Kuehn, A. A., & Hamburger, M. J. (1963). A heuristic program for locating warehouses. *Management Science*, *9*(4), 643–666.

Lazic, N., Frey, B. J., & Aarabi, P. (2010). Solving the uncapacitated facility location problem using message passing algorithms. *Expert Systems with Applications*, *9*(6), 429–436.

Lazic, N., Givoni, I., Aarabi, P., & Frey, B. (2009). FLoSS: Facility location for subspace segmentation. In *IEEE international conference on computer vision, vol. 30, no. 2* (pp. 825–832).

Li, S. (2013). A 1.488 approximation algorithm for the uncapacitated facility location problem. In *Lecture Notes in Computer Science*: *vol. 6756*, (pp. 77–88).

Manne, A. S. (1964). Plant location under economies of scale decentralization and computation. *Management Science*, *11*(2), 213–235.

Meng, Z., Chen, Y., Li, X., Yang, C., & Zhong, Y. (2020). Enhancing QUasi-affine transformation evolution (QUATRE) with adaptation scheme on numerical optimization. *Knowledge-Based Systems*, *197*, Article 105908.

Meng, Z., & Pan, J. (2016). Monkey king evolution: A new memetic evolutionary algorithm and its application in vehicle fuel consumption optimization. *Knowledge-Based Systems*, *97*, 144–157.

Meng, Z., Pan, J.-S., & Tseng, K.-K. (2019). PaDE: An enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization. *Knowledge-Based Systems*, *168*, 80–99.

Meng, Z., Pan, J. S., & Xu, H. (2016). QUasi-Affine TRansformation evolutionary (QUATRE) algorithm: A cooperative swarm based algorithm for global optimization. *Knowledge-Based Systems*, *109*(OCT.1), 104–121.

Meng, Z., & Yang, C. (2021). Hip-DE: Historical population based mutation strategy in differential evolution with parameter adaptive mechanism. *Information Sciences*, *562*, 44–77.

Michel, L., & Hentenryck, P. V. (2004). A simple Tabu search for warehouse location. *European Journal of Operational Research*, *157*(3), 576–591.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51–67.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advance in Engineering Software*, *69*(3), 46–61.

Mitchell, M. (1996). *An introduction to genetic algorithms*. Massachusetts Institute of Technology.

Moses, C., & Sudipto, G. (1999). Improved combinatorial algorithms for facility location problems. In *Foundations of computer science, 1999. 40th annual symposium on*.

Punnen, A. P., & Pandey, P. (2019). Representations of quadratic combinatorial optimization problems: A case study using the quadratic set covering problem. *Computers and Operations Research*, *112*, Article 104769.

Sevkli, M., & Guner, A. R. (2006). A continuous particle swarm optimization algorithm for uncapacitated facility location problem. In *Lecture notes in computer science*, (pp. 316–323).

Sprent, P., & Smeeton, N. (2007). *Applied nonparamteric statistical methods* (4th ed.).

Stollsteimer, & F., J. (1963). A working model for plant numbers and locations. *Journal of Farm Economics*, *45*(3), 631–645.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In *Proceedings of the 3rd international conference on genetic algorithms* (pp. 2–9).

Tan, Y., & Zhu, Y. (2010). Fireworks algorithm for optimization. In *Advances in swarm intelligence: first international conference, vol. LNCS 6145* (pp. 355–364).

Taymaz, A., Saeid, A., & Rahim, D. (2022). BinBRO: Binary battle royale optimizer algorithm. *Expert Systems with Applications*, *195*, Article 116599.

Tsuya, K., Takaya, M., & Yamamurb, A. (2017). Application of the firefly algorithm to the uncapacitated facility location problem. *Intelligent & Fuzzy Systems*, *32*(4), 3201–3208.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, *1*(6), 80–83.

Yazdani, M., & Jolai, F. (2016). Lion optimization algorithm (LOA): A nature-inspired metaheuristic algorithm. *Journal of Computational Design and Engineering*, *3*(1), 24–36.