A Machine Learning Enhanced Variable Neighborhood Search Approach for the Uncapacitated Facility Location Problem

Lucas Martín-García[1], Isaac Lozano-Osorio[1(✉)], J. Manuel Colmenar[1], and Belén Melián-Batista[2]

[1] Universidad Rey Juan Carlos, Madrid, Spain
{lucas.martin,isaac.lozano,josemanuel.colmenar}@urjc.es
[2] Universidad de La Laguna, La Laguna, Spain
mbmelian@ull.edu.com

Abstract. The Uncapacitated Facility Location Problem (UFLP) is widely recognized as a relevant problem in logistics, resource distribution, and telecommunications network planning. Given a set of potential facility locations and a set of customers, the goal is to determine which facilities to open to serve all customers while minimizing both opening and assignment costs. Since this problem is classified as $\mathcal{NP}$-hard, obtaining exact solutions at large scales could not be possible, thereby motivating the use of approximation techniques and metaheuristics. Although early studies used exact formulations derived from the UFLP model, recent research has emphasized the efficacy of approximate and metaheuristic algorithms, which achieve high-quality solutions with substantially reduced computational effort. This work introduces a Variable Neighborhood Search approach to tackle this problem. With the aim of guiding the search toward higher-quality solutions, machine learning techniques have been incorporated to this process. Experimental results on well-known benchmark datasets demonstrate that our method reaches solutions very close to the optimal values, with significantly shorter execution times, outperforming state-of-the-art algorithms validated by the pairwise non-parametric Wilcoxon statistical test.

Keywords: Variable Neighborhood Search · Machine Learning · Uncapacitated Facility Location Problem · Reinforcement Learning

1 Introduction

The Uncapacitated Facility Location Problem (UFLP) is one of the most widely studied optimization problems within the broader family of Facility Location Problems (FLP), devoted to optimally placing facilities to serve customers [5]. Due to its fundamental importance in location theory and its direct applicability to practical domains, ranging from logistics and telecommunications to

S. Cavero et al. (Eds.): ICVNS 2025, LNCS 16256, pp. 15–30, 2026.
https://doi.org/10.1007/978-3-032-19582-1_2

the strategic design of road and railway networks [12], the UFLP has been the subject of extensive research.

Since its introduction [16], the approach to solving the UFLP has evolved significantly. During the latter half of the 20th century, the research was focused on exact algorithms in the UFLP [8] and other FLP variants [1,6], which established a strong formal basis but were limited to instances of moderate size . To overcome this scalability issue, the field later pivoted towards approximate methods, such as the notable greedy local search with progressive cost adaptation [3], designed to handle larger and more complex scenarios. This trend has continued with the development of sophisticated metaheuristics, including genetic algorithms [15], hybrid multistart heuristics [21], and evolutionary algorithms [20], which now dominate the landscape of UFLP solution techniques.

More recently, the best results on this problems were obtained by the Enhanced Group Theory-Based Optimization Algorithm (EGTOA) [23]. As an advanced version of the Group Theory-Based Optimization Algorithm (GTOA) [11], EGTOA has demonstrated superior performance in both solution quality and convergence, outperforming 16 other literature algorithms on the established OR-Library instances. While evolutionary methods like EGTOA are highly effective, this paper explores a different yet powerful paradigm.

A trend in recent years has been the integration of Machine Learning (ML) techniques with metaheuristics to solve complex combinatorial optimization problems [14]. This synergy aims to create a more intelligent search process, improving performance in terms of solution quality, convergence rate, and robustness by extracting useful knowledge from data generated during the search. The literature shows various successful integration points, including enhancing the initialization of solutions and the evolutionary process itself, for instance, through adaptive operator selection.

This approach has proven effective both for the Variable Neighborhood Search (VNS) framework and for problems in the FLP family. For example, Reinforcement Learning has been used within VNS to automate the selection of the most effective neighborhood operators, demonstrating significant improvements in solution quality over conventional VNS implementations [13]. Separately, ML has been applied to the Facility Location Problem by learning from historical data to predict how solutions should be modified in response to parameter perturbations, which can then guide the solver [17].

The demonstrated success of these hybrid strategies provides the primary motivation for this work. In this work, a novel integration of ML into VNS is proposed, specifically designed to guide the constructive phase of the algorithm. Specifically, our approach extracts information of the instance problems, learns from it, and uses the learned information to generate better initial solutions that help the convergence of our algorithmic proposal towards near-optimal solutions for the UFLP. Our experimental results confirm that this VNS-based method yields competitive solutions, establishing it as a viable and effective alternative for tackling the UFLP.

Regarding the structure of the document, the foundational context of the problem and our algorithmic contribution are firstly stated, proceeding then to its empirical validation. More precisely, the paper continues in Sect. 2 with a detailed mathematical formulation of the UFLP and defines the solution representation used in this work. Based on this, Sect. 3 presents our proposed VNS methodology. The effectiveness of the algorithm is then evaluated in Sect. 4, which details the experimental setup and analyzes the results. Finally, the conclusions are drawn in Sect. 5.

2 Problem Formulation

The Uncapacitated Facility Location Problem (UFLP) seeks to determine which subset from a group of potential facilities should be opened to serve a set of customers at a minimum total cost. This total cost is the sum of fixed costs for opening the selected facilities and the costs associated with assigning customers to them. A key feature of the problem is that the number of facilities to be opened is not predetermined.

More formally, let $A = \{a_1, a_2, \ldots, a_M\}$ be a set of M customers and $B = \{b_1, b_2, \ldots, b_N\}$ be a set of N potential facility locations. The cost of serving customer a_i from facility b_j is given by c_{ij} in the cost matrix $C = [c_{ij}]_{M \times N}$. Additionally, each facility b_j has a fixed opening cost d_j.

The problem can be formulated using binary decision variables [15]. For each facility b_j (where $j = 1, \ldots, N$), the variable x_j is equal to 1 if the facility is opened and 0 otherwise. Similarly, the variable y_{ij} is equal to 1 if customer a_i (for $i = 1, \ldots, M$) is assigned to facility b_j, and 0 otherwise. The objective function is presented in Equation (1), while equations (2) through (4) define the constraints that ensure that each customer is assigned to exactly one open facility.

$$\min \quad \sum_{j=1}^{N} d_j x_j + \sum_{i=1}^{M} \sum_{j=1}^{N} c_{ij} y_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{N} y_{ij} = 1, \quad \forall i \in \{1, 2, ..., M\} \tag{2}$$

$$y_{ij} - x_j \leq 0, \quad \forall j \in \{1, 2, ..., N\}, \forall i \in \{1, 2, ..., M\} \tag{3}$$

$$y_{ij}, x_j \in \{0, 1\}, \quad \forall j \in \{1, 2, ..., N\}, \forall i \in \{1, 2, ..., M\}. \tag{4}$$

A crucial property of the UFLP is that for any given configuration of open facilities, the optimal customer assignment can be determined efficiently. Specifically, each customer is assigned to the open facility that offers the lowest service cost for that customer. Consequently, any solution to the problem can be uniquely and compactly represented by the set of opened facilities, without the need to explicitly state the customer assignments. Therefore, we represent a solution S as the set of open facilities, i.e., $S = \{b_j \in B : \ x_j = 1\}$. The total cost of

a solution S, as determined by the objective function in Eq. (1), will be denoted as $\mathcal{F}(S)$.

To illustrate the difficulty of this problem, consider the example shown in Fig. 1, which consists of four customers (denoted by blue circles) and three potential facilities, b_1, b_2, and b_3 (denoted by empty squares). Each facility has an opening cost of 4 units, and the assignment costs, shown on the edges, correspond to the Euclidean distances.

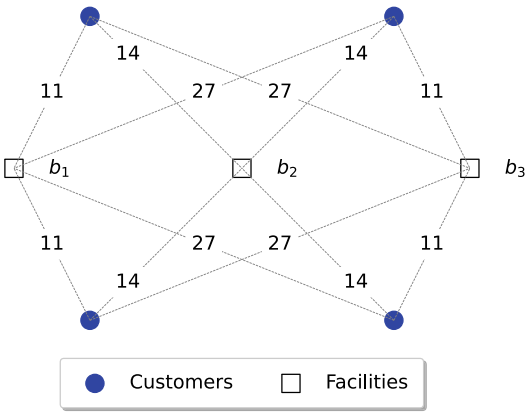
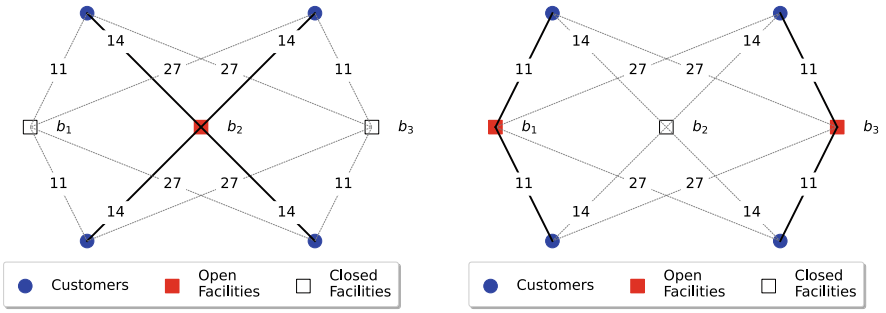


Fig. 1. An instance of the UFLP with four customers (blue circles) and three potential facilities (empty squares). The values on the edges represent the cost of serving each customer from the corresponding facility. (Color figure online)

This simple example highlights the core trade-off of the problem. One possible solution, shown in Fig. 2a, is to open only facility b_2, which is centrally located. This solution, $S_1 = \{b_2\}$, incurs a low opening cost of 4, but the total assignment cost is 56 (14 for each of the four customers), leading to a total cost of $\mathcal{F}(S_1) = 60$. However, a better solution can be obtained by opening two facilities. More precisely, $S_2 = \{b_1, b_3\}$, shown in Fig. 2b. While the opening cost of this solution doubles to 8, the sum of assignment costs is significantly reduced to 44, resulting in a lower total cost of $\mathcal{F}(S_2) = 52$, which is the optimal solution for this instance. This example demonstrates the inherent complexity of the UFLP. Since the number of facilities to open is unconstrained, the number of feasible configurations grows significantly, even for a small instance with only three potential locations.

3 Proposed Algorithm

The Uncapacitated Facility Location Problem (UFLP) is classified as $\mathcal{NP}$-hard [23], which means that exact approaches are often computationally infeasible for



(a) Sub-optimal solution: $S_1 = \{b_2\}$ (b) Optimal solution: $S_2 = \{b_1, b_3\}$

Fig. 2. Alternative solutions for the example instance.

large-scale instances. To address this complexity, this paper proposes an algorithm based on the Variable Neighborhood Search (VNS) metaheuristic, designed to find high-quality solutions within a reasonable timeframe. VNS operates by systematically exploring multiple neighborhood structures, complemented by a perturbation mechanism that allows the search to effectively escape local optima [10,19].

The VNS metaheuristic has been widely and successfully applied to a variety of combinatorial optimization problems. For instance, in the context of location problems, a parallelized VNS has proven highly efficient for the p-Median Problem [9]. Similarly, an optimized version of VNS obtained competitive results for the Capacitated p-Median Problem (CPMP) [7]. More recent applications include a VNS variant for the median problem with interconnected facilities [18] and a parallelized VNS for the Bus Terminal Location Problem (BTLP), which achieved significant improvements in solution quality and computation time [4]. This strong track record motivates its application to the UFLP.

In this work, we adapt VNS for the UFLP to leverage its ability to efficiently explore the solution space and avoid premature convergence. Specifically, our proposal is based on the Basic VNS (BVNS) variant [10]. BVNS effectively balances diversification (through random perturbations) and intensification (through local search), which is key to finding high-quality solutions. The pseudocode for our proposed BVNS algorithm is detailed in Algorithm 1.

The `BVNS(`$k_{\max}$`)` method requires the maximum neighborhood size ($k_{\max}$) as an input parameter. The execution begins by generating an initial solution S (line 1) using a constructive method (see Sect. 3.1), which is then set as the best-found solution, denoted S^* (line 2).

The main loop (lines 4–13) iterates until no improvement is found for $k_{\max}$ consecutive neighborhood sizes. In each iteration, the current solution S is perturbed by the `Shake` function (line 5), which modifies k facilities randomly selected to diversify the search (see Sect. 3.2). Following this, the `Improvement`

Algorithm 1. BVNS($k_{\max}$)

```
1: S ← Constructive()
2: S* ← S
3: k ← 0
4: while k < k_max do
5:     S ← Shake(S*, k)
6:     S ← Improvement(S)
7:     if F(S) < F(S*) then
8:         S* ← S
9:         k ← 1
10:    else
11:        k ← k + 1
12:    end if
13: end while
14: return S*
```

function is applied to the new solution S (line 6) to perform an intensification phase (see Sect. 3.3).

Finally, the algorithm checks if the resulting solution S is better than the best-found solution S^*. If an improvement is found, S^* is updated, and the neighborhood counter k is reset to 1 (lines 7–9). Otherwise, k is incremented (line 11). The size of the perturbation generated by the `Shake` step increases with k. After the termination condition is met, the algorithm returns the best overall solution found, S^* (line 14).

3.1 Constructive Phase

The performance of the BVNS algorithm can be significantly influenced by the quality of its starting point. A well-chosen initial solution can guide the search towards promising regions of the solution space, while a diverse set of starting points can enhance exploration. To this end, we propose and evaluate two different constructive strategies for generating the initial solution S (line 1 of Algorithm 1): a simple random method and a more sophisticated method guided by a reinforcement learning mechanism.

Random Constructive. The first approach is a straightforward random method designed to ensure a valid and diverse starting point with minimal computational cost. The process begins by initializing an empty solution, $S = \emptyset$, which means that all facilities are initially closed. Then, the algorithm iterates through each potential facility and, for each one, decides whether to open it with a probability of 50%.

This stochastic process promotes diversity in the starting points for the VNS. However, it is possible that after iterating through all facilities, the solution set S remains empty. To guarantee a feasible solution where all customers can be served, a post-construction check is performed to ensure at least one facility

is open. This simple and computationally inexpensive method ensures a valid starting point and introduces variability, which can help the subsequent search phases to explore different parts of the solution landscape.

Reinforcement Learning Constructive. The second constructive method employs a learning mechanism to intelligently guide the generation of the initial solution. This strategy is designed to obtain information gathered from previous search efforts in the same instance to build progressively better starting solutions over multiple runs of the algorithm. It operates by maintaining a vector of probabilities or rewards, R_k, where each element $R_k(i)$ represents the learned desirability of opening the facility i in iteration k.

Initially, all rewards are set to a neutral value, $R_0(i) = 0.5$ for all facilities $i = 1, \ldots, N$. Consequently, the first execution of this constructive is identical to the random constructive method. A solution S is built by iterating through all facilities and opening each facility i with a probability equal to its current reward value, $R_k(i)$.

The core of this approach lies in its update rule, which is applied after the BVNS algorithm has completed a full execution and returned a final, best-found solution, S^*. The rewards are then updated for the next run of the algorithm based on the composition of S^*. The update rule is defined as follows:

$$R_{k+1}(i) = \min\big(1,\, \max(0,\, R_k(i) + \delta(i))\big)$$

where for each facility $i = 1, \ldots, N$:

$$\delta(i) = \begin{cases} +\alpha & \text{if facility } i \in S^* \\ -\alpha & \text{if facility } i \notin S^* \end{cases}$$

Here, α is a learning rate, which we set to $1/N$ to ensure a gradual adaptation. This update mechanism reinforces the selection of facilities that were part of the best-found solution and penalizes those that were not.

This strategy can be framed as an application of Reinforcement Learning by defining its components at a higher level of abstraction. Here, the constructive mechanism itself is the agent, whose policy is parameterized by the reward vector R. A single action consists of generating a complete initial solution S, and a full run of the BVNS algorithm represents one episode. The agent receives a single, delayed reward at the conclusion of the episode, which is implicitly encoded in the final solution S^*. The update rule then adjusts the agent's policy by reinforcing the choices that contributed to this high-quality outcome. By iteratively refining its policy across multiple episodes, the agent learns to generate more effective starting points, concentrating the search effort in more promising areas of the solution landscape.

3.2 Diversification Phase

The `Shake` function, detailed in Algorithm 2, is responsible for the diversification component of the algorithm. Its main purpose is to perturb a given solution S to

guide the search to a new neighborhood and escape local optima when the local search can no longer find improvements. The perturbation strategy is based on opening facilities, designed to complement the local search phase, which focuses on closing facilities (see Sect. 3.3). By adding a subset of currently closed facilities to the solution, the search effectively distances itself from the explored region, with the parameter k determining the size of this subset.

Algorithm 2. `Shake(`S`,`k`)`

1: $L \leftarrow \{b_j : b_j \in B \wedge b_j \notin S\}$
2: $L' \leftarrow$ `RandomSelection(L,k)`
3: $S \leftarrow S \cup L'$
4: **return** S

The process outlined in Algorithm 2 begins by creating an initial list L composed of all facilities that are closed in solution S (line 1). Subsequently, to ensure a varied perturbation, k facilities are selected from this list at random, taking into account that $min(k, |L|)$ (line 2) is the maximum number of facilities that can be opened. Finally, the selected facilities are added to the current solution (line 3), which is then returned by the function (line 4).

3.3 Improvement Phase

The improvement phase of our proposed algorithm serves as the intensification mechanism, where solutions are refined through local search. The core operator for this phase is the closure of an open facility, a move that defines the neighborhoods explored within the BVNS schema. Previous work has shown that this type of move improves both the efficiency and quality of the search [23]. The motivation is that closing facilities reduces the total number of active locations, thereby lowering fixed opening costs and promoting higher-quality solutions. It is also highlighted that maintaining an excessive number of open facilities tends to inflate the objective function value; thus, closing those that do not contribute positively is an effective refinement strategy.

A best improvement strategy has been proposed based on the facility closure move. This procedure evaluates all neighboring solutions, closing the facility that yields the greatest reduction of the objective function.

In addition to the best improvement strategy, a variant based on first improvement was also implemented. In this version, instead of evaluating all neighbors to find the best one, potential facility closures are explored in a random order. The first move that improves the current solution in this random order is immediately accepted. Once an improving closure is found, the solution is updated, and the search process restarts from this new improved solution.

4 Computational Results

This section presents the set of experiments conducted to evaluate the proposed algorithm and compare it against the evolutionary algorithm EGTOA and an exact model, which are considered state-of-the-art references. All experiments were performed on a server equipped with an AMD EPYC 7643 48-core processor capped to 32 cores and 32 GB of RAM. The proposed algorithm was implemented using the Java 21 programming language.

The benchmark instances used in this work are drawn from the literature. Notably, we have expanded upon these by incorporating two additional, widely recognized datasets from related problems. These datasets were evaluated under the same experimental environment to facilitate a robust comparison between algorithms and to analyze their performance across different problem parameters. The characteristics of all instance sets are detailed in Table 1. The first set consists of the Cap instances, which are frequently used for this family of facility location problems and are available from the OR-Library [2][1]. Given the limited size of this initial set, we opted to include more extensive instance sets from the state of the art. Consequently, the K90 and G700 sets [12] were added to the comparison. These instances were generated from the railway and road networks of the Slovak Republic and are openly available[2]. The K90 set originally consisted of 90 individual instances. However, three of them contain negative cost values and were therefore excluded from our analysis to maintain consistency, following the authors of the instances [12]. In summary, a total number of 802 instances were studied, ranging from 16 to 1000 facilities and 50 to 2906 customers.

Table 1. Description of the benchmark instance sets studied.

Name	Instances	Facilities	Customers	Reference
Cap	15	16–100	50–1000	[2]
K90	87	45–457	457	[12]
G700	700	100–1000	2906	[12]

The EGTOA algorithm was executed on the new instance sets, which was made possible by the authors' decision to publish the source code alongside their article [23], allowing for its adaptation and application in this study. The exact model, on the other hand, was implemented in Gurobi using the mathematical formulation of the problem.

The results tables in our experiments present four key metrics to evaluate algorithm performance. The metric denoted as $\mathcal{F}$ reflects the average objective function value obtained by each algorithm, where a lower value indicates a better

[1] https://people.brunel.ac.uk/~mastjjb/jeb/orlib/uncapinfo.html.
[2] http://frdsa.uniza.sk/~buzna/supplement.

solution. *Time (s)* represents the average computation time in seconds required for each algorithm to run. *Gap (%)* measures the average percentage difference between the solutions found by an algorithm and the optimal solutions, providing a measure of relative error. Finally, # *Opt* indicates the number of instances for which each algorithm found the optimal solution.

To ensure the robustness of our results and account for the stochastic elements of the algorithm, the proposed VNS variants, the random constructive based BVNS-R and the reinforcement learning constructive based BVNS-RL, were executed 30 times each for every instance. This multi-run approach serves a dual purpose. For the BVNS-R variant, it ensures that the search begins from multiple, diverse random starting points. For the BVNS-RL variant, it provides the necessary number of episodes for the reinforcement learning mechanism to effectively learn and adapt its constructive strategy based on previous outcomes in the same instance. The results presented in the tables are aggregated from these 30 runs as follows: the objective function ($\mathcal{F}$) and Gap (%) values reflect the best solution found across all 30 executions, while the reported Time (s) represents the total cumulative time for the full 30 runs to complete.

The experiments are structured in two stages: a preliminary phase and a final comparison phase. The preliminary phase involves conducting tests to determine the best parameter values for each variant of the proposed algorithm. Subsequently, the final phase evaluates the performance of the selected best configurations, comparing them against the other aforementioned methods.

4.1 Preliminary Results

The preliminary experimental phase is designed to tune the parameters of our proposed algorithm. This was carried out on a representative subset of the instances, created by randomly selecting 20% of the instances from each dataset using stratified sampling [22]. This approach ensures that the preliminary subset maintains the proportional representation and diversity of the full benchmark, which includes instances with varying origins, sizes, and cost distributions.

The primary goal of this phase is to determine the optimal configuration of the algorithms considering two variants depending on the constructive method: Random (BVNS-R), and Reinforcement Learning (BVNS-RL). The configuration implies determining the local search strategy (First Improvement vs. Best Improvement) and the perturbation size factor, K, for which we tested the values $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. The parameter K is used to define the maximum neighborhood size in Algorithm 1 as $k_{\max} = \lceil N \cdot K \rceil$, where N is the total number of potential facilities. The results for each of the two variants are presented and analyzed separately.

Table 2 shows the aggregated results for the BVNS-R variant. A clear observation across all values tested for K is that the Best Improvement local search strategy consistently and significantly outperforms the First Improvement strategy across all metrics: it achieves lower objective function values ($\mathcal{F}$), substantially smaller Gap values, finds a much higher number of optimal solutions (# Opt) and spends a shorter computational time.

Focusing on the Best Improvement strategy to select the best value for K, we analyze the trade-off between the quality of the solution and the computational time. As expected, execution time is directly proportional to K, as a larger maximum perturbation size increases the computational effort of the local search. While $K = 0.1$ is the fastest configuration, increasing the parameter to $K = 0.2$ yields a dramatic improvement in solution quality reducing the Gap by more than 70% (from 0.006122% to 0.001724%). This substantial gain justifies more than doubling the execution time. However, a further increase to $K = 0.3$ nearly doubles the time again, but offers only a marginal improvement in the Gap (from 0.001724% to 0.001707%). This indicates diminishing returns for larger values of K. Therefore, we select $K = 0.2$ as the best-balanced configuration for the Random Constructive variant BVNS-R, as it provides an excellent compromise between computational cost and solution quality.

Table 2. Preliminary results for the BVNS algorithm with the Random Constructive, BVNS-R, evaluated on 20% of the dataset.

Local Search	K	$\mathcal{F}$	Time (s)	Gap (%)	# Opt
First Improvement	0.1	3,174,785.33	792.34	0.274205	58
	0.2	3,170,779.44	1007.23	0.233152	66
	0.3	3,171,017.58	1240.25	0.243865	65
	0.4	3,171,210.25	1502.95	0.244462	66
	0.5	3,170,757.61	1728.05	0.233776	65
Best Improvement	0.1	3,167,364.00	**93.92**	0.006122	133
	0.2	3,167,281.28	227.00	0.001724	148
	0.3	3,167,279.87	423.14	0.001707	**151**
	0.4	3,167,278.90	684.37	0.001595	**151**
	0.5	**3,167,276.27**	1017.10	**0.001489**	150

The results for the BVNS-RL variant are presented in Table 3. As with the random version, the Best Improvement local search is unequivocally superior to First Improvement, yielding better results in all metrics for all tested K values.

Therefore, we proceed to select the optimal K value for the Best Improvement strategy. The analysis reveals a similar pattern. Moving from $K = 0.1$ to $K = 0.2$ leads to a significant improvement in solution quality, with the Gap dropping from 0.210642% to 0.001082%. This significant leap in performance strongly justifies the increase in execution time from 88 s to 225 s. In contrast, increasing the parameter further to $K = 0.3$ almost doubles the runtime again, but provides a much smaller relative improvement in the Gap (to 0.000836%). This again points to $K = 0.2$ as the best value in this balance. Although other configurations achieve better individual metrics (e.g., $K = 0.3$ has the best $\mathcal{F}$ and Gap), the configuration with $K = 0.2$ provides a solution of nearly identical quality with less computational cost.

Table 3. Preliminary results for the BVNS algorithm with the Reinforcement Learning Constructive, BVNS-RL, evaluated on 20% of the dataset.

Local Search	K	$\mathcal{F}$	Time (s)	Gap (%)	# Opt
First Improvement	0.1	3,177,592.01	939.49	0.497938	58
	0.2	3,172,158.29	1093.90	0.245668	65
	0.3	3,171,408.92	1316.49	0.239123	66
	0.4	3,174,379.58	1598.42	0.262672	63
	0.5	3,170,757.10	1853.15	0.232076	63
Best Improvement	0.1	3,169,410.98	**88.01**	0.210642	139
	0.2	3,167,271.68	224.93	0.001082	**151**
	0.3	**3,167,269.01**	424.91	**0.000836**	**151**
	0.4	3,167,276.50	683.34	0.001309	147
	0.5	3,167,271.70	1020.71	0.001015	149

Based on this preliminary analysis, we conclude that the Best Improvement is the most suitable local search strategy for this problem. For the perturbation size, $K = 0.2$ provides the best overall trade-off between execution time and solution quality for both constructive methods. Therefore, the configurations selected for the final comparison are BVNS-R (Random Constructive) and BVNS-RL (RL Constructive), both using the Best Improvement local search and a $K = 0.2$ value.

4.2 Final Results

Following the parameter tuning in the preliminary phase, the two best configurations of our proposed algorithm, BVNS-R and BVNS-RL, were evaluated against the state-of-the-art EGTOA algorithm and the exact model. This final comparison was performed on the entire set of benchmark instances. The comprehensive results, broken down by instance set as well as aggregated, are presented in Table 4.

On the standard Cap instances, both the EGTOA and BVNS-RL algorithms successfully find all 15 optimal solutions, matching the performance of the Exact Model. The BVNS-R variant performs nearly as well, finding 14 optima with a low Gap of 0.00218%. In terms of computational time, our proposed VNS methods are exceptionally efficient, with BVNS-RL being the fastest overall at 0.208 s, significantly outperforming both the exact model (0.802 s) and the much slower EGTOA (258.93 s).

For the K90 instance set, the performance differences become more pronounced. EGTOA struggles significantly, resulting in a large average Gap of 12.13% and failing to find any optimal solutions, all while requiring over 1,400 s. In contrast, both BVNS-R and BVNS-RL achieve near-optimal results, with gaps of just 0.00568% and 0.00374%, respectively. Their execution times are

Table 4. Final results comparing EGTOA, the proposed BVNS variants, and the exact model.

Instances	Algorithm	$\mathcal{F}$	Time (s)	Gap (%)	# Opt
Cap	EGTOA	3,502,545.59	258.931	0.00000	15
	BVNS-R	3,502,796.74	0.254	0.00218	14
	BVNS-RL	3,502,545.59	0.208	0.00000	15
	Exact Model	3,502,545.59	0.802	0.00000	15
K90	EGTOA	3,069,045.61	1,405.238	12.12858	0
	BVNS-R	2,765,428.69	3.825	0.00568	71
	BVNS-RL	2,765,390.11	3.508	0.00374	73
	Exact Model	2,765,298.57	2.642	0.00000	87
G700	EGTOA	27,202,096.73	3,597.980	174.81629	23
	BVNS-R	2,943,809.46	248.588	0.00018	666
	BVNS-RL	2,943,809.24	239.329	0.00010	678
	Exact Model	2,943,809.00	62.861	0.00000	700
Aggregated	EGTOA	24,140,913.79	1,236.663	153.89849	38
	BVNS-R	2,934,913.80	217.392	0.00081	751
	BVNS-RL	2,934,904.73	209.275	0.00050	766
	Exact Model	**2,934,894.58**	**55.168**	**0.00000**	**802**

also highly competitive, remaining in the same order of magnitude as the exact model and finding a large majority of the optimal solutions.

The trend continues and is further amplified on the large-scale G700 instances. Here, EGTOA's performance degrades severely, yielding a very high Gap of 174.8% and requiring nearly an hour of computation time. Our VNS variants, however, maintain their excellent performance, delivering solutions with gaps close to zero and finding almost all 700 optimal solutions. While the exact model is fastest on this set, our methods provide a robust and effective heuristic alternative, demonstrating strong scalability.

The aggregated results provide a clear overview of overall performance. The state-of-the-art EGTOA is shown to be non-competitive on the broader set of instances, with an extremely high average gap and long execution time. In contrast, both of our proposed methods, BVNS-R and BVNS-RL, deliver solutions of exceptional quality, with average gaps of just 0.00081% and 0.00050%, respectively. While the exact model is the fastest on average, our algorithms are orders of magnitude faster than EGTOA. Furthermore, when comparing our two proposed variants, the BVNS-RL consistently shows a slight edge over BVNS-R across all aggregated metrics: a better objective function value, a smaller gap, more optima found, and a faster runtime. This confirms the positive contribution of the Reinforcement Learning mechanism. To conclude the experimental section, our VNS-based approaches, particularly BVNS-RL, prove to be highly

effective and efficient alternatives for solving the UFLP, providing near-optimal results with remarkable consistency and speed.

Finally, we apply the widely used non-parametric Wilcoxon signed-rank test for pairwise comparisons to investigate whether the solutions generated by the two algorithms originate from distinct populations. The obtained p-value, which is smaller than 0.0001 when comparing BVNS-RL with EGTOA and BVNS-R with EGTOA, provides strong evidence supporting the superior performance of the proposed BVNS-RL and BVNS-R algorithms.

5 Conclusions

In this paper, we have proposed and evaluated a Variable Neighborhood Search algorithm for the Uncapacitated Facility Location Problem. The methodology is built upon a Basic VNS framework that strikes a balance between intensification and diversification, using an improvement phase based on closing facilities under two different strategies, and a perturbation mechanism that opens new ones. We explored two distinct constructive heuristics to generate the initial solutions: a simple random approach and a more sophisticated method guided by a Reinforcement Learning mechanism.

The computational results analyzed a total number of 802 instances from well-known benchmarks including the large-scale K90 and G700 sets. The experiments demonstrated the effectiveness and efficiency of our proposed VNS approach. Both variants of our algorithm deliver highly competitive solutions, achieving near-zero deviation from the verified optimal solutions. In comparison to the state-of-the-art EGTOA algorithm, our methods show vastly superior performance, particularly on larger and more complex instances where EGTOA's solution quality degraded significantly. The analysis also revealed that the Reinforcement Learning constructive (BVNS-RL) consistently provided a slight but clear improvement over the random constructive (BVNS-R) in all performance metrics, highlighting the value of incorporating a learning component to guide the search towards promising regions of the solution space.

For future work, several research directions are identified. One practical avenue is to enhance computational efficiency by investigating methods to accelerate the objective function calculation, which is a bottleneck during the intensive local search phase. Another promising direction is to further explore the integration of different Machine Learning techniques. This could involve not only developing more advanced constructive heuristics but also applying Machine Learning to other parts of the algorithm, such as the dynamic selection of neighborhoods or guiding the perturbation strategy, thereby further enhancing the algorithm's performance and adaptability.

Acknowledgment. This work was made possible through the support of the Comunidad Autónoma de Madrid (grant ref. TEC-2024/COM-404), Ministerio de Economía y Competitividad (grants ref. PID2021-125709OA-C22, PID2019-104410RB-I00/AEI/10.13039/501100011033), and Ministerio para la Transformación Digital y de la Función Pública (Cátedra ENIA AI4DDS, grant ref. TSI-100930-2023-3).

References

1. Akinc, U., Khumawala, B.M.: An efficient branch and bound algorithm for the capacitated warehouse location problem. Manage. Sci. **23**(6), 585–594 (1977). https://doi.org/10.1287/mnsc.23.6.585
2. Beasley, J.E.: Or-library: distributing test problems by electronic mail. J. Oper. Res. Soc. **41**(11), 1069–1072 (1990). http://www.jstor.org/stable/2582903
3. Charikar, M., Guha, S.: Improved combinatorial algorithms for the facility location and k-median problems. In: 40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039), pp. 378–388 (1999). https://doi.org/10.1109/SFFCS.1999.814609
4. Djenić, A., Radojičić, N., Marić, M., Mladenović, M.: Parallel VNS for bus terminal location problem. Appl. Soft Comput. **42**, 448–458 (2016). https://doi.org/10.1016/j.asoc.2016.02.002
5. Drezner, Z., Hamacher, H.: Facility Location: Applications and Theory. Springer, Heidelberg (2002)
6. Efroymson, M.A., Ray, T.L.: A branch-bound algorithm for plant location. Oper. Res. **14**(3), 361–368 (1966). https://doi.org/10.1287/opre.14.3.361
7. Fleszar, K., Hindi, K.: An effective VNS for the capacitated p-median problem. Eur. J. Oper. Res. **191**(3), 612–622 (2008). https://doi.org/10.1016/j.ejor.2006.12.055
8. Galvão, R.D., Raggi, L.A.: A method for solving to optimality uncapacitated location problems. Ann. Oper. Res. **18**(1), 225–244 (1989). https://doi.org/10.1007/bf02097805
9. García-López, F., Melián-Batista, B., Moreno-Pérez, J.A., Moreno-Vega, J.M.: The parallel variable neighborhood search for the p-median problem. J. Heuristics **8**(3), 375–388 (2002). https://doi.org/10.1023/a:1015013919497
10. Hansen, P., Mladenović, N.: Variable neighborhood search: principles and applications. Eur. J. Oper. Res. **130**(3), 449–467 (2001). https://doi.org/10.1016/S0377-2217(00)00100-4
11. He, Y., Wang, X.: Group theory-based optimization algorithm for solving knapsack problems. Knowl. Based Syst. **219**, 104445 (2021). https://doi.org/10.1016/j.knosys.2018.07.045
12. Janáček, J., Buzna, Ľ: An acceleration of erlenkotter-körkel's algorithms for the uncapacitated facility location problem. Ann. Oper. Res. **164**(1), 97–109 (2008). https://doi.org/10.1007/s10479-008-0343-0
13. Kalatzantonakis, P., Sifaleras, A., Samaras, N.: A reinforcement learning-Variable neighborhood search method for the capacitated Vehicle Routing Problem. Exp. Syst. Appl. **213**, 118812 (2023). https://doi.org/10.1016/j.eswa.2022.118812
14. Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A.M., Talbi, E.G.: Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: a state-of-the-art. Eur. J. Oper. Res. **296**(2), 393–422 (2022). https://doi.org/10.1016/j.ejor.2021.04.032
15. Kiran, M.S.: The continuous artificial bee colony algorithm for binary optimization. Appl. Soft Comput. **33**, 15–23 (2015). https://doi.org/10.1016/j.asoc.2015.04.007
16. Kuehn, A., Hamburger, M.: A heuristic program for locating warehouses. Manage. Sci. **9**(4), 643–666 (1963). https://doi.org/10.1287/mnsc.9.4.643
17. Lodi, A., Mossina, L., Rachelson, E.: Learning to handle parameter perturbations in Combinatorial Optimization: an application to facility location. EURO J. Transp. Logist. **9**(4), 100023 (2020). https://doi.org/10.1016/j.ejtl.2020.100023

18. Lozano-Osorio, I., Sánchez-Oro, J., López-Sánchez, A.D., Duarte, A.: A variable neighborhood search for the median location problem with interconnected facilities. Int. Trans. Oper. Res. **32**(1), 69–89 (2025). https://doi.org/10.1111/itor.13468
19. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997). https://doi.org/10.1016/s0305-0548(97)00031-2
20. Rahkar Farshi, T.A., Agahian, S., Dehkharghani, R.: BinBRO: binary battle royale optimizer algorithm. Exp. Syst. Appl. **195**, 116599 (2022). https://doi.org/10.1016/j.eswa.2022.116599
21. Resende, M.G., Werneck, R.F.: A hybrid multistart heuristic for the uncapacitated facility location problem. Eur. J. Oper. Res. **174**(1), 54–68. https://doi.org/10.1016/j.ejor.2005.02.046
22. Singh, R., Mangat, N.S.: Stratified sampling. In: Singh, R., Mangat, N.S. (eds.) Elements of Survey Sampling, pp. 102–144. Springer, Dordrecht (1996). https://doi.org/10.1007/978-94-017-1404-4_5
23. Zhang, F., He, Y., Ouyang, H., Li, W.: A fast and efficient discrete evolutionary algorithm for the uncapacitated facility location problem. Exp. Syst. Appl. **213**, 118978 (2023). https://doi.org/10.1016/j.eswa.2022.118978