Computer and Decision Making
An International Journal

Journal homepage: www.comdem.org
eISSN: 3008-1416

A Hybrid Genetic Algorithm and Simulated Annealing Approach for the Uncapacitated Facility Location Problem

Songül Kısaboyun[1], Emrullah Sonuç[1,*]

[1] Department of Computer Engineering, Karabük University, Karabük, 78050, Türkiye

ARTICLE INFO

Article history:
Received 25 February 2025
Received in revised form 27 March 2025
Accepted 29 March 2025
Available online 30 March 2025

Keywords:
Genetic A lgorithm; H ybrid Metaheuristics; Uncapacitated Facility Location Problem; Simulated Annealing.

ABSTRACT

The Uncapacitated Facility Location Problem (UFLP), a well-known NP-hard combinatorial optimization problem, aims to minimize the costs associated with opening facilities and servicing customers. This study proposes a hybrid metaheuristic algorithm integrating Genetic Algorithm (GA) and Simulated Annealing (SA) to address UFLP. The proposed approach integrates the global exploration capabilities of GA with the local search effectiveness of SA to achieve robust optimization performance. By combining elite preservation, uniform crossover, and a systematic local search mechanism, the proposed algorithm effectively balances exploration and exploitation, allowing it to escape local optima while efficiently exploring large solution spaces. Extensive computational experiments were conducted on 15 different UFLP benchmark instances show that the hybrid algorithm consistently achieves optimal solutions for small and medium-sized problems. For larger instances (capa, capb, capc), the algorithm maintains competitive performance with minimal gaps from optimal values. Comparative analysis with other optimization algorithms shows that the proposed hybrid approach provides superior solution quality, especially for large instances. The effectiveness of the method is highlighted by its ability to achieve smaller gaps with more consistent solutions compared to existing algorithms, making it a promising approach for complex binary optimization problems.

1. Introduction

Optimization problems are generally divided into two main categories: continuous and discrete [25]. Within discrete optimization problems, binary optimization problems are of particular importance. Such problems have decision variables that can only take the values 0 and 1, and are commonly encountered in classical problems such as the 0/1 knapsack problem [11], the feature selection problem [7], the set-union knapsack problem [24] and the facility location problem [22]. Binary optimization problems are often tackled using metaheuristic algorithms due to their complexity and the need to effectively guide the search in the solution space [23].

[*] *Corresponding author.*
E-mail address: esonuc@karabuk.edu.tr

https://doi.org/10.59543/comdem.v2i.13781

Metaheuristic algorithms are one of the optimization methods that are often used at points where traditional methods fail or are insufficient. Today, the need to provide maximum benefit with minimum resource usage has made the use of metaheuristic algorithms widespread in many fields such as natural and social sciences, physics, mathematics, engineering, and economics [6].

Binary Optimization Problem (BOP) is defined as a binary-based problem space that represents an important class of optimization problems. In a continuous optimization problem, the variables in the search space take continuous values, while in a binary optimization problem, the variables in the search space take the values [0,1]. The value '0' represents the absence of a situation regarding the operation in question, while the value '1' represents its presence. While some algorithms have the ability to solve problems by operating on continuous search spaces, some problems have the ability to solve problems by updating the values [0, 1] on discrete search spaces [13].

The Facility Location Problem (FLP) is a well-known BOP involving location problems for public services (dormitories, schools, libraries, police stations, hospitals, etc.) and private business facilities (markets, stores, warehouses, etc.). These problems are usually considered as no capacity, which means that facilities usually have unlimited capacity to meet even large demands. This variant of FLP is called Uncapacitated FLP (UFLP), which is widely used as a test problem in binary optimization field [9].

The main objective in UFLP is to minimize the total cost, which consists of the cost of establishing the facilities and the service provided by the open facilities to the customers [23]. Facility location is one of the most important stages that directly affects the success of the supply chain. The distance of the facility to be established from the regions it will serve is an important criterion to be evaluated by the decision maker. Because in facility design and layout problems, when a decision is made to establish a facility in a region, the aim is to provide service to as many people as possible [22]. In addition, the distance between the facility location and the service area is expected to be minimum. In other words, facilities should be located as close as possible to demand centers, or service areas should be arranged so that service is provided by the nearest facility [3]. In general, facility location planning aims to minimize the distance between the weighted demand point and the region to be served, while also taking into account customer expectations. FLPs are applied in various areas such as hospitals, blood banks, bank branches, emergency service points such as police/fire stations, main distribution bases, warehouse and logistics centers, communication centers and the positioning of energy networks [1, 26]. Various factors such as demand, supply, distance and cost are important factors that affect facility location decisions [2]. These decisions may change over time; new facilities may be opened, some existing facilities may be closed, facilities may be combined, some facilities may be relocated or their capacity may be expanded.

The contribution of this study is the designing of a hybrid algorithm combining genetic algorithm and simulated annealing for UFLP. The proposed algorithm combines the ability of the genetic algorithm to efficiently explore a large solution space with the ability of the simulated annealing algorithm to prevent local optima stuck, thus providing faster and higher quality solutions. The effectiveness of the hybrid approach is tested on 15 different problem instances and the results are compared with other methods in the literature. In particular, the algorithm regularly achieves optimal results on small and medium-sized problems and shows competitive performance even on large-scale problems. Experimental studies show that the proposed method is superior in terms of solution quality, consistency, and stability. The rest of this paper is organized as follows. In the following section, the mathematical formulation of UFLP is described. The Related Work section briefly discusses previous methods in the literature related to UFLP. In the Proposed Method section, the design process of the hybrid algorithm, the combination of methods, and the implementation steps are explained. In the Experimental Results section, the results obtained by testing the algorithm on 15 different problem instances are presented and compared with other methods. Finally, in the Conclusion section, a general

evaluation of the study is made and suggestions for future studies are presented.

2. Mathematical Formulation of UFLP

UFLP is a combinatorial optimization problem first proposed by Kuehn and Hamburger in 1963, and is one of the most important NP-hard problems in location theory [18]. The mathematical formulation of the UFLP, assuming that n is the number of facilities and m is the number of customers, is as follows [12]:

$$min \quad \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} + \sum_{i=1}^{n} f_i y_i \tag{1}$$

s.t.

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, ..., m \tag{2}$$

$$x_{ij} - y_i \leq 0, \quad i = 1, ..., n, \quad j = 1, ..., m \tag{3}$$

$$x_{ij} \in \{0, 1\}, \quad i = 1, ..., n, \quad j = 1, ..., m \tag{4}$$

$$y_i \in \{0, 1\}, \quad i = 1, ..., n \tag{5}$$

where c_{ij} represents the service cost for facility i to meet the demand of customer j, while f_i denotes the cost of opening a facility at location i. The aim of the UFLP is to minimize the total cost, which comprises the service cost and the facility opening cost, corresponding to the first and second terms in Equation (1), respectively. Constraint (2) states that each customer must be served by exactly one facility. Constraint (3) ensures that customers cannot receive service from a facility that is not operational. Constraints (4) and (5) specify that the decision variables are binary, meaning they can only take values of 0 or 1 [23].

3. Related Work

UFLP is a well-known NP-hard optimization problem [10] that has been studied extensively in the field of metaheuristics and optimization algorithms. Recent studies have proposed several new approaches to improve the solution quality and computational efficiency.

Evolutionary algorithms with unique modifications have been introduced in several studies. An enhanced group theory-based optimization algorithm was proposed [26], which incorporates a one direction mutation operator and a redundant checking strategy. Superior results in both solution quality and speed were demonstrated when compared to existing algorithms. An adaptive binary parallel evolutionary algorithm was developed [23], using reinforcement learning was employed to dynamically select operators, and significant acceleration in solution times was achieved. An improved adaptive differential evolution algorithm was introduced [14], outperforming traditional methods in convergence speed and optimization performance.

Binary optimization techniques have been extensively explored in recent research. A binary social spider algorithm was developed [4], in which similarity measure and logic gate techniques were incorporated. A binary crow search algorithm was presented [22], which used bitwise operations to search directly in binary space and achieved optimal solutions for most benchmark instances. The BinGSO method was proposed [15], in which binary galactic swarm optimization was combined with

the binary artificial algae algorithm, and exploration and exploitation were effectively balanced. A binary enhanced moth flame optimization algorithm was introduced [19], with competitive performance demonstrated for various problem sizes. A binary arithmetic optimization algorithm was introduced [5], through which promising results were shown. The flower pollination algorithm was modified [20], demonstrating robustness and adaptability.

Hybrid approaches combining metaheuristics with simulation techniques have been investigated. A simheuristic algorithm was proposed [21], by which deterministic solutions were outperformed. A comprehensive comparison of binary variants of the archimedes optimization algorithm was conducted [8], where better solutions were yielded by non-stationary transfer functions. Finally, a modified binary vortex search algorithm was presented [3], which demonstrated robust performance over several UFLP instances.

4. The Proposed Hybrid Metaheuristic Algorithm

This section describes the proposed hybrid metaheuristic algorithm that combines genetic algorithm (GA) and simulated annealing (SA) to solve UFLP. The algorithm utilizes the global exploration capabilities of GA with the local search effectiveness of SA to achieve robust optimization performance. The overall workflow of the hybrid algorithm is presented in Algorithm 1.

Algorithm 1 Hybrid Algorithm

Require: F: Facility costs, S: Service costs, N: Population size, G: Max generations, T_0: Initial temperature, α: Cooling rate
Ensure: Best solution and its cost

Initialize population P with N random solutions
$S^* \leftarrow \emptyset$
$C^* \leftarrow \infty$
for $g = 1$ to G **do**
    Evaluate P by sorting based on cost function $C(s, F, S)$
    $S_g \leftarrow P_1$
    $C_g \leftarrow C(S_g, F, S)$
    if $C_g < C^*$ **then**
        $S^* \leftarrow S_g$
        $C^* \leftarrow C_g$
    end if
    Select top 10 solutions as P_{new}
    while $|P_{new}| < N$ **do**
        Select two parents s_1, s_2 from top 20 solutions
        Generate child s_c using uniform crossover
        Apply simulated annealing on s_c with T_0, α
        Apply local search on s_c
        Add s_c to P_{new}
    end while
    $P \leftarrow P_{new}$
end for
return S^*, C^*

The algorithm begins by initializing a population of N random candidate solutions, where each solution represents a potential configuration of facility locations. Each solution is encoded as a binary

string where '1' indicates an open facility and '0' indicates a closed facility. The algorithm requires several key parameters including population size N, maximum number of generations G, initial temperature T_0, cooling rate α, facility opening costs F, and service costs S between facilities and customers.

The main optimization process operates through successive generations, with each generation involving multiple integrated phases. Initially, the algorithm evaluates each solution in the population using the cost function $C(s, F, S)$ that calculates the total cost including facility opening costs and service costs as defined in Equation (1). The solutions are then sorted based on their costs, and the best solution in the current generation (S_g) is compared with the overall best solution (S^*). To maintain elite solutions, the top 10 solutions are automatically preserved for the next generation.

The generation of new solutions continues until the population size N is reached. Parent selection is performed from the top 20 solutions, and uniform crossover is applied to generate child solutions. Each newly generated solution then undergoes a SA optimization phase, as detailed in Algorithm 2. During this phase, a neighbor solution is generated by flipping a randomly selected facility state. The cost difference ΔC between the current and the neighbor solution determines the acceptance criteria: better solutions ($\Delta C < 0$) are always accepted, while worse solutions are accepted with probability $\exp(-\Delta C/T)$. The temperature T is gradually reduced according to cooling rate α, allowing the algorithm to escape local optima by occasionally accepting worse solutions.

Algorithm 2 Simulated Annealing

Require: s: Initial solution, F: Facility costs, S: Service costs, T_0: Initial temperature, α: Cooling rate
Ensure: Optimized solution and its cost

$s_{current} \leftarrow s$
$C_{current} \leftarrow C(s, F, S)$
$T \leftarrow T_0$
while $T > 1$ **do**
    Generate neighbor solution $s_{neighbor}$ by flipping a random bit in $s_{current}$
    $C_{neighbor} \leftarrow C(s_{neighbor}, F, S)$
    $\Delta C \leftarrow C_{neighbor} - C_{current}$
    if $\Delta C < 0$ or $\exp(-\Delta C/T) >$ random$(0, 1)$ **then**
        $s_{current} \leftarrow s_{neighbor}$
        $C_{current} \leftarrow C_{neighbor}$
    end if
    $T \leftarrow T \times \alpha$
end while
return $s_{current}, C_{current}$

Following the SA phase, each solution undergoes a local search refinement process, implemented as shown in Algorithm 3. This systematic evaluation considers flipping each bit in the solution and accepts improvements that reduce the total cost, ensuring that the solution is locally optimal before proceeding to the next generation. This comprehensive approach combines the advantages of multiple optimization strategies: GA provides diverse exploration of the solution space, elite preservation maintains good solutions, SA allows escape from local optima, and local search ensures solution refinement.

The algorithm terminates after G generations, or when no significant improvement is observed in the best solution for a specified number of generations. The final output includes both the best solution found (S^*) and its corresponding cost (C^*). The effectiveness of this hybrid approach lies in its ability to balance exploration and exploitation by combining the global search capabilities of GA and the local search capabilities of SA, while maintaining population diversity and preserving elite

solutions. The SA's probabilistic acceptance criterion facilitates escape from local optima, and the systematic local search ensures thorough solution refinement.

Algorithm 3 Local Search

Require: s: Initial solution, F: Facility costs, S: Service costs
Ensure: Locally optimized solution

$s_{best} \leftarrow s$
$C_{best} \leftarrow C(s, F, S)$
for $i = 1$ to $|s|$ **do**
    Generate neighbor solution $s_{neighbor}$ by flipping the i^{th} bit of s
    Compute $C_{neighbor} \leftarrow C(s_{neighbor}, F, S)$
    if $C_{neighbor} < C_{best}$ **then**
        $s_{best} \leftarrow s_{neighbor}$
        $C_{best} \leftarrow C_{neighbor}$
    end if
end for
return s_{best}

The overall computational complexity of the proposed hybrid algorithm can be analyzed by considering its main components. In the initialization phase, a population of N random solutions is generated, which requires $\mathcal{O}(N)$ time. In each of the G generations, every solution is evaluated using the cost function $C(s, F, S)$, with an evaluation cost of $\mathcal{O}(E)$ per solution (where E denotes the complexity of computing the cost for a given solution). Sorting the population based on these costs incurs an additional complexity of $\mathcal{O}(N \log N)$.

During the reproduction phase, the top 10 solutions are preserved while the remaining $N - 10$ individuals are generated via uniform crossover and subsequently refined using SA and a local search procedure. Uniform crossover operates in $\mathcal{O}(n)$ time, where n is the length of the binary solution. The SA procedure performs I_{SA} iterations, each requiring a cost evaluation, resulting in a per-individual complexity of $\mathcal{O}(I_{SA} \cdot E)$. The local search examines all n bits, with a complexity of $\mathcal{O}(n \cdot E)$. Therefore, the overall complexity per generation can be approximated by $\mathcal{O}\Big(N \log N + N \cdot E + (N-10) \cdot \big(n + I_{SA} \cdot E + n \cdot E\big)\Big)$. Over G generations, the total computational time is given by $\mathcal{O}\Big(G \cdot \Big[N \log N + N \cdot E + (N-10) \cdot \big(n + I_{SA} \cdot E + n \cdot E\big)\Big]\Big)$. This analysis shows that while the computational cost is influenced by the parameters N, G, n, and I_{SA}, appropriate tuning allows for a balance between solution quality and computational effort, rendering the algorithm feasible for medium-sized instances.

The performance of the algorithm can be fine-tuned by its key parameters. The population size N influences the solution diversity, while the number of generations G determines the computational budget. The initial temperature T_0 and cooling rate α control the acceptance probability of the SA, and the elite preservation and parental selection mechanisms balance exploitation and exploration. Through extensive testing, this hybrid algorithm has demonstrated robust performance on various UFLP instances, as detailed in the next section.

5. Experimental Results

To evaluate the effectiveness of our proposed hybrid algorithm, extensive computational experiments were conducted on a computer with an 11th Generation Intel(R) Core(TM) i5-1135G7 processor

running at 2.40 GHz, 8 GB RAM, and Windows 11 Home operating system. The algorithm was implemented in Python programming language.

We conducted a parameter analysis on the cap133 instance to evaluate the effect of different settings on performance. The parameters analyzed include the population size (N), initial temperature (T_0), and cooling rate (α). The performance was assessed using the gap score, representing the relative difference from the optimal solution. The results are given in Table 1.

Table 1: Parameter analysis on cap133 instance.

N	T_0	α	gap
30	500	0.93	0.0201
30	500	0.95	0.0020
30	500	0.98	0.0118
30	1000	0.93	0.0112
30	1000	0.95	0.0082
30	1000	0.98	0.0007
50	500	0.93	0.0038
50	500	0.95	0.0007
50	500	0.98	0.0013
50	1000	0.93	0.0007
50	**1000**	**0.95**	**0.0000**
50	1000	0.98	0.0007

The results in Table 1 show that increasing T_0 generally improves performance. For instance, the best result ($gap = 0.0$) was obtained with $N = 50$, $T_0 = 1000$, and $\alpha = 0.95$. Based on this analysis, we selected this configuration for our proposed method. Additionally, the choice of α significantly affects solution quality, with $\alpha = 0.95$ performing well across different configurations. These findings suggest that a higher T_0 combined with a moderate cooling rate and adequate population size leads to improved solution quality, providing useful insights for parameter tuning in similar optimization problems.

The proposed hybrid GA and SA approach was tested on 15 different UFLP benchmark instances. For each instance, the algorithm was executed 30 times to ensure statistical significance. The performance was evaluated using several metrics including best value, average value, worst value, median, standard deviation, and optimal value. In the implementation of the proposed hybrid algorithm, several key parameters were calibrated to optimize performance across UFLP instances. The population size was set to 50, which provided sufficient diversity to explore the solution space while maintaining computational efficiency. The evolutionary process was limited to 10 generations, which empirical tests showed to be sufficient for convergence on most problem instances. For the simulated annealing, an initial temperature of 1000 was chosen to allow extensive exploration in the early stages and to allow the algorithm to escape local optima by accepting worse solutions with reasonable probability. The cooling rate was set to 0.95, providing a gradual temperature reduction that effectively balances the exploration and exploitation phases. This cooling schedule allows the algorithm to transition smoothly from broad exploration to focused exploitation as the search progresses. These parameter values were determined through preliminary experimentation and represent an effective configuration for addressing the range of UFLP instances considered in this study. The combination of these parameter settings allows the hybrid algorithm to maintain solution diversity while efficiently converging on high-quality solutions.

Table 2 presents the detailed computational results obtained by our hybrid algorithm. The results demonstrate that the proposed algorithm achieves optimal solutions consistently for small and medium-sized problems, with a standard deviation of 0, indicating high solution stability and quality.

For larger instances (capa, capb, capc), while the algorithm may not always reach the optimal solution, it consistently finds high-quality solutions very close to the optimal values, with relatively small standard deviations.

Table 2: Computational results of the hybrid algorithm on UFLP instances.

Instance	Optimal	Best	Average	Worst	Median	Std Dev
cap71	932,615.75	932,615.75	932,615.75	932,615.75	932,615.75	0.00
cap72	977,799.40	977,799.40	977,799.40	977,799.40	977,799.40	0.00
cap73	1,010,641.45	1,010,641.45	1,010,641.45	1,010,641.45	1,010,641.45	0.00
cap74	1,034,976.98	1,034,976.98	1,034,976.98	1,034,976.98	1,034,976.98	0.00
cap101	796,648.44	796,648.44	796,648.44	796,648.44	796,648.44	0.00
cap102	854,704.20	854,704.20	854,704.20	854,704.20	854,704.20	0.00
cap103	893,782.11	893,782.11	893,782.11	893,782.11	893,782.11	0.00
cap104	928,941.75	928,941.75	928,941.75	928,941.75	928,941.75	0.00
cap131	793,439.56	793,439.56	793,439.56	793,439.56	793,439.56	0.00
cap132	851,495.33	851,495.33	851,495.33	851,495.33	851,495.33	0.00
cap133	893,076.71	893,076.71	893,076.71	893,076.71	893,076.71	0.00
cap134	928,941.75	928,941.75	928,941.75	928,941.75	928,941.75	0.00
capa	17,156,454.48	17,156,454.48	17,214,424.13	17,821,593.85	17,156,454.48	157,416.50
capb	12,979,071.58	12,979,071.58	12,997,631.01	13,084,984.12	12,979,071.58	32,673.20
capc	11,505,594.33	11,505,594.33	11,515,316.06	11,535,255.51	11,509,361.66	10,088.93

The performance analysis of our algorithm reveals several key findings:

1. **Solution Quality:** For small and medium-sized instances (cap71-cap134), the algorithm consistently achieves optimal solutions with perfect reliability, demonstrating its effectiveness in handling typical problem sizes.

2. **Stability:** The algorithm shows remarkable stability in solution quality across multiple runs, especially for smaller instances. This is evidenced by the identical best, average, and median values for most problems.

3. **Scalability:** While dealing with larger instances (capa, capb, capc), the algorithm maintains competitive performance. Although the standard deviation increases slightly for these instances, the solutions remain within a reasonable range of the optimal values:

    - For capa: Standard deviation of 157,416.50 (less than 1% of the optimal value)
    - For capb: Standard deviation of 32,673.20 (approximately 0.25% of the optimal value)
    - For capc: Standard deviation of 10,088.93 (less than 0.1% of optimal value)

Figure 1 shows comparative box plots for three different UFLP instances (capa, capb, and capc), revealing several informative patterns in the optimal total cost distributions. The distribution characteristics vary across instances, with capa exhibiting the largest spread and notable outliers, suggesting more variability in solution quality, while capb exhibits a moderately compact distribution with relatively symmetric quartiles, and capc exhibits the most compact distribution, suggesting more consistent solution quality. For all instances, the means (orange dashed lines) stay closer to the minima (blue dashed lines) than to the maxima (red dashed lines), suggesting that the algorithm tends to find solutions closer to the optimum more often. The decreasing relative spread from capa to capc (as shown by the height of the boxes compared to the total range) indicates improved solution stability

for smaller-valued instances, and the median lines generally staying closer to the minimum values suggests that the algorithm frequently achieves near-optimal solutions. These observations indicate that while the hybrid algorithm performs well across all instances, its consistency and reliability vary depending on the problem instance, with better stability shown in the capc instance compared to capa and capb. The visualization also facilitates a direct comparison of the algorithm's performance in terms of solution quality, stability, and consistency. For instance, the compact distribution and the median closely aligned with the minimum in instance capc indicate a high level of reliability in obtaining near-optimal solutions. In contrast, the broader spread and presence of outliers in instance capa suggest higher variability, likely due to increased problem scale and complexity, and may signal the need for further parameter tuning. Overall, these insights confirm the effectiveness of the proposed algorithm in balancing exploration and exploitation across different problem scales.

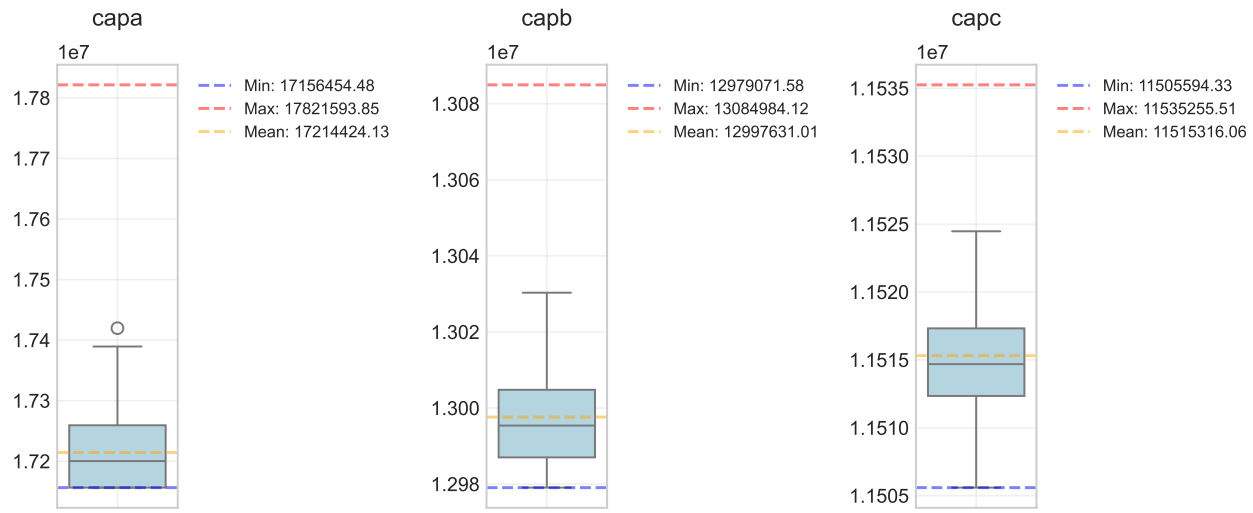

Figure 1: Box plots comparing the optimal total cost distributions for three UFLP instances (capa, capb, and capc), showing minimum, maximum, mean values, and quartile distributions.

The results demonstrate that our hybrid approach successfully combines the exploration capabilities of GA with the exploitation power of SA. The SA component provides broad exploration of the solution space, while the SA component helps to fine-tune solutions and escape local optima. This synergy is particularly evident in the algorithm's ability to consistently find optimal or near-optimal solutions across different problem sizes.

We compare the performance of the proposed hybrid GA-SA method with three binary variants of optimization algorithms. The Modified Binary Vortex Search (MBVS [3]) algorithm adapts the standard vortex search mechanism to handle binary optimization problems by incorporating probability-based position updates. The binary Artificial Bee Colony (binABC [16]) algorithm modifies the classical ABC algorithm by introducing binary solution representations and appropriate search operators for the binary domain.The binary Artificial Algae Algorithm (binAAA [17]) extends the biological-inspired algae optimization process to binary spaces through specialized movement and adaptation strategies. The comparative results presented in Table 3 shows significant insights across different problem scales. For small-scale instances (cap71-cap74 and cap101-cap104), all algorithms demonstrate exceptional performance by consistently reaching optimal solutions, with only MBVS showing a minimal deviation in cap103 with a gap of 0.0017 and a standard deviation of 57.34. This indicates that these instances are relatively easy for all the methods tested.

Table 3: Comparative results of optimization algorithms on UFLP instances in terms of gap and standard deviation values.

	MBVS [3]		binABC [16]		binAAA [17]		Hybrid GA-SA	
Instance	Gap	Std Dev	Gap	Std Dev	Gap	Std Dev	Gap	Std Dev
cap71	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
cap72	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
cap73	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
cap74	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
cap101	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
cap102	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
cap103	0.0017	57.34	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
cap104	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
cap131	0.0112	270.21	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
cap132	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
cap133	0.0591	328.65	0.1215	200.24	0.0007	31.91	**0.0000**	**0.00**
cap134	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**	**0.0000**	**0.00**
capa	5.8962	1,334,986.34	2.9622	236,833.50	**0.2345**	**78,259.57**	0.3379	157,416.50
capb	1.5752	313,844.03	2.5081	91,430.13	0.6747	57,572.96	**0.1430**	**32,673.20**
capc	0.6960	69,018.86	2.5800	82,312.70	0.4360	43,963.88	**0.0845**	**10,088.93**

The performance differences begin to emerge in medium-scale instances, particularly in cap133. Here, MBVS shows the highest gap of 0.0591 with considerable variability (standard deviation of 328.65), followed by binABC with a gap of 0.1215 and standard deviation of 200.24. The proposed hybrid GA-SA method achieve superior results achieving the optimal for all trials. The binAAA follows the proposed algorithm with a gap of 0.0007 and a lower standard deviation of 31.91. For other medium-scale instances, all algorithms maintain optimal performance.

The largest differences between algorithms appear in large instances (capa, capb, and capc). MBVS shows the highest gaps in these instances, reaching up to 5.8962 for capa, with extremely high standard deviations peaking at 1,334,986.34. The binABC algorithm shows moderate performance with gaps consistently around 2.5-3.0 and lower, though still significant, standard deviations compared to MBVS. Both binAAA and the proposed hybrid metaheuristic algorithm show superior performance on these hard instances. The proposed algorithm achieves the lowest gaps (0.1430 and 0.0845 for capb, and capc, respectively), while binAAA follows with slightly higher gaps. It can be clearly seen that both methods provide more stable solutions, as indicated by their lower standard deviations.

Overall, the comparative analysis shows that the proposed hybrid GA-SA method provides the most consistent and robust performance across all instance sizes. Its superiority is particularly evident in handling large-scale instances while maintaining solution stability. The effectiveness of the method is highlighted by its ability to achieve lower gaps with more consistent solutions compared to other algorithms, making it a promising approach for complex binary optimization problems.

6. Conclusion

This study introduced a novel hybrid metaheuristic algorithm that integrates the global exploration capabilities of GA with the local refinement strengths of SA to solve UFLP which is a binary-encoded optimization problem. Extensive computational experiments on 15 benchmark instances have shown that the proposed GA-SA approach consistently achieves optimal or near-optimal solutions with exceptional reliability for small and medium-sized problems, while maintaining competitive performance on larger instances. The comparative analyses with other binary optimization methods further underscore the robustness, stability, and scalability of the algorithm, particularly evident in its ability to

produce minimal gaps and low standard deviations over multiple independent runs.

Building on these promising results, several avenues for future research can be pursued. Adaptive parameter tuning strategies could be developed to dynamically adjust key algorithm parameters (e.g., population size, cooling rate, and temperature settings) based on real-time performance feedback, potentially improving both convergence speed and solution quality. In addition, integration of other metaheuristic techniques or incorporation of machine learning methods, such as reinforcement learning, to guide the search process can further improve the balance between exploration and exploitation. Parallel implementations of the hybrid algorithm are also worth investigating, as they could significantly reduce the computational time for large instances. The hybrid GA-SA method provides a robust and effective framework for solving complex binary optimization problems, with room for improvement and real-world applications.

Acknowledgement

This research was not funded by any grant.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Alidaee, B., & Wang, H. (2022). Uncapacitated (Facility) Location Problem: A Hybrid Genetic-Tabu Search Approach. *IFAC-PapersOnLine*, *55*(10), 1619–1624. https://doi.org/10.1016/j.ifacol.2022.09.622

[2] Al-Sultan, K. S., & Al-Fawzan, M. A. (1999). A tabu search approach to the uncapacitated facility location problem. *Annals of Operations Research*, *86*(0), 91–103. https://doi.org/10.1023/A:1018956213524

[3] Aslan, M., & Pavone, M. (2024). MBVS: A modified binary vortex search algorithm for solving uncapacitated facility location problem. *Neural Computing and Applications*, *36*(5), 2573–2595. https://doi.org/10.1007/s00521-023-09190-9

[4] Baş, E., & Ülker, E. (2020). A binary social spider algorithm for uncapacitated facility location problem. *Expert Systems with Applications*, *161*, 113618. https://doi.org/10.1016/j.eswa.2020.113618

[5] Baş, E., & Yildizdan, G. (2024). A new binary arithmetic optimization algorithm for uncapacitated facility location problem [Company: Springer Distributor: Springer Institution: Springer Label: Springer Number: 8 Publisher: Springer London]. *Neural Computing and Applications*, *36*(8), 4151–4177. https://doi.org/10.1007/s00521-023-09261-x

[6] Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, *64*(12), 1695–1724. https://doi.org/10.1057/jors.2013.71

[7] Cinar, A. C. (2023). A novel adaptive memetic binary optimization algorithm for feature selection. *Artificial Intelligence Review*, *56*(11), 13463–13520. https://doi.org/10.1007/s10462-023-10482-8

[8] Çınar, A. C. (2022). A Comprehensive Comparison of Binary Archimedes Optimization Algorithms on Uncapacitated Facility Location Problems. *Duzce University Journal of Science and Technology*, *10*(1). https://doi.org/10.29130/dubited.876284

[9] Durgut, R., & Aydin, M. E. (2021). Adaptive binary artificial bee colony algorithm. *Applied Soft Computing*, *101*, 107054. https://doi.org/10.1016/j.asoc.2020.107054

[10] Efroymson, M. A., & Ray, T. L. (1966). A Branch-Bound Algorithm for Plant Location. *Operations Research*, *14*(3), 361–368. https://doi.org/10.1287/opre.14.3.361

[11] Erdoğan, F., Karakoyun, M., & Gülcü, Ş. (2024). An effective binary dynamic grey wolf optimization algorithm for the 0-1 knapsack problem. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-024-20121-1

[12] Erlenkotter, D. (1978). A Dual-Based Procedure for Uncapacitated Facility Location. *Operations Research*, *26*(6), 992–1009. https://doi.org/10.1287/opre.26.6.992

[13] Gölcük, İ., & Ozsoydan, F. B. (2020). Evolutionary and adaptive inheritance enhanced Grey Wolf Optimization algorithm for binary domains. *Knowledge-Based Systems*, *194*, 105586. https://doi.org/10.1016/j.knosys.2020.105586

[14] Jiang, N., & Zhang, H. (2023). Improved Adaptive Differential Evolution Algorithm for the Un-Capacitated Facility Location Problem. *Open Journal of Applied Sciences*, *13*(5), 685–695. https://doi.org/10.4236/ojapps.2023.135054

[15] Kaya, E. (2022). BinGSO: Galactic swarm optimization powered by binary artificial algae algorithm for solving uncapacitated facility location problems. *Neural Computing and Applications*, *34*(13), 11063–11082. https://doi.org/10.1007/s00521-022-07058-y

[16] Kiran, M. S., & Gündüz, M. (2013). Xor-based artificial bee colony algorithm for binary optimization. *Turkish Journal of Electrical Engineering and Computer Sciences*, *21*(8), 2307–2328. https://doi.org/10.3906/elk-1203-104

[17] Korkmaz, S., Babalik, A., & Kiran, M. S. (2018). An artificial algae algorithm for solving binary optimization problems. *International Journal of Machine Learning and Cybernetics*, *9*, 1233–1247. https://doi.org/10.1007/s13042-017-0772-7

[18] Kuehn, A. A., & Hamburger, M. J. (1963). A Heuristic Program for Locating Warehouses. *Management Science*, *9*(4), 643–666. https://doi.org/10.1287/mnsc.9.4.643

[19] Özkış, A., & Karakoyun, M. (2023). A binary enhanced moth flame optimization algorithm for uncapacitated facility location problems. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, *29*(7), 737–751. https://doi.org/10.5505/pajes.2023.49576

[20] Ozsoydan, F. B., & Kasırga, A. E. (2024). Evolution inspired binary flower pollination for the uncapacitated facility location problem. *Neural Computing and Applications*, *36*(20), 12117–12130. https://doi.org/10.1007/s00521-024-09684-0

[21] Peidro, D., Martin, X. A., Panadero, J., & Juan, A. A. (2024). Solving the uncapacitated facility location problem under uncertainty: A hybrid tabu search with path-relinking simheuristic approach. *Applied Intelligence*, *54*(7), 5617–5638. https://doi.org/10.1007/s10489-024-05441-x

[22] Sonuç, E. (2021). Binary crow search algorithm for the uncapacitated facility location problem. *Neural Computing and Applications*, *33*(21), 14669–14685. https://doi.org/10.1007/s00521-021-06107-2

[23] Sonuç, E., & Özcan, E. (2023). An adaptive parallel evolutionary algorithm for solving the uncapacitated facility location problem. *Expert Systems with Applications*, *224*, 119956. https://doi.org/10.1016/j.eswa.2023.119956

[24] Sonuç, E., & Özcan, E. (2024). CUDA-based parallel local search for the set-union knapsack problem. *Knowledge-Based Systems*, *299*, 112095. https://doi.org/10.1016/j.knosys.2024.112095

[25] Talbi, E. (2009). *Metaheuristics: From Design to Implementation*. Wiley. https://doi.org/10.1002/9780470496916

[26] Zhang, F., He, Y., Ouyang, H., & Li, W. (2023). A fast and efficient discrete evolutionary algorithm for the uncapacitated facility location problem. *Expert Systems with Applications*, *213*, 118978. https://doi.org/10.1016/j.eswa.2022.118978