Scientific Research Publishing

Open Journal of Applied Sciences, 2023, 13, 685-695
https://www.scirp.org/journal/ojapps
ISSN Online: 2165-3925
ISSN Print: 2165-3917

Improved Adaptive Differential Evolution Algorithm for the Un-Capacitated Facility Location Problem

Nan Jiang, Huizhen Zhang*

Business School, University of Shanghai for Science and Technology, Shanghai, China
Email: *jiangnan70482@126.com

How to cite this paper: Jiang, N. and Zhang, H.Z. (2023) Improved Adaptive Differential Evolution Algorithm for the Un-Capacitated Facility Location Problem. *Open Journal of Applied Sciences*, **13**, 685-695.
https://doi.org/10.4236/ojapps.2023.135054

Received: April 13, 2023
Accepted: May 16, 2023
Published: May 19, 2023

Abstract

The differential evolution algorithm is an evolutionary algorithm for global optimization and the un-capacitated facility location problem (UFL) is one of the classic NP-Hard problems. In this paper, combined with the specific characteristics of the UFL problem, we introduce the activation function to the algorithm for solving UFL problem and name it improved adaptive differential evolution algorithm (IADEA). Next, to improve the efficiency of the algorithm and to alleviate the problem of being stuck in a local optimum, an adaptive operator was added. To test the improvement of our algorithm, we compare the IADEA with the basic differential evolution algorithm by solving typical instances of UFL problem respectively. Moreover, to compare with other heuristic algorithm, we use the hybrid ant colony algorithm to solve the same instances. The computational results show that IADEA improves the performance of the basic DE and it outperforms the hybrid ant colony algorithm.

Keywords

Un-Capacitated Facility Location Problem, Differential Evolution Algorithm, Adaptive Operator

1. Introduction

Facility location problem is a classic problem in operational research. Its objective is to select the optimal location for a facility from candidate sites to minimize the total cost while meeting the demand of customers. One of the most studied variants of facility location problems is the Un-capacitated Facility Location (UFL) Problem. In this variant, each facility can serve an unlimited number of

*Corresponding author.

customers, but each facility has a fixed cost associated with it. The UFL problem can also be applied for practical issues, for instance, the optimal solution of the problem can determine the network of oilfield pipeline [1]. Building the path of railway resource reserve location based on the UFL problem, the operation time can be shortened to find optimal solution [2]. Additionally, the UFL problem can also be applied in scheduling and routing [3].

UFL problem is known to be a NP-Hard problem, which means that finding an exact solution to the problem requires an exponentially increasing amount of computation as the problem size grows. Current algorithms for solving the problem can be categorized into three types: exact algorithms, approximate algorithms and heuristic algorithms. Exact algorithms are able to find the optimal solution to the problem [4], but they are not practical for large-scale problems due to their low efficiency. Approximate algorithms, on the other hand, can provide a feasible solution to the problem in less time, but they may not be able to give an optimal solution to the problem [5].

In recent years, heuristic algorithms have been developed to solve UFL problems. These algorithms combine various search to improve the convergence speed and to obtain better feasible solutions for large scale problems, for instance, the Lagrangian wolf pack algorithm [6], pseudo-Boolean model and heuristic algorithm [7], hybrid bat algorithm [8]. These algorithms demonstrate the feasibility of heuristic algorithms and better optimization performance than other types of algorithms.

One of the most promising heuristic algorithms is the differential evolution (DE) algorithm. DE was first proposed by Storn and Price [9] for solving continuous optimization problems, but it has been found to be computationally inefficient and slow to converge when solving discrete optimization problems [10] such as UFL problem.

Therefore, in this paper, we propose an improved differential evolution algorithm for solving the UFL problem. This approach combines the characteristics of the UFL problem with the DE algorithm, using the activation function to transform variable into binary one, for solving UFL problems. Meanwhile, to improve the convergence speed and optimization performance of the basic DE algorithm, we introduce the adaptive operator into the algorithm. We demonstrate the feasibility and effectiveness of our algorithm through experiments on classic UFL problem instances. The computational results show that our algorithm outperforms the basic DE algorithm and other heuristic algorithms in terms of convergence speed and optimization performance, making it a practical approach for solving large-scale UFL problems.

The remainder of the paper is organized as follows. Section 2 is the brief introduction and mathematical model of UFL problem. Section 3 presents the improved adaptive differential algorithm for the UFL problem. Section 4 provides the computational results of our algorithm compared to the basic DE algorithm and hybrid ant colony algorithm. Section 5 presents our concluding remark.

2. Un-Capacitated Facility Location Problem

Assuming that there is no capacity limit on facilities, given a set of facility locations $F=\{1,2,\cdots,m\}$, and a set of customers $C=\{1,\cdots,n\}$, for any $i\in F$, $j\in C$, C_{ij} represent the transportation cost between facility *i* and customer *j*, f_i represent the construction cost of the facility *i*, and the requirement that for all customers there must be and only be one facility that satisfies their demand while the total cost is minimized eventually. The UFL problem can be represented by the following mathematical model:

$$\min Z=\sum_{j=1}^{m}\sum_{i=1}^{n}C_{ij}\cdot x_{ij}+\sum_{i=1}^{m}f_i\cdot y_i \tag{1}$$

$$\text{s.t.}\quad \sum_{i=1}^{m}x_{ij}=1,\quad \forall j\in C \tag{2}$$

$$x_{ij}\le y_i,\quad \forall i\in F,\quad \forall j\in C \tag{3}$$

$$x_{ij}\in\{0,1\},\quad \forall i\in F,\quad \forall j\in C \tag{4}$$

$$y_i\in\{0,1\},\quad \forall i\in F \tag{5}$$

The x_{ij} indicates whether customer *j* chooses facility *i* to serve him or her, if so then $x_{ij}=1$, otherwise $x_{ij}=0$, and y_i indicates whether facility *i* is open, if so then $y_i=1$, otherwise $y_i=0$. The objective function is given by Equation (1), minimizing the total cost; constraint (2) ensures that each customer has and only has one facility to meet his demand; constraint (3) guarantees that only the construction of the facility will make it possible to serve the customer.

Combined with the analysis of the specific features of the UFL problem in the literature [11], the theorem can be concluded as follows:

Theorem 1 Suppose the optimal set of solutions to the UFL problem is X^*, a solution to the problem is $(x^*,y^*)\in X^*$, the constructed set of facilities is $O=\{i\mid y_i^*=1,i\in F\}$, for $\forall i\in O$, the set of clients is $B_i=\{j\mid x_{ij}^*=1,j\in C\}$, then the following is satisfied:

1) $\sum_{j\in B_i}C_{ij}+f_i=\min_{i'\in F}\left\{\sum_{j\in B_i}C_{i'j}+f_{i'}\right\}$

2) $\forall j\in B_i$, $c_{ij}=\min_{i'\in O}\{C_{i'j}\}$

Theorem 1 shows that for any set of facilities $F'\in F$, the facility with the lowest sum of transportation cost and construction cost in the set of facilities should be chosen to minimize the total cost; furthermore, for any customer *j*, the facility with the lowest corresponding service cost should be chosen to serve him.

3. Improved Adaptive Differential Evolution Algorithm

3.1. Differential Evolution Algorithm

Differential Evolution (DE) algorithm is a heuristic algorithm which is similar to genetic algorithms, simulating the laws of biological evolution in nature and including the operations of mutation, hybridization and selection. The main idea is to select two solutions from the initial solution vectors, by subtracting one solution from the other, multiply the difference by the variation operator and add

it back to the initial vector; compare the fitness of the initial vector and the variation vector, then choose the better one to keep completing this evolutionary process.

The DE algorithm is efficient in the optimization of continuous spaces, but the variable in the UFL problem is binary. Thus, to use the DE algorithm in the UFL problem, we need to modify the algorithm to transform the continuous variable into binary one.

3.2. Activation Function

Since the variables in Equation (5), indicating whether the facility the facility is opened, are binary variables, but the variables in basic differential evolution algorithm are continuous variables. Therefore, it is necessary to find a function to map the corresponding continuous values to binary values {0, 1}. Inspired by the activation function in neural networks, the hyperbolic tangent function (tanh) as a activation function is applied to convert continuous variables into (−1, 1). The formula for the tanh function:

$$\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

Next, to demonstrate the conversion of activation function, the following is the overall flow of the differential algorithm combined with the activation function.

1) Initialization Given the initial population size $M = m$, the dimension of the vector $N = n$, let $P(i, j)$ of two-dimensional array P denote every y_i in the population; i indicates the number of individual in the population, j indicates the corresponding y_i in the individual, and the demonstration of transformation is shown in Figure 1. The For every $P(i, j)$ in the array, let $P(i, j) = \text{rand}(P_{\min}, P_{\max})$, where $P_{\min}$ denotes the lower bound of the $P(i, j)$, $P_{\max}$ denotes the upper bound of $P(i, j)$, which means every elements in the array is a random number between lower bound and upper bound in the initialization. The upper limit of iteration is EP.

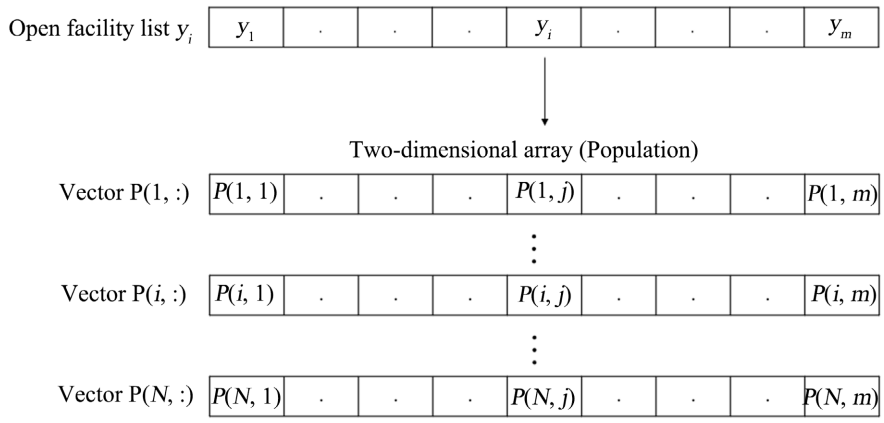

Figure 1. Initialization of population.

To avoid the situation that Equation (2) is not satisfied in the following steps, for every vector $P(i,:)$ in the P, if every $P(i,j) < \tanh^{-1}(T)$, let the maximum variable in the vector be $\tanh^{-1}(T)$, where T is the threshold of the activation function.

2) Differential For every vector $P(i,:)$ in P, randomly select two vectors $P(a,:)$ and $P(b,:)$ from it, and let vector $P'(i,:)$ in a two-dimensional array $P'(i,j)$ record the difference of $P(a,:)$ and $P(b,:)$, where $i,a,b \in \{1,2,\cdots,m\}$ and $a \neq b \neq i$.

3) Mutation For each value $P'(i,j)$ in every vector $P'(i,:)$, if the random value $R < CR$, $P'(i,j) = P(i,j)$, otherwise $P'(i,j)$ remains.

4) Selection To calculate the fitness of every individual in the population, the array Y and Y' should firstly transformed into binary variable, thus, let the two-dimensional array $B(i,j) = \tanh(P(i,j))$, and the two-dimensional array $B'(i,j) = \tanh(P'(i,j))$. Similarly, to avoid the situation that all the $P'(i,j)$ in one vector $P'(i,:)$ is 0, for every vector $P'(i,:)$ in the P', if $\text{sum}(B'(i,:)) = 0$, let the maximum $P'(i,j)$ in vector $P'(i,:) = \tanh^{-1}(T)$, and change the corresponding $B'(i,j) = 1$ so that Equation (1) can be satisfied, which is shown in Figure 2.

According to the theorem 1, when the set of facilities has been determined, the customer can be served by finding the lowest cost among the facilities that are open. This choice must be optimal under the condition that the set of facility is determined. Therefore, we can find the correspondingly optimal x_{ij} for every vector in P or P' as the open facility set, by searching the lowest service cost among the open facilities according to the P or P', by which the fitness can also be calculated.

5) Determination If iteration times $e = EP$, end the algorithm and output the best solution. Otherwise, $e = e + 1$, and go to (2).

3.3. Adaptive Optimization

To consider the current population optimum in the algorithm, the formula in (2) differential is improved to

$$P'(i,:) = P(i,:) + \left(P(a,:) - P(b,:)\right) + \text{rand}(1) * \left(P_{\text{best}} - P(i,:)\right),$$

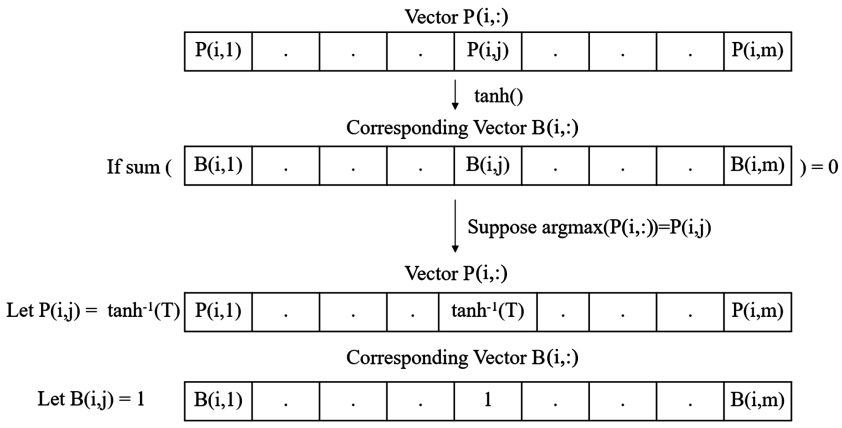

Figure 2. Satisfy Equation (2).

where $i,b,c \in \{1,2,\cdots,m\}$ and $b \neq c \neq i$, rand(1) is a random number from [0, 1], and P_{best} represent the vector with the best fitness in the population.

In addition, the differential evolution algorithm has the problem of slow descent and the tendency to fall into local optimum as the introduction of population optimum solution, this article improves the calculation of the vector v_i in the original algorithm by adding the adaptive operator $W = W_{\max} - w * c$, $W \geq W_{\min}$, where $W_{\max}$ is the upper bound of the operator, w is the coefficient, c is the number of repetitions of the current result, and $W_{\min}$ is the lower bound of the operator. The final formula of differential as follow:

$$P'(i,:) = P(i,:) + W * (P(a,:) - P(b,:)) + \text{rand}(1) * (P_{\text{best}} - P(i,:))$$

Note that the upper bound of the W is lower than 1 but the lower bound of W usually higher than 0.4, otherwise the algorithm will excessively focus on the population optimum which will make the population more likely to stuck at the local optimum. Conversely, if the upper bound of W is too high, the effect of the population optimum will be slight, which means the speed of convergence will not be improved. Therefore, it is significant to choose the upper bound and lower bound of the adaptive operator W.

3.4. Algorithm Flow

Let m = the number of facilities, n = the number of clients, N = the size of population and the set of facilities $Y_i = \{y_1, y_2, \cdots, y_m\}$, $i \in \{1,2,\cdots,N\}$. Set the upper bound $P_{\max}$, the lower bound $P_{\min}$, the adaptive cross factor CR. P denote the two-dimensional array, P' denote the two-dimensional array after the difference step of P. B refers to the binary array corresponding to P, $B(i,j) \in \{0,1\}$, 1 means the current facility is open, otherwise it is 0, and the B' refers to the corresponding binary array of P'.

The activation function is tanh(), and the threshold is T. The Cost matrix records the transportation cost between the facilities and the customers, of which the size is $m * n$. According to the theorem 1, $S(x, y)$ indicate the action of selecting the cost of open facilities from cost matrix by vector in P or P', where x is the cost of one customer and y is the vector in P or P'. Vector fit(i) record the fitness of every individual in P, similarly, vector fit (i) record the fitness of every individual in P', EP = total iteration limit and e = the current number of iteration. After the introduction of activation function and the adaptive optimization to the differential algorithm, the pseudocode of the IADEA as follow (**Algorithm 1**):

3.5. Complexity Analysis

The time complexity of step 1 is $O(N * m)$. Since the complexity of the action $S(x)$ is $O(m)$, the time complexity of step 2 is $O(n * m^2)$ and the time complexity of step 5 is $O(n * m^2 + N)$. The time complexity of step 3 is $O(N)$ and the time complexity of step 4 is $O(N * m)$. Thus, the overall time complexity of the algorithm is $O(N * m) + O(n * m^2) + O(N) + (N * m) + O(n * m^2 + N)$.

```
1:while e <= EP
2:    for i=1:N // step1 Determine the binary matrix by activation function
3:        for j = 1:m
4:            if tanh(P(i,j)) > T
5:                B(i,j) = 1
6:            else
7:                B(i,j) = 0
8:            end
9:        end
10:   end
11:   for i=1:N // step2 Calculate the current fitness
12:       for j=1:m
13:           fit(x) = fit(x) + min(S(Cost(:,j),P(i,:)))
14:       end
15:   end
16:   for i=1:N // step3 Differential
17:       c = argmin(fit(i))
18:       P'(i,:)= P(i,:) + W*(P(a,:)-P(b,:)) + rand(1)*(P(c,:)-P(i,:))
19:   end
20:   for i=1:N // step4 Cross
21:       for j=1:m
22:           if rand(1) < CR
23:               P'(i,j)=P(i,j)
24:           end
25:           B'(i,j) = tanh(P'(i,j))
26:       end
27:   end
28:   for i=1:N // step5 Calculate the fitness after cross step
29:       for j=1:m
30:           fit'(i) = fit'(i) + min(S(Cost(:,j),P'(i,:)))
31:       end
32:      for i=1:N
33:           if fit(i)>fit'(i)
34:               P(i,:) = P'(i,:)
35:           end
36:       end
37:   end
38:   e=e+1
```

Algorithm 1. IADEA.

4. Numerical Analysis

To verify the feasibility of the IADEA and to evaluate its performance, 18 test problems from the UFL benchmark problem library are selected for solution. Then, the computational results of the basic differential evolution algorithm, IADEA and the hybrid ant colony algorithm are compared and analyzed.

4.1. Experimental Environment and Parameter Settings

The basic differential evolution algorithm and IADEA were coded on the Matlab R2017b, and all experiments are conducted on: Intel(R) Core(TM) i7-10875H CPU @2.30 GHz, 16.0 GB RAM, 64-bit Windows10.

To determine the population size N in this paper, we test the IADEA to solve the instance ga250a1 with various population size, ranging from 5 to 60. The total iterations of every population is 100 and the optimal value of this instance is 257,969.

As displayed in Table 1, the increasement in the population size bring better target value, but when the size is bigger than 20, the difference in the target value is slight compared to the difference in the elapsed time, which also increases in multiple-rate due to the larger size of population. Therefore, we decide to let the population = 20 for more efficiency.

The threshold T of the activation function determine the opening of the facility in the algorithm, if $T < 0$, it means more facility will be closed in the iteration, otherwise more facility will open. To applied to all instances, we let the $T = 0$.

The upper bound $P_{\max}$ and the lower bound $P_{\min}$ are related to the coefficient w in the adaptive operator and the random number in the differential step, but the upper limit of random number is 1 in the paper, so the magnitude of the $P_{\max}$ and $P_{\min}$ will not make a difference in the algorithm, and we set the $P_{\max} = P_{\min} = 30$. Note that if $P_{\max} \neq P_{\min}$, the possibility of the opening and closing of the facility will not be the same in the algorithm, which is similar to the threshold T.

According to the discussion above and previous research, the parameters in the IADEA is set as follow: threshold $T = 0$, the upper bound $P_{\max} = 30$, the lower bound $P_{\min} = -30$, $W_{\max} = 0.6$, $w = 0.05$, $W_{\min} = 0.3$.

Table 1. The target values obtained by using different population size N.

Population size N	Elapsed time (s)	Target value
5	6.86 s	263,740
10	13.95 s	262,057
20	28.81	260,921
30	41.76	260,325
40	56.03	259,965
50	70.45	259,870
60	82.81	259,484

The hybrid ant colony algorithm in the literature [12] was conducted on: Intel(R) Core(TM) i7-3770 CPU @3.4 GHz, 3.41 GB RAM, Windows 7, and data processing was done by Matlab2010.

4.2. Computational Results

Table 2 shows the computational results of the basic differential evolution algorithm and IADEA for 18 instances from the UFL benchmark problem library in the same experimental setting, where the number of facilities m and the number of customers n are equal in all problems. The target value refers to the optimal value in the population after 100 iterations and the optimal value indicate the known optimal value. The Gap value is the difference between the target value and the known optimal value, which is given by:

$$\text{Gap} = \frac{\text{target value} - \text{known optimal value}}{\text{known optimal value}} * 100\%$$

Table 2. The comparison of computational results between basic DE and IADEA.

Instance	Size	Optimal value	Basic differential evolution algorithm			IADEA		
			Target value	Elapsed time (s)	Gap (%)	Target value	Elapsed time (s)	Gap (%)
ga250a1	250	257,957	263,230	29.31	2.04	259,397	28.41	0.55
ga250a2	250	257,502	262,463	26.23	1.93	259,540	27.29	0.79
ga250b1	250	276,339	350,531	30.26	26.84	296,989	31.48	7.47
ga250b2	250	275,141	342,790	26.01	24.59	290,505	27.74	5.58
ga500a1	500	511,422	528,265	523.45	3.29	517,836	464.49	1.25
ga500a2	500	511,333	525,639	432.90	2.80	519,643	466.04	1.63
ga500b1	500	538,060	732,604	469.71	36.16	644,440	468.39	19.77
ga500b2	500	537,850	736,714	448.80	36.97	663,517	460.01	23.36
ga750a1	750	763,576	792,588	1589.15	3.80	776,280	1569.61	1.66
ga750a2	750	763,674	791,666	1235.87	3.67	783,083	1715.88	2.15
ga750b1	750	796,480	1,138,657	1897.19	42.96	1,029,122	1673.81	29.21
ga750b2	750	796,056	1,140,646	1863.46	49.36	1,026,189	1899.64	28.91
gs250a1	250	257,964	262,755	26.51	1.86	259,579	29.03	0.70
gs250b1	250	276,761	341,097	26.27	23.25	297,720	28.25	7.57
gs500a1	500	511,229	526,647	459.36	3.02	518,227	465.88	1.37
gs500b1	500	537,931	737,951	454.47	37.18	681,050	470.53	26.61
gs750a1	750	763,671	792,767	2293.45	3.81	781,756	2361.83	2.37
gs750b1	750	797,026	1,140,706	2202.09	43.12	1,016,147	2472.10	27.49

Table 3. The computational result of hybrid ant colony algorithm.

Instance	Optimal value	Hybrid ant colony algorithm		
		Target value	Elapsed time (s)	Gap (%)
ga250a1	257,957	257,989	142.80	0.01
ga250b1	276,339	350,951	144.48	27.00
ga500a1	511,422	520,780	571.65	1.83
ga500b1	538,060	731,632	578.93	35.98
ga750a1	763,576	780,901	1278.58	2.27
ga750b1	796,480	1,076,357	1281.73	35.14
gs250a1	257,964	258,875	142.80	0.35
gs250b1	276,761	351,571	144.48	27.03
gs500a1	511,229	518,357	551.53	1.39
gs500b1	537,931	709,514	542.51	31.90
gs750a1	763,671	782,013	1295.36	2.40
gs750b1	797,026	1,077,446	1334.50	35.18

As can be seen from Table 2, the introduction of the adaptive factor does not make a significant difference in the elapsed time between basic differential algorithm and IADEA for the same number of iterations, but the solution results of the IADEA are substantially better than those of the basic differential evolution algorithm. Mainly because the adaptive factor alleviates the defect that the basic differential evolution algorithm tend to fall into local optimal in the optimization process. Moreover, 12 Gap values of all 18 instances are below 8%, which proves that the adaptive differential algorithm is able to effectively find a satisfactory solution to the problem.

A comparison between Table 2 and Table 3 shows that IADEA outperforms the hybrid ant colony algorithm in all 18 instances except the ga250a1 and gs250a1 with similar or lower elapsed time.

5. Conclusions

This paper improves the basic differential evolution algorithm and proposes a differential evolution algorithm for the UFL problem based on the characteristics of the UFL problem. The proposed approach for solving UFL problem is based on the DE algorithm. To map the variable in the DE algorithm into binary one, we use the activation function to transform variables. Meanwhile, to improve the efficiency of the algorithm, we introduce the adaptive operator to alleviate the issue of being stuck in a local optimum.

It is demonstrated that the IADEA is feasible for solving the UFL problem and significantly improves the optimization efficiency of the basic differential evolution algorithm. At the same time, the IADEA can obtain better target values in similar or shorter time compared with the hybrid ant colony algorithm.

Therefore, this paper provides a new solution to the UFL problem and extends the application of the differential evolution algorithm. Further research is required to improve the convergence speed and operational efficiency of this algorithm.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Zhou, J., Wang, X.Q., Zhang, L.L., Zhou, X., Jing, S.Q. and Liang, G.C. (2022) An Improved Genetic Algorithm for the Uncapacitated Facility Location Problem and Applications in Oil and Gas Fields. *Journal of Physics: Conference Series*, **2224**, Article ID: 012134. https://doi.org/10.1088/1742-6596/2224/1/012134

[2] Tang, Z.P. and Sun, J.P. (2022) Railway Emergency Facility Location Optimization Based on Non-Probabilistic Reliability Theory (in Chinese). *Operations Research and Management Science*, **31**, 100-108.

[3] Singh, K.N. (2008) The Uncapacitated Facility Location Problem: Some Applications in Scheduling and Routing. *International Journal of Operations Research*, **5**, 36-43.

[4] Li, Z.K., Liu, L.D. and Yu, C.C. (2022) The Method for the Inverse Uncapacitated Facility Location Problem under One Norm (in Chinese). *Operations Research and Management Science*, **31**, 86-92.

[5] Alenezy, E.J. (2021) An Integer Programming Formulation of Capacitated Facility Location Problem. *International Journal of Mathematics & Computer Science*, **16**, 1087-1101.

[6] Zhang, H.Z., Chen, Y.T., Ye, Y. and Ni, J. (2020) Lagrangian Wolf Pack Algorithm and Its Application for Solving the Location Problem of Un-Capacitated Facilities (in Chinese). *Journal of Systems & Management*, **29**, 957-963+973

[7] Ling, H.F. (2018) Solving Uncapacitated Facility Location Problems Using Pseudo-Boolean Model and Heuristic Algorithms (in Chinese). *China Mechanical Engineering*, **29**, 2966-2971.

[8] Liu, C.M. and Zhang, H.Z. (2018) Hybrid Bat Algorithm for Un-Capacitated Facility Location Problem (in Chinese). *Computer Engineering and Applications*, **54**, 28-34.

[9] Storn, R. and Price, K (1997) Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, **11**, 341-359. https://doi.org/10.1023/A:1008202821328

[10] Kashan, M.H., Kashan, A.H. and Nahavandi, N. (2013) A Novel Differential Evolution Algorithm for Binary Optimization. *Computational Optimization and Applications*, **55**, 481-513. https://doi.org/10.1007/s10589-012-9521-8

[11] Lewis, B.M., Erera, A.L., Nowak, M.A. and White III, C.C. (2013) Managing Inventory in Global Supply Chains Facing Port-of-Entry Disruption Risks. *Transportation Science*, **47**, 162-180. https://doi.org/10.1287/trsc.1120.0406

[12] Li, Q., Zhang, H.Z. and Cesar, B.-R. (2016) Hybrid Ant Colony Algorithm for the Uncapacitated Facility Location Problem (in Chinese). *Journal of University of Shanghai for Science and Technology*, **38**, 367-372.