An improved genetic algorithm for efficiently solving the uncapacitated facility location problem*

Dilara Ozdemir[1,*,†], Emrullah Sonuc[1,†] and Caner Ozcan[2,†]

[1] Karabuk University, Department of Computer Engineering, 78050 Karabuk, Turkiye
[2] Karabuk University, Department of Software Engineering, 78050 Karabuk, Turkiye

Abstract

The Facility Location Problem (FLP) is an optimization problem that aims to reduce operational costs by strategically locating facilities. The Uncapacitated Facility Location Problem (UFLP) is considered a subset of the FLP. In the UFLP, facility optimization is addressed without including capacity constraints in the calculations. The UFLP problem is frequently used to model and solve problems in areas such as telecommunications, logistics, and emergency services. In this problem, reaching the optimal solution is computationally hard. The exponential growth in problem size causes significant computational costs, making exact solution methods difficult and necessitating better, more efficient algorithms. To address this problem, this study presents an Improved Genetic Algorithm (IGA) approach for solving UFLP instances. The IGA uses problem-specific operators to increase cost and solution quality while employing GA methodology. Furthermore, the proposed method demonstrates good performance across problems of different scales in terms of computational cost and accuracy in reaching the optimal solution. In addition, the proposed IGA demonstrates particularly strong performance on complex problem instances and maintains a high level of solution quality while effectively controlling computational cost. The results prove the effectiveness and practical applicability of the proposed algorithm in real-world UFLP scenarios.

Keywords

combinatorial optimization, uncapacitated facility location problem, genetic algorithms, metaheuristics

1. Introduction

The Facility Location Problem (FLP) is a mathematical optimization problem that aims to determine optimal facility locations considering many operational parameters including transportation costs, travel time, operational expenses, and location expenses [1]. FLP has two main subcategories: Capacitated FLP (CFLP) and Uncapacitated FLP (UFLP). CFLP considers the operational status and service capabilities of the facility, while imposing constraints on production and transportation capacity. In contrast, UFLP operates without capacity constraints, prioritizes customer supply requirements, and focuses on minimizing total costs through optimal customer-facility assignments [2-4].

The importance of UFLP extends to various application areas where it serves as a fundamental optimization criterion, assuming that customer demands can be fully satisfied without capacity constraints [5]. Its practical applications span many areas: logistics and supply chain management [6], telecommunications network design [7], resource allocation and infrastructure development [8]. The methodology is particularly valuable in the location of critical services including emergency response facilities (fire stations, ambulances, police units) [9] and the location optimization of healthcare facilities [10]. This application versatility demonstrates the essential role of UFLP in solving complex location optimization challenges in various industries and utility sectors.

* CMIS'26: The Ninth International Workshop on Computer Modeling and Intelligent Systems, May 5, 2026, Zaporizhzhia, Ukraine
* Corresponding author.
† These authors contributed equally.
ozdemirdilara@outlook.com.tr (D. Ozdemir); esonuc@karabuk.edu.tr (E. Sonuc); canerozcan@karabuk.edu.tr (C. Ozcan)
0000-0003-1304-3053 (D. Ozdemir); 0000-0001-7425-6963 (E. Sonuc); 0000-0002-2854-4005 (C. Ozcan)

1.1. Related work

Due to its NP-hard nature, the UFLP has attracted significant interest from the optimization community, leading to a variety of solution approaches. Ramshani et al [11] proposed metaheuristic approaches to address two-level UFLP and increase robustness in supply chain modeling. They also investigated the effect of uncertainty in FLP and discovered the advantage of methods such as tabu search to solve such problems. As a result, they developed a problem-specific Route Subset Selector (RSS) with two mathematical programming formulations and a tabu search algorithm. Thus, they presented a new solution method for the UFLP problem. Hakli and Ortacay [12] proposed a scatter search algorithm for UFLP. They used crossover techniques to improve the global search capability of this algorithm. They also used mutation operations to improve and optimize the local search capability. In the concluding part of this study, they compared 12 different methods for 15 UFLP instances. The experimental results of the study show that 13 out of 15 proposed methods obtain the best value. Their study shows that the improved scatter search algorithm is an effective and successful method for solving UFLP. Baş and Ülker [13] proposed a binary variant of a metaheuristic called Binary Social Spider Algorithm (BinSSA) to solve UFLP. BinSSA was tested on 15 UFLP instances of small, medium and, large-scale instances. The results were compared with 18 state-of-the-art algorithms, and BinSSA obtained competitive results in solving UFLPs. Sonuç [14] adapted the structure of the Crow Search Algorithm (CSA), designed for continuous optimization, to binary optimization problems. As part of the study, Binary CSA (BinCSA) was developed, which can work efficiently in the binary search space. The study shows that BinCSA is effective for binary optimization problems such as UFLP. Oliveira et al [15] conducted a study to compute two different variants of dynamic or multi-period two-stage UFLP. The study aims to optimize the installation and transportation costs of first-level facilities by using second-level facilities to serve distributed customers with different demand patterns. The proposed solution is compared with CPLEX and its integrated algorithm to show better performance. Kaya [16] proposed the Binary GSO (BinGSO) approach based on the Galactic Swarm Optimization (GSO) method. In addition to this approach, they adopted Binary Artificial Algae Algorithm as the main search algorithm and proposed a new method to solve UFLP. Cinar [17] proposed a binary variant of the Archimedes optimization algorithm for UFLP. The study investigates the effectiveness of these binary variants to solve UFLPs using 17 different transfer functions and the effect of different transfer functions on the solution quality. Alidaee and Wang [18] proposed a method that can solve UFLP by combining a hybrid genetic algorithm (GA) and tabu search. The GA used in the study is based on a random key applied to sorting problems, while tabu search focuses on adaptive critical events. The key component is adapted to modify the search component. The proposed method is compared with benchmarks and a GRASP-based algorithm.

Sonuc and Ozcan [19] proposed an adaptive binary evolutionary algorithm (EA) called Adaptive Binary Parallel EA (ABPEA) to solve UFLP. ABPEA generates solutions using single and binary operators and is effectively applied to UFLP. The study aims to provide a solution that can be modeled by automating operator and parameter selection. The study reveals that ABPEA can effectively adapt to NP-hard problems such as UFLP and perform competitively. Moreover, this study makes an essential contribution to the literature by studying the impact of adaptive optimization methods on complex problems. Zhang et al. [20] proposed Enhanced Group Theory-Based Optimization Algorithm (EGTOA) for UFLP and compare its performance results with 16 different algorithms. They improved the optimization by using Direction Mutation Operator (ODMO), a local search operator, and Redundant Control Strategy (RCS). The study shows that the specially developed EGTOA is a competitive algorithm with better results than the algorithms presented in the literature.

Jang and Zhang [21] proposed the Adaptive Differential EA (IADEA) for solving UFLP. IADEA integrates the activation function into the algorithm by considering the critical features of UFLP. Moreover, an adaptive operator is added to extend the local optimum. The proposed IADEA is tested on UFLP benchmark problems and compared with a hybrid ant colony algorithm. Ozsoydan

and Kasırga [22] applied the Flower Pollination Algorithm (FPA) to binary optimization problems such as UFLP by enhancing it with evolutionary operators. The developed method used evolutionary operators such as crossover and mutation to adapt the FPA for binary optimization. The study shows that the developed method can solve existing benchmarks and outperform other heuristic algorithms, an essential step for binary optimization problems. Yang and Luo [23] focused on solving the k-product UFLP problem given a set of customer points in an environment where k-different products are supplied, and facilities can be opened. In the work, they focus on two different integer programs describing UFLP and use the (2k+1)-approximation algorithm for the solution. Ozkıs and Karakoyun [24] proposed the binary Enhanced MFO Desert Bush (binEMFO-DB) algorithm, which is a modified version of Moth Flame Optimization (MFO) for solving UFLP. Three main modifications to the proposed algorithm were considered and tested on 15 different UFL problems. In their study, Taguchi linear array design was used for parameter analysis. Soufyane et al. [25] presented a method that uses artificial neural networks and clustering algorithms to solve the UFLP problem. The technique uses Maximum Stable Set Problem (MSSP) and Continuous Hopfield Network (CHN) to determine the optimal locations for the facilities. The result of the study shows that the proposed method for solving UFLP performs well in terms of solution quality and time.

Bas and Yildizdan [26] presented BinAOA, a binary form of the Arithmetic Optimization Algorithm (AOA). They adapted the BinAOA method to binary optimization problems. They also proposed the BinAOAX method by improving the candidate solution generation process of BinAOA with the XOR logical gate. The study also performs performance tests on classical benchmark functions and shows that BinAOA and BinAOAX methods are successful in UFLP. Aslan and Pavone [27] presented the Modified Binary Vortex (MBVS) method, a binary version of the Vortex Search (VS) algorithm. MBVS makes three essential modifications to the basic structure of the VS algorithm, resulting in significant improvements in the solution of UFLP instances. The proposed method is tested on 15 different UFLP instances and shows a significant performance compared to other works presented in the literature. In another study, Pratiwi et al. [28] also used FPA to solve UFLP. In addition to FPA, they also applied the Artificial Tree (AT) algorithm, which is inspired by the life cycle of plants, to solve UFLP. According to the experimental results, both algorithms perform highly on the UFLP instances. Gendron et al. [29] studied industrial applications arising in two-level UFLPs with single assignment constraints (TUFLP-S). They present an integer programming formulation for a modular cost variant of TUFLP-S and adapt the variable neighborhood search algorithm for this classical problem. Matos [30] used the Relaxation Adaptive Memory Programming (RAMP) approach with different levels to solve UFLP. The study tests more straightforward and more complex versions of the RAMP approach on UFLP. The literature shows a trend toward hybrid and nature-inspired algorithms, with a focus on adaptive mechanisms and binary optimization. While these methods have shown good results on standard problems, there is still room for improvement in solution quality, efficiency, and real-world applicability.

2. Problem statement

UFLP aims to determine the optimal facility layout and customer-facility assignment plan to effectively meet customer demands, whereby the locations of facilities to be opened are strategically identified and each customer's demand is allocated to a designated facility. In its application, UFLP simultaneously addresses three core objectives: determining the locations of facilities to be established, assigning each customer's demand to the most appropriate facility, and minimizing the aggregate cost comprising both facility-opening costs and transportation costs between facilities and customers.

UFLP minimizes the total cost associated with opening a facility, assigning customers, and transportation can be formulated as follows:

$$Minimize \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} + x_{ij} + \sum_{i=1}^{n} f_i y_i \quad (1)$$

where:

- c_{ij} is the transportation cost from facility i to customer j. (Distance)
- f_i is the cost of opening facility i.

The constraints of the mathematical model are stated as follows:

- Each customer is to be served by a single facility:

$$Minimize \sum_{i=1}^{n} x_{ij} = 1, j = 1, 2, \ldots, m \quad (2)$$

- Only open facilities are eligible to serve customers:

$$x_{ij} - y_i \leq 0, i = 1, 2, \ldots, n \text{ and } y = 1, 2, \ldots, m \quad (3)$$

- The decision variables x_{ij} and y_i must be binary:

$$x_{ij} \in \{0,1\}, i = 1, 2, \ldots, n \text{ and } y = 1, 2, \ldots, m$$
$$y_i \in \{0,1\}, i = 1, 2, \ldots, n \quad (4)$$

where:

- $x_{ij} = 1$ if the customer (j) is served by the facility i and 0 otherwise.
- $y_i = 1$ if the facility (i) is opened and 0 otherwise.

3. Materials and methods

3.1. Genetic algorithm

GA is an optimization method inspired by natural selection and the principles of genetics. GA performs by encoding possible solutions to optimization problems, such as chromosomes. The declaration of these chromosomes is realized as a binary series. The initial step of the algorithm is an initial population of chromosomes to represent each potential solution. GA aims to develop the population to reach possible solutions through crossover, selection, and mutation processes. Mechanisms such as roulette wheel selection, tournament selection, and stochastic universal sampling are used for the selection process. The selection mechanism aims to select more optimal solutions within the population so that the individuals produce better descendants that are more likely to be optimal. Crossover operations are used to select individuals in pairs to produce descendants. In this way, genetic elements are mixed, and possibilities closer to the possible solution are discovered. Random changes are made in some individuals with the mutation used in GA. These changes create genetic diversity in the population and ensure that the algorithm does not stay at the local optimum and reaches the global optimum. Random changes are made in some individuals with the mutation used in GA. Genetic diversity is created within the population, and the algorithm can reach the global optimum by not staying in the local optimum region. This process in GA continues until a certain criterion is reached. In other words, it stops when the solutions approach or reach the global optimum. GAs can achieve effective results for complex and high-dimensional optimization problems such as UFLP. When traditional methods are used to solve

complex problems, cost and performance losses may occur. GAs can investigate different regions in the problem space and produce optimal or near-optimal solutions thanks to their many elements, which are modeled from genetics.

3.2. The proposed method (IGA)

The proposed IGA incorporates modifications designed to improve solution quality while keeping computational load manageable by applying GA methodology. In the algorithm, each candidate solution is represented as a binary chromosome indicating whether a facility is opened or closed. In other words, every chromosome corresponds to a specific configuration of open facilities.

IGA first creates a random population of candidate solutions. Each chromosome is represented as a binary vector with randomly assigned states. A possible solution must contain at least one active facility. Therefore, open facilities are added to chromosomes that do not contain active facilities at random. With this adjustment, the search process can begin with feasible solutions while preserving the diversity of the initial population. Parent solutions are selected using tournament selection. In this strategy, a small random subset of the population is chosen. The individual with the best fitness value in this subset is selected as the parent. Tournament selection allows stronger individuals to be determined more frequently while still giving weaker individuals a chance to participate in the evolutionary process, helping maintain population diversity.

To generate offspring, a cost-guided crossover operator is used. The operator incorporates facility opening costs into crossover decisions. Firstly, facility opening costs are normalized to create comparative values:

$$f_i = y_i \frac{f_i - f_{min}}{f_{max} - f_{min}} \tag{5}$$

where f_i denotes the opening cost of facility i, and f_{min} and f_{max} represent the minimum and maximum facility opening costs in the instance. When two parent chromosomes differ at a particular gene position i, the probability of opening that facility in the offspring is determined using the following rule:

$$p_{open} = 0.6 + 0.3(1 - f_i) \tag{6}$$

This formulation ensures diversity while increasing the probability of selecting facilities with lower opening costs. During the crossover step, two offspring are produced. One of the offspring is biased toward the better parent, thus ensuring the selection of better solutions. The other provides additional randomness to make the exploration process in the search area more sustainable. After crossover, mutation is applied using a bit-flip mechanism. In this operation, each gene may change its state with a small probability. While preserving population diversity through mutation, the risk of early convergence is reduced.

In addition to the standard GA operators, the IGA applies a probabilistic greedy repair mechanism. This mechanism aims to improve solution quality by removing facilities that unnecessarily increase the total cost. In the repair approach, all open facilities in a chromosome are first identified and sorted according to their opening costs. The most expensive facilities are then considered as repair candidates. Each candidate facility is temporarily removed, and the resulting solution is evaluated. If removing the facility improves the objective value, the modification is accepted. To avoid high computational cost, the repair procedure is applied only with a predefined probability and evaluates only a limited number of candidate facilities.

An elitism strategy is also applied during the evolutionary process. The best individuals in the population are preserved and transferred directly to the next generation. This mechanism ensures that high-quality solutions discovered during the search are not lost. The evolutionary process

continues over a fixed, defined number of generations. At the end of the algorithm, the chromosome with the lowest objective value in the population is selected as the final solution.

4. Experimental results

The performance of the proposed algorithm has been evaluated on commonly used benchmark problems for UFLP [31]. The IGA's performance was evaluated on cap71–cap74, cap101–cap104, cap131–cap134, and larger-scale capa, capb, and capc instances. These instances are frequently used in literature to evaluate UFLP solutions. Because they present different levels of problem size and complexity, algorithms can be tested under different conditions. However, the performance of the approaches also allows for evaluation in terms of solution quality and computational efficiency.

Table 1
Computational Results of the Proposed IGA on the UFLP Instances

Instance	Optimal	Best	Mean	Std. Dev.	Gap	Hit	Time (s)
cap71	932,615.75	932,615.75	932,615.75	1.16e-10	0.00	10/10	5.20
cap72	977,799.40	977,799.40	977,799.40	0.00	0.00	10/10	5.24
cap73	1,010,641.45	1,010,641.45	1,010,641.45	1.16e-10	0.00	10/10	5.41
cap74	1,034,976.98	1,034,976.98	1,034,976.98	1.16e-10	0.00	10/10	5.32
cap101	796,648.44	796,648.44	796,734.47	258.09	0.11	9/10	5.65
cap102	854,704.20	854,704.20	854,704.20	1.16e-10	0.00	10/10	5.61
cap103	893,782.11	893,782.11	893,884.02	305.71	0.11	9/10	5.69
cap104	928,941.75	928,941.75	928,941.75	0.00	0.00	10/10	5.72
cap131	793,439.56	793,439.56	793,525.59	258.09	0.11	9/10	7.04
cap132	851,495.33	851,495.33	851,495.33	1.16e-10	0.00	10/10	7.19
cap133	893,076.71	893,076.71	893,429.41	352.70	0.08	5/10	7.26
cap134	928,941.75	928,941.75	928,941.75	0.00	0.00	10/10	7.20
capa	17,156.454.48	17,156,454.48	17,156,454.48	0.00	0.00	10/10	11.40
capb	12,979.071.58	12,979,071.58	13,015,336.87	42,497.10	0.79	5/10	11.61
capc	11,505,594.33	11,505,594.33	11,525,263.36	14,999.77	0.40	1/10	11.79

To evaluate the performance of the proposed algorithm, the algorithm was run 10 times on each instance. Experiments and tests were conducted on an Intel CPU with 16 GB of RAM. All experiments were performed with the same parameter configuration defined. The initial population size was set to 100 individuals, and the algorithm was run for 2000 generations. The mutation rate was fixed at 0.02. The elitism strategy was applied by retaining the top 10 individuals in each generation. The tournament size was selected as 3 for the parent selection step.

In addition to standard GA operators, a probabilistic greedy repair mechanism has been added to the proposed algorithm. A probability of 0.05 is considered for the repair process. To control the computational cost during the repair phase, only a limited number of candidates are evaluated. The algorithm evaluates the five most expensive open facilities as repair candidates. In each repair attempt, the removal operation is performed once based on success.

For consistency in statistical results, each instance was solved independently 10 times with different random seeds. The IGA's performance was evaluated using the best solution, mean solution value, standard deviation, gap, and time metrics as shown in Table 1. When the results were examined, it was observed that the IGA generally achieved optimal solutions in the test instances. In particular, the most suitable solutions were obtained in all runs for instances such as cap71, cap72, cap73, cap74, cap102, cap104, cap132, cap134, and capa. An examination of the hit and gap values in these instances reveals that the convergence value is satisfactory.

Table 2
Performance Comparison of EA, GA, and the Proposed IGA for the UFLP

Instance	Optimal	EA	GA	IGA
cap71	932,615.75	117,691,32.30	944,057.39	932,615.75
cap72	977,799.40	977,799.40	995,217.83	977,799.40
cap73	1,010,641.45	1,014,679.64	1,010,641.45	1,010,641.45
cap74	1,034,976.98	1,034,976.98	1,064,662.28	1,034,976.98
cap101	796,648.44	808,212.06	796,648.44	796,648.44
cap102	854,704.20	872,620.05	854,704.20	854,704.20
cap103	893,782.11	893,782.11	893,782.11	893,782.11
cap104	928,941.75	928,941.75	928,941.75	928,941.75
cap131	793,439.56	793,439.56	793,439.56	793,439.56
cap132	851,495.33	851,495.33	851,495.33	851,495.33
cap133	893,076.71	893,076.71	893,076.71	893,076.71
cap134	928,941.75	928,941.75	928,941.75	928,941.75
capa	17,156.454.48	17,796,978.56	17,156,454.48	17,156,454.48
capb	12,979.071.58	12,991,528.78	12,979,071.58	12,979,071.58
capc	11,505,594.33	11,722,193.11	11,505,594.33	11,505,594.33

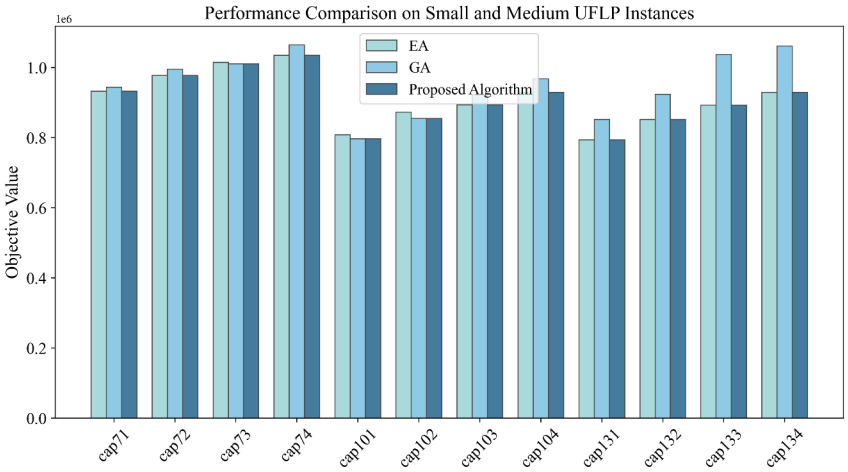
The IGA has been observed to produce solutions very close to optimal values with minor deviations between runs on large and complex instances. This demonstrates that the algorithm can produce reliable solutions even when the search space becomes more complex on large and complex instances. The generally low standard deviation values prove that the algorithm produces consistent solutions across multiple independent runs.

The average execution time also varies depending on the complexity and size of the example. The IGA reaches the optimal solution in approximately 5–6 seconds for smaller instances such as

cap71–cap104, whereas takes approximately 11 seconds for larger instances such as capa, capb, and capc.

Table 2 compares the performance of EA, classical GA, and the proposed IGA. In UFLP instances, the proposed approach reaches the optimal solution, while classical GA and EA produce more costly solutions. Classical GA is seen to produce worse results than other methods, especially in more complex instances such as capa, capb, and capc. Although EA shows competitive performance in some instances, it consistently fails to reach optimal solutions compared to the proposed algorithm. These results show that the proposed improved GA can reach more reliable and accurate solutions than both EA and classical GA.

Figure 1:



Performance comparison of EA, GA and IGA on small and medium UFLP benchmark instances.

Figures 1 and 2 present the performance differences between EA, classical GA, and IGA on UFLP benchmark instances. Figure 1 shows the results on small and medium-sized instances. Examining the values in the graph, it can be seen that in some instances, classical GA presents higher-cost solutions, while EA achieves competitive results. However, it is observed that both methods show lower performance compared to the proposed method.

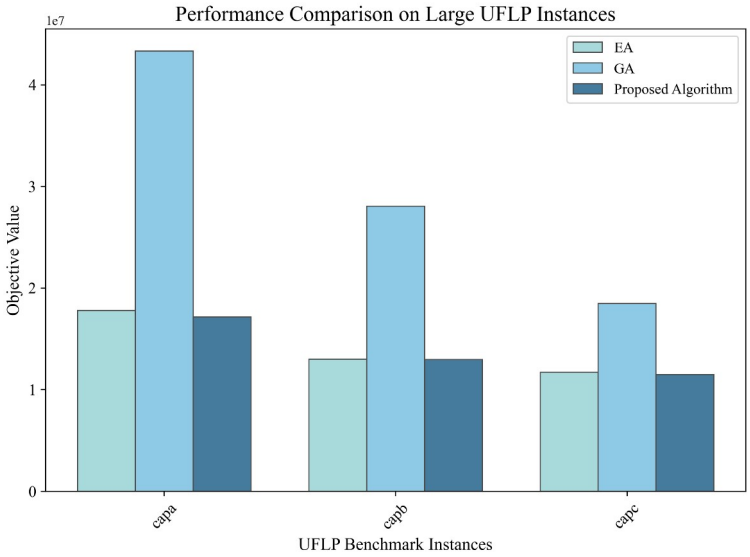


Figure 2: Performance comparison of EA, GA and the proposed algorithm on large-scale UFLP benchmark instances.

Figure 2 compares the performance of the algorithms on complex and large-scale instances. Classical GA struggles to reach optimal solutions in instances such as cap and capb. While EA performs well in some instances, it shows lower performance compared to the proposed algorithm.

The results show that despite the increasing size and complexity of the problem, the proposed algorithm can reach the optimal solution and produce reliable results.

5. Conclusion

This study analyzes the literature review on the improved genetic algorithm (IGA) and its experimental performance in UFLP. The experimental results presented in this study demonstrate that IGA obtains optimal solutions for various UFLP instances that are nearly identical to the best solutions reported in the literature. Overall, the results confirm that the proposed IGA is a reliable and efficient approach for solving the UFLP problem. This consistently produces optimal solutions, particularly in complex and large-scale problem instances. Thus, this study demonstrates the feasibility and performance of the proposed approach in providing more efficient solutions in optimization processes. Future research should include more comprehensive experimental studies of different problem sizes and characteristics, with a focus on improving the algorithm for practical applications.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: Grammar and spelling check. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] R.Z. Farahani, M. Hekmatfar, B. Fahimnia, N. Kazemzadeh, Hierarchical facility location problem: Models, classifications, techniques, and applications, Comput. Ind. Eng. 68 (2014) 104 –117. doi:10.1016/j.cie.2013.12.005.

[2] Z. Drezner, H.W. Hamacher (Eds.), Facility Location: Applications and Theory, Springer, New York, (2004). doi:10.1007/978-3-642-56082-8.

[3] A. Klose, A. Drexl, Facility location models for distribution system design, Eur. J. Oper. Res. 162 (2005) 4–29. doi:10.1016/j.ejor.2003.10.031.

[4] A.R. Conn, G. Cornuéjols. A projection method for the uncapacitated facility location problem. Mathematical Programming 46 (1990), 273–298. doi:10.1007/bf0158574.

[5] V. Verter. Uncapacitated and capacitated facility location problems. In: Foundations of Location Analysis. Springer, New York, NY, (2011), pp. 25–37. doi:10.1007/978-1-4419-7572-0_2

[6] G.P. Mendoza-Ortega, M. Soto, J. Ruiz-Meza, R. Salgado, A. Torregroza, Scenario-based model for the location of multiple uncapacitated facilities: Case study in an agro-food supply chain, in: Applied Computer Sciences in Engineering, Springer, Cham, (2021), pp. 386–398. doi:10.1007/978-3-030-86702-7_33.

[7] H. Yaman, A survey on location problems with applications in telecommunications, in: Concentrator Location in Telecommunications Networks, Springer, Boston, (2005), pp. 9–23. doi:10.1007/0-387-23532-9_2.

[8] J. Zhou, X. Wang, L. Zhang, X. Zhou, S. Jing, G. Liang, An improved genetic algorithm for the uncapacitated facility location problem and applications in oil and gas fields, J. Phys. Conf. Ser. 2224 (2022), 012134. doi:10.1088/1742-6596/2224/1/012134.

[9] B. Zhang, J. Peng, S. Li, Covering location problem of emergency service facilities in an uncertain environment, Appl. Math. Model. 51 (2017), 429–447. doi:10.1016/j.apm.2017.06.043.

[10] A. Ghaderi, M.S. Jabalameli, Modeling the budget-constrained dynamic uncapacitated facility location–network design problem and solving it via two efficient heuristics: A case study of health care, Math. Comput. Model. 57 (2013), 382–400. doi:10.1016/j.mcm.2012.06.017.

[11] M. Ramshani, J. Ostrowski, K. Zhang, X. Li, Two level uncapacitated facility location problem with disruptions, Comput. Ind. Eng. 137 (2019), 106089. doi:10.1016/j.cie.2019.106089.

[12] H. Hakli, Z. Ortacay, An improved scatter search algorithm for the uncapacitated facility location problem, Comput. Ind. Eng. 135 (2019), 855–867. doi:10.1016/j.cie.2019.06.060.
[13] E. Baş, E. Ülker, A binary social spider algorithm for uncapacitated facility location problem, Expert Syst. Appl. 161 (2020), 113618. doi:10.1016/j.eswa.2020.113618.
[14] E. Sonuç, Binary crow search algorithm for the uncapacitated facility location problem, Neural Comput. Appl. 33 (2021), 14669–14685. doi:10.1007/s00521-021-06107-2.
[15] P.B. de Oliveira, R.S. de Camargo, G. de Miranda Jr., A.X. Martins, A computational study of a decomposition approach for the dynamic two-level uncapacitated facility location problem, Comput. Ind. Eng. 151 (2021), 106964. doi:10.1016/j.cie.2020.106964.
[16] E. Kaya, BinGSO: Galactic swarm optimization powered by binary artificial algae algorithm for solving uncapacitated facility location problems, Neural Comput. Appl. 34 (2022), 11063–11082. doi:10.1007/s00521-022-07058-y.
[17] A.C. Çınar, A comprehensive comparison of binary Archimedes optimization algorithms on uncapacitated facility location problems, Duzce Univ. J. Sci. Technol. 10 (2022), 27–38. doi:10.29130/dubited.876284.
[18] B. Alidaee, H. Wang, Uncapacitated facility location problem: A hybrid genetic–tabu search approach, IFAC-PapersOnLine 55 (2022), 1619–1624. doi:10.1016/j.ifacol.2022.09.622.
[19] E. Sonuç, E. Özcan, An adaptive parallel evolutionary algorithm for solving the uncapacitated facility location problem, Expert Syst. Appl. 224 (2023), 119956. doi:10.1016/j.eswa.2023.119956.
[20] F. Zhang, Y. He, H. Ouyang, W. Li, A fast and efficient discrete evolutionary algorithm for the uncapacitated facility location problem, Expert Syst. Appl. 213 (2023), 118978. doi:10.1016/j.eswa.2022.118978.
[21] N. Jiang, H. Zhang, Improved adaptive differential evolution algorithm for the uncapacitated facility location problem, Open J. Appl. Sci. 13 (2023), 685–695. doi:10.4236/ojapps.2023.135054.
[22] F.B. Ozsoydan, A.E. Kasırga, Evolution inspired binary flower pollination for the uncapacitated facility location problem, Neural Comput. Appl. (2024), 1–14. doi:10.1007/s00521-024-09684-0.
[23] P.J. Yang, W.C. Luo, Approximation algorithm for the k-product uncapacitated facility location problem with penalties, J. Oper. Res. Soc. China (2023), 1–10. doi:10.1007/s40305-023-00478-0.
[24] A. Özkış, M. Karakoyun, A binary enhanced moth flame optimization algorithm for uncapacitated facility location problems, Pamukkale Univ. J. Eng. Sci. 29 (2023), 737–751. doi:10.5505/pajes.2023.49576.
[25] M. Soufyane, L. Chakir, B. Jaouad, New approach to solve uncapacitated facility location problem using continuous Hopfield network, in: Proc. ICOA, IEEE, (2023), pp. 1–9. doi:10.1109/icoa58279.2023.10308818.
[26] E. Baş, G. Yıldızdan, A new binary arithmetic optimization algorithm for uncapacitated facility location problem, (2022). doi:10.21203/rs.3.rs-2088938/v1.
[27] M. Aslan, M. Pavone, MBVS: A modified binary vortex search algorithm for solving uncapacitated facility location problem, Neural Comput. Appl. 36 (2024), 2573–2595. doi:10.1007/s00521-023-09190-9.
[28] A.B. Pratiwi, R. Pamungkas, H. Suprajitno, Plants inspired algorithms for uncapacitated facility location problems, in: AIP Conf. Proc. 2264 (2020), 020009. doi:10.1063/5.0023481.
[29] B. Gendron, P.-V. Khuong, F. Semet, Models and methods for two-level facility location problems, in: Combinatorial Optimization and Applications: A Tribute to Bernard Gendron, Springer Nature Switzerland, Cham, (2024), pp. 59–75. doi:10.1007/978-3-031-57603-4_4.
[30] T. Matos, RAMP experiments in solving the uncapacitated facility location problem, Ann. Math. Artif. Intell. (2024), 1–20. doi:10.1007/s10472-023-09920-8.
[31] J.E. Beasley, OR-library: Distributing test problems by electronic mail, J. Oper. Res. Soc. 41 (1990), 1069–1072. doi:10.1057/jors.1990.166.