



Solving the uncapacitated facility location problem under uncertainty: a hybrid tabu search with path-relinking simheuristic approach

David Peidro¹ · Xabier A. Martin¹ · Javier Panadero² · Angel A. Juan¹

Accepted: 31 March 2024 / Published online: 24 April 2024
© The Author(s) 2024

Abstract

The uncapacitated facility location problem (UFLP) is a well-known combinatorial optimization problem that finds practical applications in several fields, such as logistics and telecommunication networks. While the existing literature primarily focuses on the deterministic version of the problem, real-life scenarios often involve uncertainties like fluctuating customer demands or service costs. This paper presents a novel algorithm for addressing the UFLP under uncertainty. Our approach combines a tabu search metaheuristic with path-relinking to obtain near-optimal solutions in short computational times for the deterministic version of the problem. The algorithm is further enhanced by integrating it with simulation techniques to solve the UFLP with random service costs. A set of computational experiments is run to illustrate the effectiveness of the solving method.

Keywords Facility location problem · Uncertainty · Tabu search · Path-relinking · Simheuristics

1 Introduction

The facility location problem (FLP) was initially addressed by Stollsteimer [1], Kuehn and Hamburger [2], Manne [3] and Balinski [4] to determine a set of facilities that minimizes the aggregation of two inversely correlated costs: (i) the cost of opening facilities; (ii) the cost related to servicing customers from the opened facilities. In most formulations of the problem, a set of customers and a set of potential facility locations are known in advance. Likewise, the opening costs associated with each facility and the costs of servicing each customer from every potential facility are also known. Hence

all inputs are deterministic insofar as they are static inputs that are given from the very beginning. So the FLP is a frequent optimization problem used in very diverse application fields, from logistics and inventory planning (e.g., where to allocate distribution or retailing centers in a supply chain) to telecommunication and computing networks (e.g., where to allocate cloud service servers in a distributed network, cabinets in optical fiber networks, etc.).

As one of the most frequent optimization problems in the logistics and supply chain management area, several versions of the FLP have been analyzed in the scientific literature. The uncapacitated FLP (UFLP) assumes that each facility's capacity is virtually unlimited or is, at least, far beyond expected customer demands. Despite being known as the simple facility location problem, the simple warehouse location problem, or the simple plant location problem for its apparent simplicity [5], the UFLP has been proved to be *NP-hard* [6]. Therefore, heuristic and metaheuristic approaches become a natural choice for solving large-scale instances of the UFLP in reasonably short computing times. This is because exact and approximate methods are unable to accurately solve large instances during short periods of time. Another variant is the capacitated FLP (CFLP), where each open facility has a limited servicing capacity; i.e., there is limit to customers' demand that can be served by any single facility. According to Silva and de la Figuera [7], both Lagrangian-based heuristics and metaheuristics have been

✉ David Peidro
dapeipa@upv.es

Xabier A. Martin
xamarsol@upv.es

Javier Panadero
javier.panadero@uab.cat

Angel A. Juan
ajuanp@upv.es

¹ Research Center on Production Management and Engineering, Universitat Politècnica de València, Plaza Ferrandiz Carbonell, Alcoy 03801, Spain

² Department of Computer Architecture and Operating Systems, Universitat Autònoma de Barcelona, Carrer de les sitges, Bellaterra 08193, Spain

demonstrated as effective methods for solving the CFLP. The single-source CFLP (SSCFLP) also requires each customer having to be supplied by exactly one facility. As stated by Klose and Drexl [8], the SSCFLP is generally more difficult to solve than the multiple-source CFLP. In fact the SSCFLP also belongs to the class of *NP-hard* problems [9]. Therefore, heuristic approaches become a natural choice for solving large-scale instances of the SSCFLP in reasonably short computing times.

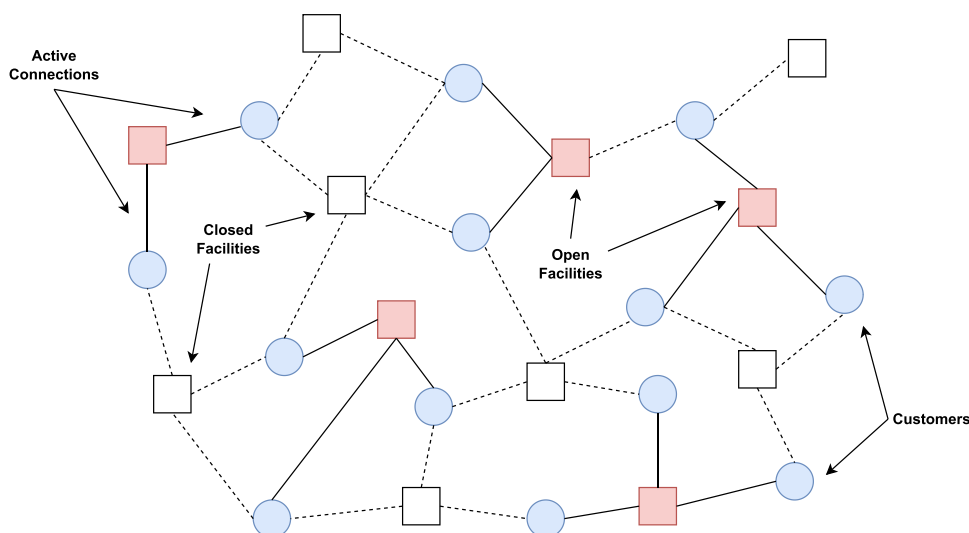
Uncertainty permeates all real-world systems, including supply chain management, logistics and facility locations. The inherent unpredictability of various variables poses a significant challenge in designing efficient and cost-effective solutions. Correia and Saldanha-da Gama [10] reviewed the FLP with stochastic components by exploring different methods proposed in the recent literature to optimize the FLP under uncertainty. These stochastic components arise when inputs, such as customer demands or service costs, are random variables instead of deterministic values. Effectively recognizing and accounting for this uncertainty become crucial in providing optimal solutions for real-life combinatorial optimization problems. Simulation-based optimization approaches have been proposed to tackle such problems [11, 12]. These approaches encompass diverse optimization methods, including mathematical programming, metaheuristics, and even machine learning. In recent decades, the hybridization of metaheuristics with simulation has emerged as a popular and effective approach for solving stochastic optimization problems [12]. Simheuristics, a simulation-optimization method that combines simulation with metaheuristics, has been widely used to address various combinatorial optimization problems with stochastic elements [13, 14]. This work proposes a novel simheuristic algorithm to address the UFLP under uncertainty by specifically considering stochastic components in the form

of random service costs. These service costs can be modeled using probability distributions that can be either theoretical or empirical. Figure 1 provides an illustrative example of a solution for the UFLP under uncertainty. The example depicts the locations where facilities have been opened (indicated by red squares) and the closed facilities (represented by white squares). Customers, depicted by blue circles, are served by the open facilities to which they are actively connected. Each facility has a fixed opening cost, and servicing a customer throughout an open facility has an associated service cost, which is uncertain and, therefore, modeled as a random variable.

Accordingly, the main contributions of this paper can be summarized as follows: (i) a tabu search metaheuristic [15] to efficiently solve the UFLP in short computational times; (ii) a path-relinking approach to obtain near-optimal solutions by exploring paths that connect good quality solutions; (iii) a simheuristic algorithm that integrates the tabu search metaheuristic and path-relinking approach with simulation techniques to efficiently solve the aforementioned problem. Note that the tabu search and path-relinking algorithms are shown to be highly effective in practice for finding near-optimal or optimal solutions to large-scale optimization instances in short computational times [16, 17]. However, without the support of simulation-based extensions like that introduced in this paper, these techniques do not account for the inherent unpredictability found in real-life systems when solving the UFLP under uncertainty.

The remainder of this paper is structured as follows: Section 2 formulates the mathematical model for the UFLP with random service costs. Section 3 reviews the literature on the UFLP under uncertainty. Section 4 presents a tabu search metaheuristic with path-relinking as a solving method for the UFLP, and how the algorithm can be extended into a simheuristic one to solve the UFLP under uncertainty.

Fig. 1 Illustrative example of a solution for the UFLP under uncertainty



Section 5 describes the computational study performed to test the proposed solving method. Finally, Section 6 discusses the main conclusions of this work and open research lines.

2 Model formulation

By following the mixed-integer linear programming model for the deterministic UFLP proposed by Erlenkotter [18], the stochastic version of the problem can be modeled using the following set of parameters and variables

- I - Set of m alternative facility locations, indexed by i .
- J - Set of n customer zones, indexed by j .
- f_i - Fixed cost of establishing a facility at location i .
- C_{ij} - Random variable that models the cost of servicing customer j from facility i .
- x_{ij} - It takes the value 1 if the demand of customer j is supplied from facility i , and 0 otherwise.
- y_i - It takes the value 1 if facility i is open, and 0 otherwise.

Then the stochastic UFLP considered in this work is formulated as follows:

$$\text{Minimize} \quad E \left[\sum_{i \in I} \sum_{j \in J} C_{ij} x_{ij} \right] + \sum_{i \in I} f_i y_i \quad (1)$$

subject to:

$$\sum_{i \in I} x_{ij} = 1 \quad \forall j \in J \quad (2)$$

$$x_{ij} \leq y_i \quad \forall i \in I, \forall j \in J \quad (3)$$

$$P(C_{ij} \leq c_{max}) > p_0 \quad \forall i \in I, \forall j \in J \quad (4)$$

$$y_i \in \{0, 1\} \quad \forall i \in I \quad (5)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in I, \forall j \in J \quad (6)$$

The objective function (1) represents the minimization of the sum of the expected total cost of servicing customers and the total fixed costs of facility establishment.

Constraint (2) ensures that the demand for each customer zone is met. Constraint (3) makes sure that customer demand can only be produced and shipped where the facility is established. Constraint (4) features probabilistic constraints. These constraints add a probabilistic aspect to the model by ensuring that the cost of serving each customer from each facility falls within a certain threshold c_{max} with a specified level of confidence p_0 . This allows the model to account for the uncertainty in the cost values and helps more robust decisions to be made that consider the risk associated with the random variable C_{ij} . As discussed later, this can be considered a soft constraint, which will generate a penalty cost

every time it is violated. Finally, Constraints (5) and (6) define the binary decision variables. The deterministic version of this model (considering constant costs c_{ij} instead of variable ones C_{ij}) was implemented using Python and solved with commercial solver Gurobi to calculate the optimal values for different instances of the deterministic UFLP. In this way, the performance of the metaheuristic proposed in this work can be compared to the mathematical approach. Additionally, a simheuristic was developed to solve the stochastic version of the problem at different uncertainty levels, which is a very common situation in real-life problems.

3 Related work on the UFLP under uncertainty

The FLP was initially introduced as the plant location problem by Stollsteimer [1] and Balinski [4]. Traditionally, the FLP has been approached from several perspectives, including worst-case analysis, probabilistic analysis and empirical heuristics. Although exact algorithms for the problem can be found in the existing literature, the NP-hard nature of the FLP makes heuristics a more practical approach for quickly obtaining solutions, especially for larger and more realistic instances. One of the first works on the FLP was carried out by Efraymson and Ray [19], who developed a branch-and-bound algorithm. They utilize a compact formulation of the FLP by leveraging the fact that its linear programming relaxation could be solved through inspection. However, this linear programming relaxation is known to be weak and does not, therefore, provide tight lower bounds. Another early approach was proposed by Spielberg [20], which employs a direct search or implicit enumeration method. The authors present two different algorithms based on the same directed search: one considering the facilities to be initially open and another one contemplating initially closed ones. Later Erlenkotter [18] proposed a dual-based exact approach that differed from previous approaches by considering a dual objective function. An improved version of this work was presented by Körkel [21]. Although exact approaches provide optimal solutions for small or medium instances, they are unsuitable for solving large-scale real-world problems in reasonable computational times. Therefore, employing approximate methods is advisable. One of the earliest approximation methods was that by Hochbaum [22], and it consists of a simple and fast greedy heuristic. More recently, Ghosh [23] put forward a neighborhood search heuristic by incorporating a tabu search as the local search component. This approach yields competitive solutions in significantly shorter computational times compared to exact algorithms. A similar approach with a tabu search was proposed by Michel and Van Hentenryck [24]. This algorithm shows competitive performance compared

to previous literature results. The algorithm utilizes a linear neighborhood approach, where a single facility is flipped at each iteration. Resende and Werneck [25] introduced an algorithm based on the GRASP metaheuristic. This algorithm incorporates a greedy construction phase combined with a local search operator and path-relinking. It obtains results that come close to the lower bound values for a wide range of different instances. Recently, Martins et al. [26] introduced an ‘agile optimization’ framework for the UFLP. This framework combines a biased-randomized algorithm with parallel programming techniques to offer real-time solutions. The key feature of this approach is its ability to react and swiftly adapt to rapidly changing customer demands. It achieves this by re-optimizing the system whenever new information is incorporated into the model.

The FLP was initially introduced as the plant location problem by Stollsteimer [1] and Balinski [4]. Traditionally, the FLP has been approached through various perspectives, including worst-case analysis, probabilistic analysis, and empirical heuristics. While exact algorithms for the problem can be found in the existing literature, the NP-hard nature of the FLP makes heuristics a more practical approach for obtaining solutions quickly, especially for larger and more realistic instances. One of the first works on the FLP was carried out by Efraymson and Ray [19] who developed a branch-and-bound algorithm. They utilized a compact formulation of the FLP, leveraging the fact that its linear programming relaxation could be solved through inspection. However, this linear programming relaxation is known to be weak and therefore does not provide tight lower bounds. Another early approach was proposed by Spielberg [20], which employed a direct search or implicit enumeration method. The authors present two different algorithms based on the same directed search, one considering the facilities initially open and another one considering the facilities initially closed. Later, Erlenkotter [18] propose a dual-based exact approach, differing from previous approaches by considering a dual objective function. An improved version of this work was presented by Körke [21]. Although exact approaches provide optimal solutions for small or medium instances, they are unsuitable for solving large-scale real-world problems in reasonable computational time. Therefore, it is advisable to employ approximate methods. One of the earliest approximation methods was proposed by Hochbaum [22], consisting of a simple and fast greedy heuristic. More recently, Ghosh [23] proposed a neighborhood search heuristic, incorporating tabu search as the local search component. This approach yields competitive solutions within significantly reduced computational times compared to exact algorithms. A similar approach using a tabu search was proposed by Michel and Van Hentenryck [24]. This algorithm has shown competitive performance compared to previous literature results. The algorithm utilizes a linear neighborhood approach, where a

single facility is flipped at each iteration. Resende and Werneck [25] introduced an algorithm based on the GRASP metaheuristic. This algorithm incorporates a greedy construction phase combined with a local search operator, and a path-relinking. It achieved results close to the lower bound values for a wide range of different instances. Recently, Martins et al. [27] introduced an Agile Optimization framework [26] for the UFLP. This framework combines a biased-randomized algorithm with parallel programming techniques to offer real-time solutions. The key feature of this approach is its ability to react and adapt swiftly to rapidly changing customer demands. It achieves this by re-optimizing the system whenever new information is incorporated

A wide range of variants of the FLP have been extensively addressed in the literature, regardless of the employed solution method. One of the most studied variants is the SSCFLP. Similarly to the original version of the FLP, several exact methods have also been proposed to solve small instances of this problem. For example, Holmberg et al. [28] described a matching algorithm incorporated into a Lagrangian heuristic. In another study, Díaz and Fernández [29] developed an exact algorithm that integrates a column generation procedure for finding upper and lower bounds within a branch-and-price framework. Similarly, Yang et al. [9] introduced an exact algorithm based on a cut-and-solve framework designed explicitly for the SSCFLP. Regarding the use of approximate methods, Chen and Ting [30] proposed a hybrid algorithm that combines a Lagrangian-based heuristic with an ant colony method. This approach aims to leverage the strengths of both techniques to solve the problem. Ahuja et al. [31] presented a large-scale neighborhood search algorithm, which focuses on efficiently exploring the solution space of the problem. This approach allows large-scale instances to be effectively handled in a reasonable computational time. Filho and Galvão [32] proposed a tabu search metaheuristic that is also able to handle large instances, while Delmaire et al. [33] put forward a more sophisticated algorithm also based on a tabu search, which combines a reactive GRASP algorithm with a tabu search. This algorithm incorporates elements of both techniques to enhance search capabilities and to find high-quality solutions. Finally, Estrada-Moreno et al. [34] presented a biased-randomized iterated local search metaheuristic to solve the SSCFLP with soft capacity constraints. This variant of the SSCFLP assumes that the maximum capacity at each facility can be potentially exceeded by incurring a penalty cost, which increases with the constraint-violation gap. For a more comprehensive literature review on the FLP and its variants, readers are referred to [35], [8], and the book edited by Eiselt and Marianov [36] covers many relevant works on the FLP.

In real life, the inputs of combinatorial optimization problems are typically nondeterministic. This means that they are subject to random events; i.e., random failures of

some components, stockouts due to random demands, etc. Therefore, Simulation-based optimization approaches are required. These methods aim to find a solution that performs well for any possible realization of the random variables, i.e., a robust solution that can handle variations and fluctuations in the problem parameters that may occur in real-world scenarios s [37]. Although the stochastic FLP has not been paid significant attention in the literature, we can find several works, some of which have been published only a few years after the problem's definition. For example, Balachandran and Jain [38] presented a stochastic FLP model with piece-wise linear production. This model takes demands to be random and continuous variables. Later Correia and Saldanha-da Gama [10] examined distinct modeling frameworks for facility location under uncertain conditions, which specifically distinguish among robust optimization, stochastic programming and chance-constrained models. Another simulation-optimization approach to address this problem is the simheuristics [39], which is a promising approach based on the combination of simulation with metaheuristics, used for solving efficiency different combinatorial optimization problems with stochastic elements. Indeed simheuristics has been used by different authors to solve the UFLP under uncertainty. De Armas et al. [40] proposed a simheuristic approach to address the UFLP with random service costs. This simheuristics combines an ILS metaheuristic with Monte Carlo simulation to deal with uncertainty by providing flexibility to consider diverse optimization objectives beyond minimizing the expected cost. A similar approach is that proposed in Quintero-Araujo et al. [41], which proposes a SimILS framework, a simheuristic algorithm that combines Monte Carlo simulation with a biased-randomized metaheuristic algorithm to solve the capacitated location routing problem (CLRP) with stochastic demands. Another interesting approach is that reported Bayliss and Panadero [42], where a learnheuristic algorithm [43] is presented to solve the temporary-facility location and queuing problem. It integrates a biased randomization algorithm with simulation and a machine-learning component to tackle not only uncertain components, but also dynamics components of problems.

To further extend the landscape of facility location problems under uncertainty, other methodological approaches have emerged in recent years. Marques and Dias [44] introduced a dynamic UFLP by considering uncertainty in fixed and assignment costs, customer and facility locations. They aimed to minimize the expected total cost, while explicitly considering regret. Regret, as a measure of loss for not choosing an optimal scenario solution, is upper-bounded. Their mixed integer programming model and solution approach demonstrates potential through illustrative examples and computational results. Another exciting approach was that proposed by Zhang et al. [45], which tackles another variant, the squared metric two-stage stochastic FLP by focusing

on uncertainty in client sets and facility costs with squared metric connection costs. They proposed a new integer linear programming formulation and evaluated two algorithms' performance by analyzing approximation ratios and per-scenario bounds. Moreover, Ramshani et al. [46] explored disruptions in two-level UFLPs by including additional facilities between customers and the main facilities, and by acknowledging disruptions' impact on facility reliability for meeting customer demands. They developed mathematical formulations and algorithms, such as a tabu search and a problem-specific heuristic, to address disruptions in a two-level distribution chain scenario. By extending this research, Koca et al. [47] addressed two-stage stochastic UFLPs by emphasizing system nervousness. They introduced models to consider uncertain location and allocation decisions adaptable to realizations of uncertain parameters. Their proposed models incorporated restricted recourse to control deviations between first-stage and scenario-dependent allocation decisions by showcasing Benders-type decomposition algorithms and computational enhancements.

4 Proposed methodology

Simulation methods are frequently employed by experts to address stochastic uncertainty because they enable several scenarios to be analyzed to support decision-making processes. However, it is important to note that simulation itself is not an optimization tool. Therefore, hybrid simulation-optimization methodologies have been proposed to efficiently cope with large-scale optimization problems under uncertainty. One such simulation-optimization method is simheuristics, which combines metaheuristics with simulation [48]. Its efficiency as a method for solving different combinatorial optimization problems with stochastic elements has been shown in several studies [43]. This success can be attributed to the method's ability to evaluate solutions using simulation and problem-specific analytical expressions. Simheuristic algorithms are 'white-box' approaches designed specifically to solve large-scale and NP-hard combinatorial optimization problems with stochastic elements, which can come in the form of stochastic objective functions or probabilistic constraints [49]. The proposed simheuristic method combines a tabu search metaheuristic with path-relinking, which is then integrated with Monte Carlo simulation (MCS) to solve the UFLP with random service costs. Using the tabu search metaheuristic with path-relinking combination allows us to obtain near-optimal solutions to the deterministic version of the problem in short computational times. This combination is successfully employed to solve NP-hard optimization problems in vehicle routing and scheduling domains [50, 51]. The algorithm is further combined with simulation techniques to guide the algorithm

during the search for near-optimal solutions to the stochastic version of the problem.

4.1 Tabu search with path-relinking

The algorithm for a tabu search with path-relinking is now described. We represent a state in the tabu search by vector y_i as defined in the model formulation. It is a natural choice seeing that the facility locations are the only combinatorial component. To solve the UFLP, it is enough to know the set of open facilities because customer zones are connected to the cheapest open facilities.

As shown in Algorithm 1, three sections can be identified. The first section applies the tabu search metaheuristic N times to the same problem instance, but uses different seeds. The best solutions are stored in a pool of elite solutions of size L . Subsequently, the path-relinking technique is applied for each pair of elements in the elite solution pool. The new solutions produced to create the next pool of elite solutions are employed (new generation pool). Only those solutions that improve the two used in the path-relinking process are considered the candidates to be inserted into the new generation pool of size L . Note that the best solution found so far is inserted directly into the new generation of elite solutions (1-elitism). The criterion for inserting a new solution into the elite solutions pool is the lowest total cost. The creation of new pools is repeated as long as new generation pools of at least two solutions are produced. After the path-relinking phase, the algorithm begins a local search process around sol_{best} . This procedure consists of applying a well-known 2-opt local search to every possible combination of two facility locations in the solution.

The tabu search is presented in Algorithm 2. First, a random solution is generated to provide each facility location with an opening probability. The neighborhood consists in flipping the state of a facility from open to closed, or vice versa. After scanning the neighborhood, if there is an improvement is calculated. Then the facility location involved in the improvement is added to the tabu list to prevent it from being chosen for a defined number of iterations (tenure). The tabu list is implemented by associating a simple counter $tabu[i]$ for each facility i . When a facility is inserted on the tabu list, the counter $tabu[i]$ is updated by adding the actual iteration count, plus the tenure ($tabu[i] = iteration + tenure$). This expression states that the facility will remain in the tabu search memory for as many iterations as the value of the tenure parameter. The algorithm can quickly check if a facility is on the tabu list if $tabu[i] > iteration$. If there is no improvement in the solution, the algorithm randomly closes a previously open facility location. If only one site remains to be closed, a new one is randomly opened. Tenure is adjusted during the search process. If the algorithm finds improvements in the neigh-

Algorithm 1 Tabu search with path-relinking

```

1 Function TSPR(instanceData,  $N$ ,  $L$ )
2    $eliteSols \leftarrow \emptyset$ 
3   for  $i \leftarrow 1$  to  $N$  do
4      $seed \leftarrow getNewSeed()$ 
5      $sol_{new} \leftarrow TS(seed, instanceData)$  /* Tabu
      search */
6     if  $|eliteSols| < L$  then
7        $insert(eliteSols, sol_{new})$ 
8     else if  $|eliteSols| = L$  and
        $cost(sol_{new}) < worstCost(eliteSols)$  then
9        $sol_{worst} \leftarrow getWorstSol(eliteSols)$ 
10       $replace(eliteSols, sol_{worst}, sol_{new})$ 
11  repeat
12     $newEliteSols \leftarrow \emptyset$ 
13     $sol_{best} \leftarrow getBestSol(eliteSols)$ 
14     $insert(newEliteSols, sol_{best})$ 
15    foreach  $sol_O \in eliteSols$  do
16      foreach  $sol_R \in eliteSols$  do
17        if  $sol_O \neq sol_R$  then
18           $sol_{new} \leftarrow PR(sol_O, sol_R)$ 
19          /* Path-relinking */
20          if  $cost(sol_{new}) <$ 
21             $min(cost(sol_O), cost(sol_R))$  then
22              if  $|newEliteSols| < L$  then
23                 $insert(newEliteSols, sol_{new})$ 
24              else if  $|newEliteSols| = L$  and
                 $cost(sol_{new}) <$ 
                 $worstCost(newEliteSols)$  then
25                 $sol_{worst} \leftarrow$ 
26                 $getWorstSol(newEliteSols)$ 
27                 $replace(newEliteSols, sol_{worst},$ 
28                 $sol_{new})$ 
29     $eliteSols \leftarrow copyFrom(newEliteSols)$ 
30  until  $|newEliteSols| < 2$ 
31   $sol_{best} \leftarrow LS(sol_{best})$  /* 2-opt local search
      */
32  return  $sol_{best}$ 

```

borhood, it decreases tenure to improve exploitation. On the contrary, if it does not find improvements when changing the state of any of the facility locations of a solution, tenure is increased to improve search diversification (never without exceeding a given maximum value). The process is repeated until the stopping criterion is met, which consists in exceeding a number M of iterations without finding an improvement in the solution. The tabu search algorithm uses three different pieces of information for each customer zone j : the number of open facilities that offer the cheapest connection to j , the cost of that connection, and the cost of the second cheapest connection to an open facility. These details allow the gains from opening and closing a facility to be incrementally updated by eliminating the need to compute the entire objective function for a new solution (line 10). A similar approach is used in Michel and Van Hentenryck [24].

Algorithm 2 Tabu search

```

1 Function TS (seed, instanceData)
2   solbest ← randomSolution(seed)
3   bestCost ← cost(solbest)
4   failedRuns ← 0
5   tabuList ← ∅
6   repeat
7     costFlip ← ∞
8     foreach i ∈ I do
9       if i ∉ tabuList then
10        costi ← cost(flip(solini, i))
11        if costi < costFlip then
12          costFlip ← costi
13          flipFac ← i
14      if costFlip < bestCost then
15        solbest ← flip(solbest, flipFac)
16        bestCost ← costFlip
17        insert(tabuList, flipFac)
18        decreaseTenure()
19        failedRuns ← 0
20      else
21        increaseTenure()
22        failedRuns ← failedRuns + 1
23        if countOpened(solbest) > 2 then
24          closeRandomFacility()
25        else
26          openRandomFacility()
27  until failedRuns ≥ M
28  return solbest

```

Algorithm 3 Path-relinking

```

1 Function PR (solO, solR)
2   if cost(solO) ≤ cost(solR) then
3     solbest ← solO
4   else
5     solbest ← solR
6   foreach i ∈ I do
7     if solO[i] ≠ solR[i] then
8       cardinality ← cardinality + 1
9   while cardinality > 0 do
10    solbestMove ← allOpen()
11    foreach i ∈ I do
12      if solO[i] ≠ solR[i] then
13        soltemp ← solO
14        flip(soltemp, i)
15        if cost(soltemp) < cost(solbestMove) then
16          solbestMove ← soltemp
17        if cost(soltemp) < cost(solbest) then
18          solbest ← soltemp
19    solO ← solbestMove
20    cardinality ← cardinality - 1
21  return solbest

```

The path-relinking subroutine is comprehensively described in Algorithm 3. Given the origin and reference solutions, we first calculate which one gives the best cost and store it in sol_{best} . Then cardinality is calculated as the number of differences between the origin and reference solutions. Next, the algorithm starts from sol_O and gradually transforms into sol_R by flipping the facilities that are different between the two solutions. During this transformation process, intermediate solutions are evaluated to see if they improve the overall result. Finally, the best found solution is returned.

The algorithm's computational complexity is next reported. For the tabu search component, the complexity per iteration is $O(n \cdot \log(m))$, where n represents the number of clients and m denotes the number of facilities. This complexity accounts for the calculation of the cost of flipping each facility in a solution, while tracking the best and second-best facilities to assign clients to. These operations are repeated until a number of failed runs is reached, which contributes to the constant factor of the computational complexity. In other words, as the number of iterations is constant (not dependent on the size of the problem), the overall computational complexity of the tabu search component is expressed as $O(n \cdot \log(m))$. The path-relinking component has a similar complexity per iteration of $O(n \cdot \log(m))$. This also involves updates to the best and second-best facilities assigned to clients, because intermediate solutions have to be evaluated to see if they improve current solutions. Similar to the tabu search component, the actual number of iterations performed by the path-relinking component depends on the cardinality of solutions along with the number of elite generations, which are not dependent on the size of the instance. Thus, the overall computational complexity of the path-relinking component is $O(n \cdot \log(m))$. The local search component, the computational complexity per iteration is associated with the process of exploring the neighborhood of the current solution. This involves flipping each facility in the solution and checking if the new solution is improved, whose computational complexity is $O(n \cdot \log(m))$. This process is repeated until no further improvements are found, so the overall computational complexity of the local search is $O(n \cdot \log(m))$. Finally, as the three components discussed are executed in a serial manner, we can conclude the proposed algorithm has a worst-case computational complexity of $O(n \cdot \log(m))$.

4.2 The simheuristic framework

Figure 2 illustrates a flow chart of our simheuristic methodology for dealing with the UFLP under stochastic uncertainty. Every time a new solution is generated by the tabu search, path-relinking or a local search, it is simulated with a few runs to obtain an estimate of the solution's average stochastic cost. On the one hand, regarding the tabu search, all the N times when the algorithm reaches the stopping criterion and

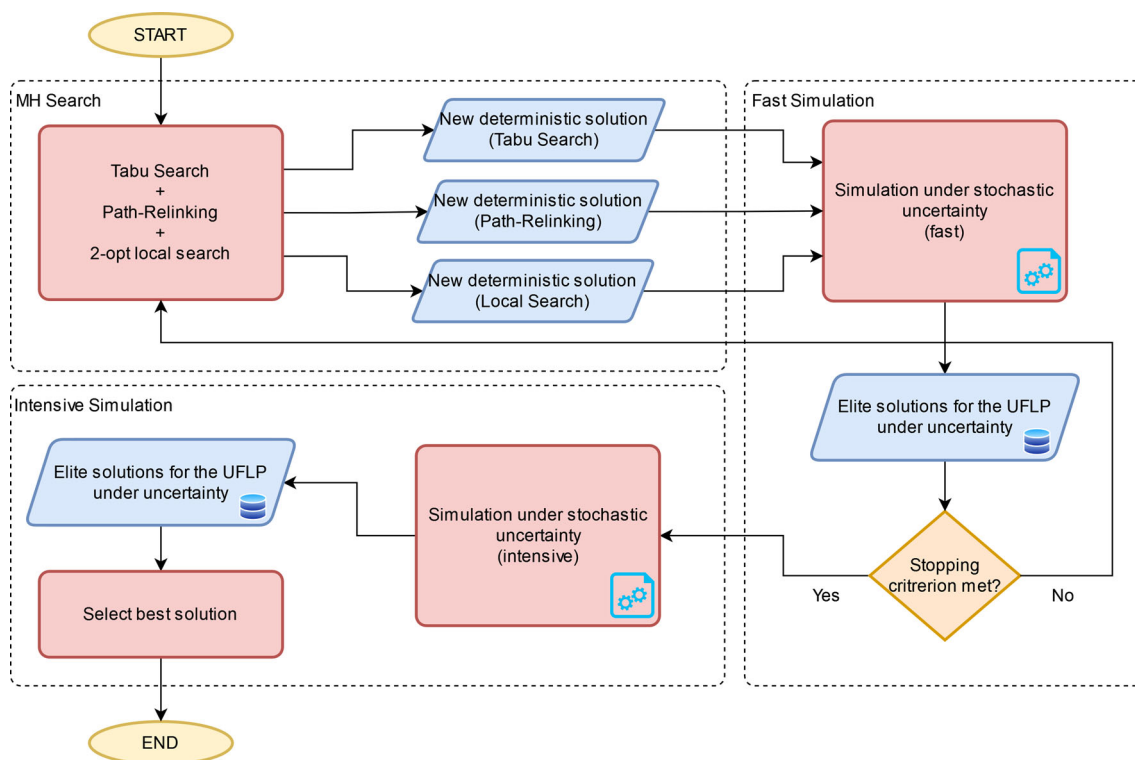


Fig. 2 Flowchart of the simheuristic methodology

finds the best solution (Algorithm 1), the fast simulation is run. On the other hand, the simulations of the path-relinking solutions are the most important in the proposed simheuristic. Path-relinking as an intensive process done on a set of elite solutions allows a sufficiently large set of potential solutions to be explored that could perform well in a stochastic environment. For this reason, every time a new solution is generated during the gradual transformation of sol_O into sol_R , it is simulated regardless of whether it improves the deterministic solution. In addition, whenever the local search procedure reaches a new improved solution, this is also simulated. The solutions with the best-estimated cost are stored in a pool of elite stochastic solutions. Once an overall stopping criterion is met, a limited set of elite stochastic solutions is sent to a more intensive simulation stage to obtain accurate estimates on their behavior in a stochastic scenario. The general stopping criterion consists of satisfying the stopping criteria of each section of the proposed metaheuristic (M failed tabu search runs, a new pool generation with $<$ than 2 solutions during path-relinking and no new improvements during the local search).

Algorithm 4 depicts the main characteristics of our simheuristic algorithm. The code is similar to that shown in Algorithm 1. The solutions obtained by the tabu search are quickly simulated, while those with the lowest expected cost are included in the pool of elite stochastic solutions (lines 13-17). Then when the path-relinking ($PR-SIMH$) and local

search ($LS-SIMH$) functions are called, the pool is passed as an argument to be updated according to the simulations of the newly generated solutions (as explained above). In lines 34-38, intensive simulation is performed to obtain better estimates of the elite solutions. The best solutions are selected according to the lowest expected cost. However, other estimates, such as risk or reliability, may also be used as discussed in Chica et al. [52].

5 Computational study

Several computational experiments were carried out to evaluate and assess the performance of our solving method. To comprehensively illustrate the experiments conducted in this section, a detailed presentation of the utilized benchmark instances, that outlines their characteristics, is first provided. Next the presented benchmark instances are extended to consider random service costs, which model the uncertainty that is inherent in practical scenarios, and penalty costs, by addressing several real-world situations. Then Section 5.4 identifies the significantly affected parameters and decides the optimal parameter settings for the algorithm through the Design of Experiments (DoE). Last, the obtained results and the discussion are presented in Section 5.5. All the numerical experiments were implemented using the C++ programming language. The code was compiled on a Manjaro Linux

Algorithm 4 Simheuristic for the UFLP

```

1 Function
  SIMHUFLP (instanceData, N, L, L2, shortSim, longSim)
2   eliteSols  $\leftarrow$   $\emptyset$ 
3   eliteStochSols  $\leftarrow$   $\emptyset$ 
4   for i  $\leftarrow$  1 to N do
5     seed  $\leftarrow$  getNewSeed()
6     // Tabu search
7     solnew  $\leftarrow$  TS(seed, instanceData)
8     if  $|$ eliteSols $|$   $<$  L then
9        $|$  insert(eliteSols, solnew)
10      else if  $|$ eliteSols $|$  = L and
11        cost(solnew)  $<$  worstCost(eliteSols) then
12           $|$  solworst  $\leftarrow$  getWorstSol(eliteSols)
13           $|$  replace(eliteSols, solworst, solnew)
14          simulation(solnew, shortSim)
15          if  $|$ eliteStochSols $|$   $<$  L2 then
16             $|$  insert(eliteStochSols, solnew)
17            else if  $|$ eliteStochSols $|$  = L2 and
18              expCost(solnew)  $<$ 
19              worstExpCost(eliteStochSols) then
20                 $|$  solworst  $\leftarrow$  getWorstStochSol(eliteSols)
21                 $|$  replace(eliteStochSols, solworst, solnew)
18  repeat
19    newEliteSols  $\leftarrow$   $\emptyset$ 
20    solbest  $\leftarrow$  getBestSol(eliteSols)
21    insert(newEliteSols, solbest)
22    foreach solO  $\in$  eliteSols do
23      foreach solR  $\in$  eliteSols do
24        if solO  $\neq$  solR then
25          // Path-relinking
26          // simheuristic
27          solnew  $\leftarrow$ 
28          PR-SIMH(solO, solR, eliteStochSols)
29          if cost(solnew)  $<$ 
30           $\min$ (cost(solO), cost(solR)) then
31            if  $|$ newEliteSols $|$   $<$  L then
32               $|$  insert(newEliteSols, solnew)
33            else if  $|$ newEliteSols $|$  = L and
34              cost(solnew)  $<$ 
35              worstCost(newEliteSols) then
36                 $|$  solworst  $\leftarrow$ 
37                getWorstSol(newEliteSols)
38                 $|$  replace(newEliteSols, solworst,
39                solnew)
32    eliteSols  $\leftarrow$  copyFrom(newEliteSols)
33  until  $|$ newEliteSols $|$   $<$  2
34  // 2-opt local search simheuristic
35  solbest  $\leftarrow$  LS-SIMH(solbest, eliteStochSols)
36  foreach sol  $\in$  eliteStochSols do
37     $|$  simulation(sol, longSim)
38  eliteSols  $\leftarrow$  sort(eliteSols)
39  topSols  $\leftarrow$  selectTop(eliteSols)
40  return topSols

```

machine with the GCC version 12.3 compiler and executed on a computer with an Intel Core i5-9600K 3.7 GHz and 16 GB RAM.

5.1 Description of benchmark instances

In order to evaluate the performance of the proposed simheuristic algorithm for the UFLP with random service costs, we used the classic instances originally proposed for the p -median problem by Ahn et al. [53], later employed in the UFLP context by Barahona and Chudak [54]. This set of large instances is called MED, which is the most used set of instances in the UFLP literature for being the most challenging ones to solve. To the best of our knowledge, the results of these instances have not been improved from 2006 [25]. This makes them the perfect benchmarks to test the quality of our algorithm. Each instance is composed of a set of n points picked uniformly at random in the unit square. A point represents both a user and facility, and the corresponding Euclidean distance determines service costs. Additionally, each instance is characterized by the following nomenclature $x - y$, where x represents the number of facilities and customers, while y refers to the opening cost scheme. The set of instances consists of six different subsets, each with a different number of facilities and customers (500, 1000, 1500, 2000, 2500 and 3000), and three different opening cost schemes per subset ($\sqrt{n}/10$, $\sqrt{n}/100$, and $\sqrt{n}/1000$ corresponding to 10, 100, and 1000 instance suffixes, respectively). The larger instances and those with suffix 1000 are the most difficult to solve. As they have lower opening costs, the number of open facilities in the solution is bigger, which increases the number of possible combinations and, thus, the complexity of finding the optimal solution.

5.2 Extending benchmark to address uncertainty

To the best of our knowledge, there are no stochastic FLP instances that employ random service costs to be used as a benchmark. So instead of assuming constant service costs c_{ij} ($\forall i \in I, \forall j \in J$), we consider a more realistic scenario in which service costs are modeled as random variables C_{ij} . The service cost represents the cost required to service a customer zone $j \in J$ throughout a facility location $i \in I$ under perfect conditions. Accordingly, we extend the previously described set of instances called MED to assess both the performance and quality of the proposed simheuristic algorithm. The dataset construction process involves transforming the deterministic service costs into random service costs following a probability distribution function. Specifically, the deterministic service costs are transformed into stochastic ones when solutions are sent to the simulation phase. Hence the service costs c_{ij} found in instances are used as the expected values of the random service costs C_{ij} , which are modeled to fol-

low a log-normal probability distribution. The log-normal distribution is a natural choice for describing nonnegative random variables, such as service costs [55]. The log-normal distribution has two parameters: the location parameter, μ_{ij} , and the scale parameter, σ_{ij} , which relate to the expected value $E[C_{ij}]$ and to variance $Var[C_{ij}]$, respectively. Equations 7-8 define how these parameters have been modeled. We assume service cost $C_{ij} \sim \text{LogNormal}(\mu_{ij}, \sigma_{ij})$ with $E[C_{ij}] = c_{ij}$, where c_{ij} is the deterministic service delay found in instances, and variance $Var[C_{ij}] = k \cdot E[C_{ij}]$. Parameter k is a design parameter that allows us to set up the uncertainty level. It determines how much stochastic costs deviate from their expected values (c_{ij} derived from deterministic costs) and, therefore, influences the degree of randomness or uncertainty in the simulated costs. So as k converges to zero, the more the results will move toward the deterministic state, and by increasing the value of k , the results will converge to the stochastic state. The selection of k values depends on the uncertainty level desired in stochastic costs. We consider three different uncertainty levels: low ($k = 5$), medium ($k = 10$) and high ($k = 20$). The main objective of setting different k values is to simulate a range of real-world scenarios for the variability and unpredictability of service costs. By employing multiple k values, the algorithm's performance can be evaluated under diverse conditions, which allows for a comprehensive assessment of its robustness and adaptability to varying cost uncertainty levels.

$$\mu_{ij} = \ln(E[C_{ij}]) - \frac{1}{2} \cdot \ln\left(1 + \frac{Var[C_{ij}]}{E[C_{ij}]^2}\right) \quad (7)$$

$$\sigma_{ij} = \sqrt{\ln\left(1 + \frac{Var[C_{ij}]}{E[C_{ij}]^2}\right)} \quad (8)$$

This approach enables us to test the effectiveness of our simheuristic algorithm in handling UFLP instances with varying uncertainty levels, and to showcase its adaptability at different degrees of random service costs without having to create brand-new datasets.

5.3 Extend benchmark with penalty costs

In order to address real-world situations where certain facilities' costs need to be controlled or limited, a nonlinear penalty cost is added. This approach aligns the optimization process with practical considerations and allows decision makers to make more informed choices regarding facility selection and cost management. Note in the basic UFLP under uncertainty with a linear objective function that the solution which minimizes the total expected cost will be the same as the optimal solution for the deterministic UFLP. This property will not hold if, for instance, a nonlinear penalty cost is added in

the objective function to account for facilities with a total higher service cost than a threshold, etc. Thus we introduce a non linear penalty cost that is twice the cost of opening a facility if the instance maximum service cost of a deterministic solution is exceeded. This parameter, denoted as $c_{\max} = \max c_{ij}$, represents the maximum c_{ij} value between all the facilities opened in the best-found deterministic solution. It is worth noting that when this nonlinear penalty is incorporated into the objective function of the UFLP under uncertainty, the solution that minimizes the total expected cost may no longer be the same as the optimal solution for the deterministic UFLP. This deviation from linearity arises because the penalty cost is included. In other words, the best-known deterministic solution, which will still be optimal in a deterministic scenario for not being affected by the c_{\max} threshold, will perform quite poorly in a stochastic scenario because in many simulation runs there will be service costs that will exceed the c_{\max} threshold, which will result in a severe penalty. Furthermore, the simheuristic algorithm tends to open some more facilities than those in the deterministic solution to avoid the number of times that the threshold is exceeded. Therefore, the simheuristic approach aims to search for solutions that mitigate the impact of penalties associated with exceeding the c_{\max} threshold in multiple simulation runs.

5.4 Parameter analysis by design of experiments

In this section, we use DoE to identify the significantly affected parameters and decide the best combination of each parameter for the TS+PR algorithm. As shown in Table 1, a two-level full factorial design with three replications [56] is adopted to investigate the significant effect of four parameters of the TS+PR algorithm on both the objective value and the calculation time. The following parameters are considered: N (maximum runs of the algorithm with different seeds), M (maximum number of the tabu iterations without improvement), L (elite pool length) and $Tenure$ (number of iterations a move is considered to be tabu, expressed as a percentage of the number of facility locations of the problem). The high and low levels of each parameter are displayed in Fig. 4.

The results are shown in Table 1 for a randomly generated test problem of size ($|I| = |J| = 2000$). The problem was created following the same structure as the large instances called MED. The selected opening cost scheme was $\sqrt{n}/1000$ because, as previously mentioned, it is the most challenging. The response variable AVG GAP (%) is the average gap of 30 different independent runs compared to the optimal solution obtained with Gurobi for the generated test problem. The response variable AVG TIME is the average execution time for the 30 independent runs. In all, $3 * 2^4 * 30 = 1440$ trials were conducted in this experiment.

Table 1 Results of the experiment

Trial	<i>N</i> (A)	<i>M</i> (B)	<i>L</i> (C)	<i>Tenure</i> (D)	<i>Response</i> AVG GAP (%)	<i>Response</i> AVG TIME (s)	Best Gap (%)
1	32	100	10	10%	0.132835	32.76	0.07
2	32	100	10	5%	0.135464	32.44	0.07
3	32	100	5	10%	0.152310	14.8	0.08
4	32	100	5	5%	0.152735	14.89	0.08
5	32	500	10	10%	0.092844	34.23	0.03
6	32	500	10	5%	0.094845	34.79	0.03
7	32	500	5	10%	0.128187	18.18	0.03
8	32	500	5	5%	0.125694	18.18	0.03
9	64	100	10	10%	0.123540	41.97	0.07
10	64	100	10	5%	0.119235	41.86	0.07
11	64	100	5	10%	0.155751	24.24	0.09
12	64	100	5	5%	0.158389	24.27	0.09
13	64	500	10	10%	0.091934	48.18	0.02
14	64	500	10	5%	0.090722	48.59	0.02
15	64	500	5	10%	0.120690	34.84	0.04
16	64	500	5	5%	0.120690	34.79	0.04
17	32	100	10	10%	0.133327	34.43	0.06
18	32	100	10	5%	0.135646	34.44	0.06
19	32	100	5	10%	0.143706	15.89	0.09
20	32	100	5	5%	0.147852	15.91	0.03
21	32	500	10	10%	0.108991	36.22	0.04
22	32	500	10	5%	0.109735	36.26	0.04
23	32	500	5	10%	0.124821	20.17	0.04
24	32	500	5	5%	0.125669	20.16	0.04
25	64	100	10	10%	0.126754	41.26	0.02
26	64	100	10	5%	0.132349	41.03	0.02
27	64	100	5	10%	0.164728	22.82	0.05
28	64	100	5	5%	0.167919	22.79	0.05
29	64	500	10	10%	0.101987	48.07	0.03
30	64	500	10	5%	0.103133	47.83	0.03
31	64	500	5	10%	0.119810	31.99	0.04
32	64	500	5	5%	0.121653	31.92	0.04
33	32	100	10	10%	0.125990	34.62	0.02
34	32	100	10	5%	0.124843	35.45	0.02
35	32	100	5	10%	0.159718	15.57	0.04
36	32	100	5	5%	0.159103	15.57	0.04
37	32	500	10	10%	0.113147	36.22	0.03
38	32	500	10	5%	0.111038	36.55	0.03
39	32	500	5	10%	0.134593	20.37	0.03
40	32	500	5	5%	0.13636	20.39	0.03
41	64	100	10	10%	0.134480	39.38	0.06
42	64	100	10	5%	0.134480	39.8	0.05
43	64	100	5	10%	0.144753	22.98	0.06

Table 1 continued

Trial	<i>N</i> (A)	<i>M</i> (B)	<i>L</i> (C)	<i>Tenure</i> (D)	<i>Response</i> AVG GAP (%)	<i>Response</i> AVG TIME (s)	Best Gap (%)
44	64	100	5	5%	0.143463	22.98	0.06
45	64	500	10	10%	0.098653	47.65	0.03
46	64	500	10	5%	0.098507	47.68	0.03
47	64	500	5	10%	0.126413	31.72	0.04
48	64	500	5	5%	0.128014	31.78	0.04

Table 1 shows the average (best) values for the 30 runs and displays 48 trials.

According to the ANOVA results (Table 2) and the Pareto chart (Fig. 3) for the response AVG GAP (%), only parameters *M* and *L* are statistically significant for the proposed problem. The values RSquare = 92.21%, RSquare(Adj) = 88.19%, and Square(Pred) = 81.33% obtained in Statgraphics mean that this model fits the data well and can be used to accordingly determine the suitable parameters (Fig. 4).

For the response AVG TIME (see Fig. 5), all the parameters except *Tenure* are the algorithm’s most significant and critical parameters at $\alpha = 0.05$. The model fits the data very well (RSquare = 99.25%, RSquare(Adj) = 98.86%, and Square(Pred) = 98.20%).

To find the best parameters for the algorithm, we focus mainly on the response AVG GAP (%). The best parameter settings on *N*, *M*, *L* and *Tenure*, according to AVG GAP(%), are 64, 500, 10 and 10%, respectively (see Fig. 4). Although *M* and *L* prolong the computation time (see Fig. 6), they are statistically significant and their higher values are necessary. Coosing a value of 64 vs. 32 for *N* negatively affects the

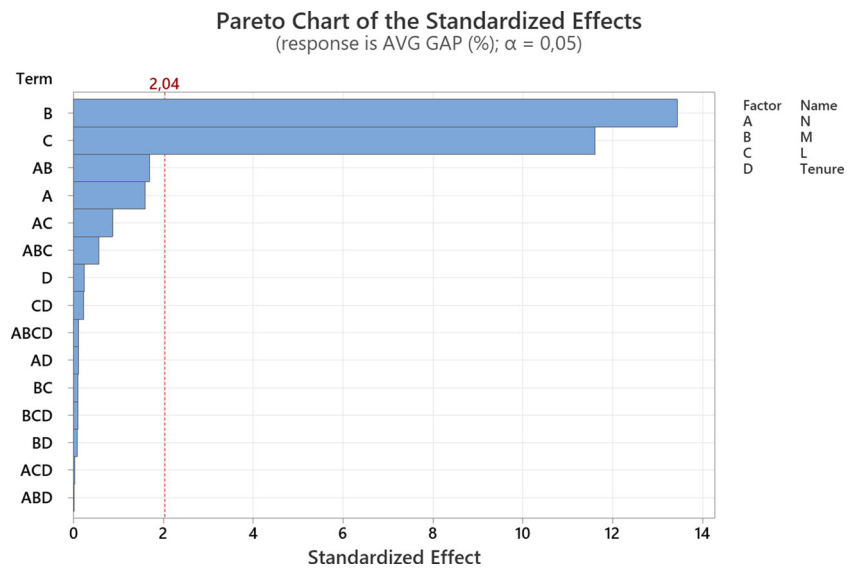
execution time (it increases by more than 40%) and is not statistically significant. Furthermore, the best gap for *N* = 64 is 0.02% and 0.03% for *N* = 32, a minimal difference for the longer computational time required. Finally, *Tenure* does not statistically affect either AVG GAP or AVG TIME. As explained previously, the reason lies in the algorithm’s capacity to adjust the tenure value as the search progresses. So it does not affect starting with a slightly higher or lower value.

The main objective of this paper is not to obtain the best possible solutions in the deterministic environment. So we explore the possibility of considering *N* = 64 and of even analyzing the results of larger values for the most significant factors, such as *M* and *L*. This paper focuses on uncertain conditions using a simulation optimization approach, such as the simheuristics concept. For this reason, having a good deterministic search algorithm (i.e. 0.03% gap) with reasonable computation times is sufficient. In this way, the algorithm can be efficiently combined with Monte Carlo simulations to search for the best solutions under uncertainty. For all these reasons, the values chosen for the adjustment of the

Table 2 Analysis of variance for AVG GAP (%)

Source	Sum of squares	Df	Mean square	F-ratio	P-value
A	0.000138063	1	0.000138063	2.90	0.0987
B	0.0096685	1	0.0096685	202.98	0.0000
C	0.00721629	1	0.00721629	151.50	0.0000
D	0.00000643618	1	0.00000643618	0.14	0.7157
AB	0.000154193	1	0.000154193	3.24	0.0817
AC	0.0000411748	1	0.0000411748	0.86	0.3597
AD	6.2062E-9	1	6.2062E-9	0.00	0.9910
BC	6.47327E-7	1	6.47327E-7	0.01	0.9079
BD	6.47327E-7	1	6.47327E-7	0.04	0.8423
CD	8.92547E-7	1	8.92547E-7	0.02	0.8920
ABC	0.0000178064	1	0.0000178064	0.37	0.5454
ABD	4.03663E-7	1	4.03663E-7	0.01	0.9272
ACD	0.00000109935	1	0.00000109935	0.02	0.8802
BCD	1.43664E-9	1	1.43664E-9	0.00	0.9957
blocks	0.00023501	2	0.000117505	2.47	0.1013
Total error	0.00147664	31	0.0000476335		
Total (corr.)	0.0189591	47			

Fig. 3 Pareto chart for AVG GAP(%)



Main Effects Plot for AVG GAP (%)
Fitted Means

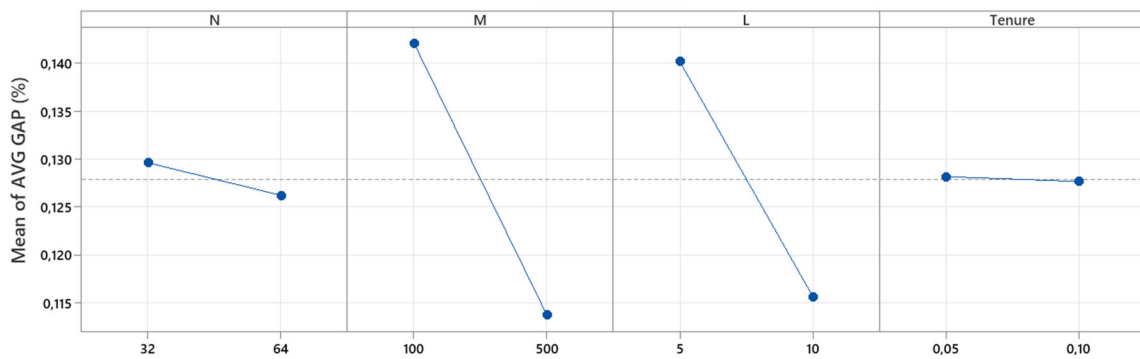
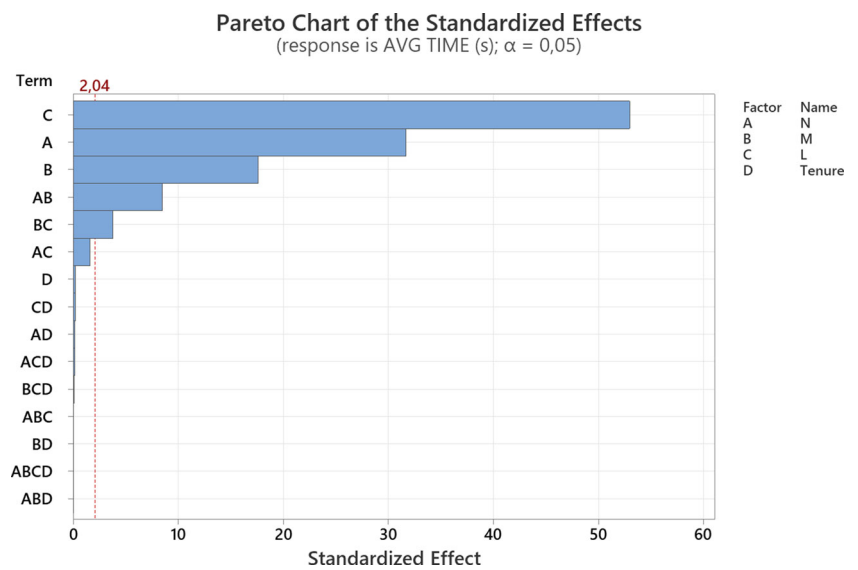


Fig. 4 Main effect plot for AVG GAP(%)

Fig. 5 Pareto chart for AVG TIME(s)



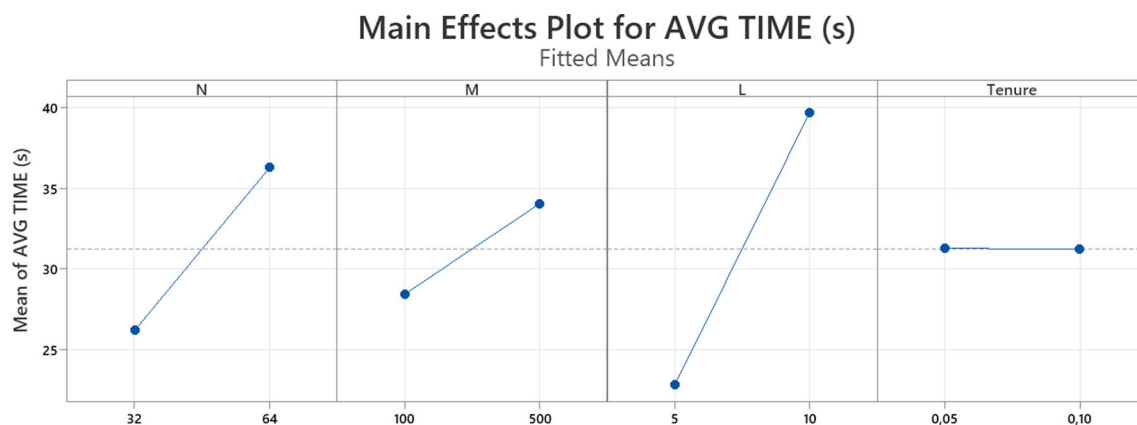


Fig. 6 Main effect plot for AVG TIME(s)

TS+PR algorithm, used later within the simheuristic framework, are $N = 32$, $M = 1000$, $L = 10$ and $Tenure = 10\%$.

Three more parameters are defined by applying the simheuristic approach described in Algorithm 2, which are *longSim*, *shortSim* and *L2*. The first parameter is defined as being large enough to have a full, meaningful simulation of a solution under stochastic uncertainty. The second parameter attempts to quickly evaluate a solution in an uncertain environment to guide the search for better ones. The last parameter consists of a pool of elite solutions (quickly evaluated) to be finally validated in the complete simulation. This pool is ordered from the best to the worst result of the fast simulation. A solution in a lower position may obtain a better result in the long simulation than another in a higher initial position. For this reason, pool size must be large enough to capture the best possible solution in the global simulation. The exploratory studies indicate that the best final solution after the full simulation appears before the first five positions of the elite pool. To ensure that the best solution is obtained in the complete simulation, a value higher than five is finally chosen.

Table 3 summarizes all the parameters and the values selected for the following computational experiments applied to the benchmark instances described in the previous points (Table 4).

5.5 Results and discussion

Table 5 includes the results obtained for four different approaches to solve the UFLP with the MED benchmark instances. The first column displays instances, whose names are combinations of the number of demand points and the opening cost scheme. The next two columns show the solutions obtained using the Gurobi Optimizer solver, along with the corresponding computational time required to reach those solutions. The subsequent three columns present the best-found solutions obtained from our tabu search with the path-relinking approach, along with the computational time needed to obtain them. Each instance is run 30 times using a different seed for the random number generator, and the best and average results are reported. Furthermore, the three columns that follow depict the best-found solutions achieved using the multi-start heuristic proposed by Resende and Werneck [25], along with the computational time required for each solution. The next three columns illustrate the best-found solutions obtained with the ILS-based approach proposed by De Armas et al. [40], along with the computational time taken to achieve those results. Finally, the last three columns display the best-found solutions obtained using the multi-wave algorithm approach by Glover et al. [57], plus the computational time taken to reach these solu-

Table 3 Parameter setting

Parameter	Description	Value
N	Maximum runs of the algorithm with different seeds	32
M	Maximum number of tabu iterations without improvement	500
L	Elite solutions pool size	10
<i>Tenure</i>	Number of iterations that a move is considered tabu	$0.1 * I $
<i>longSim</i>	Number of long MCS runs	100,000
<i>shortSim</i>	Number of short MCS runs	1,000
<i>L2</i>	Elite stochastic solutions pool size	10

Table 4 Performance measures between different computers used in the literature to evaluate MED instances of the UFLP

Literature	PC	PLUS
De Armas et al. [40]	Intel Core i5-7400T 2.4 GHz	1.62x
Glover et al. [57]	Intel Xeon E3-1505M 2.8 GHz	1.32x
Ours	Intel Core i5-9600K 3.7 GHz	1.00x

tions. To deal with the different CPU performance times for each previous method found in the literature, the results can be refactored in time using the performance characteristics of the employed CPUs. We used the tables published on specialized websites; i.e., <http://www.techarp.com> and <https://www.userbenchmark.com>, which evaluate CPU performance, to complete the conversion and normalisation for fair results. Based on the PLUS column, in Table 4 we include the number of times that our computer was faster than each computer used in the literature. It should be noted that it was not possible to normalize the times reported in Resende and Werneck [25] due to lack of comparative information and the difficulty of the conversion process when employing a CPU with a different architecture (RISC). However, it is reasonable to assume that the normalized times of their work would be very competitive.

Table 6 presents the results obtained for the uncertainty levels considered using the MED benchmark instances. Each instance was run 10 times using our simheuristic algorithm with different seeds for the random number generator, and the best results are reported. The first column identifies the instances, while the following three display the results obtained by our approach for the deterministic UFLP. First, the column labeled *BKS* reports the best-known solutions provided by the Gurobi Optimizer solver, along with our best-found deterministic solutions labeled *OBD*. We also calculate the percentage gaps of the best-found deterministic solutions compared to the best-known solutions. The remaining columns showcase the results obtained for three different uncertainty levels. In the second section of the table, we report the results obtained for the low uncertainty level. The *OBD-S* column displays the expected cost obtained when evaluating the best deterministic solution (*OBD*) in a stochastic scenario with a low uncertainty level. To compute the expected cost, an intensive simulation process is applied to the *OBD*. This process aims to assess the quality of our best-found deterministic solutions at varying uncertainty levels. The next column (*OBS*) shows the expected cost obtained using our simheuristic approach for the stochastic version of the problem. This approach considers the random service costs during the solution search. The subsequent section of the table presents the results for the medium uncertainty level. In the *OBD-S* column, we report the expected cost obtained when evaluating the best deterministic solution (*OBD*) in

a stochastic scenario with medium uncertainty. Similarly, the next column (*OBS*) displays the expected cost obtained using our simheuristic algorithm. Finally, the last section of the table exhibits the results for the high uncertainty level. The *OBD-S* column shows the expected cost obtained when evaluating the best deterministic solution (*OBD*) in a stochastic scenario with high uncertainty. The last column (*OBS*) displays the expected cost obtained with our simheuristic approach.

For the sake of completeness, Table 7 covers the additional details of the computational results omitted in Table 6. The first column identifies the instances, while the following four columns display the facilities open cost, the related nonlinear penalty cost, the threshold defined as the instance maximum service cost and the number of open facilities of the best-found solution, respectively. The remaining columns showcase the percentage gaps obtained for three different uncertainty levels, along with the number of open facilities of the best-found stochastic solutions. In the second section of the table, we report the results obtained for the low uncertainty level. The *OBD-S GAP (%)* column displays the percentage gap obtained when comparing the best deterministic solutions simulated in a low uncertainty scenario with the best-known solutions (*BKS*). The next column (*OBS GAP (%)*) shows the percentage gap obtained when comparing our simheuristic approach. The subsequent column (*OPEN*) presents the number of open facilities of the best-found stochastic solution. The subsequent section of the table offers the results for the medium uncertainty level. In the *OBD-S GAP (%)* column, we report the percentage gap obtained when comparing the best deterministic solution at a medium uncertainty level to the best-known solutions (*BKS*). The next column (*OBS (%)*) displays the percentage gap obtained using our simheuristic algorithm. Moreover, the subsequent column (*OPEN*) presents the number of open facilities of the best-found stochastic solution. Finally, the last section of the table exhibits the results for the high uncertainty level. The *OBD-S GAP (%)* column depicts the percentage gap obtained when comparing the best deterministic solution at a high uncertainty level. The next column (*OBS GAP (%)*) displays the percentage gap obtained using our simheuristic algorithm. The last column (*OPEN*) displays the number of open facilities of the best-found stochastic solution.

Figures 7, 8 and 9 depict an overview of Table 6 by showing our algorithm's performance for all the considered uncertainty levels. In these box plots, the horizontal and vertical axes represent the three uncertainty levels and the percentage gap obtained in relation to the *BKS* reported by the Gurobi Optimization solver, respectively. Note for the deterministic version of the UFLP, that our tabu search with the path re-linking approach nearly reaches the *BKS* for the $\sqrt{n}/10$, $\sqrt{n}/100$, and $\sqrt{n}/1000$ opening cost schemes, with a gap of approximately 0.00%, 0.03%, and 0.04%, respec-

Table 5 Comparison of different approaches to solve the UFLP

Instance	Gurobi		TS+PR approach		Resende and Werneck [25]		De Armas et al. [40]		Glover et al. [57]	
	OBD	Time (s)	OBD	Time (s)	OBD	Time (s)	OBD	Time (s)	OBD	Time (s)
500-10	798,577.0	9.23	798,577.0	0.68	798,577.0	33.20	798,577.0	28.86	798,577.0	6.21
500-100	326,790.0	7.31	326,852.0	1.25	326,805.4	32.90	326,894.0	11.41	326,790.0	15.48
500-1000	99,169.0	8.56	99,169.0	2.71	99,169.0	23.60	99,169.0	8.97	99,169.0	35.63
1000-10	1,434,154.0	225.23	1,434,154.0	2.03	1,434,185.4	173.90	1,434,154.0	33.82	1,434,154.0	29.28
1000-100	607,878.0	50.51	607,886.0	3.41	607,880.4	148.80	607,939.0	98.96	607,878.0	101.71
1000-1000	220,560.0	35.18	220,579.0	11.04	220,560.9	141.70	220,959.0	108.16	220,560.0	206.62
1500-10	2,000,801.0	688.74	2,000,801.0	4.41	2,001,121.7	347.80	2,001,920.0	109.98	2,000,801.0	84.04
1500-100	866,454.0	87.48	866,648.0	7.64	866,493.2	378.70	866,769.0	237.24	866,454.0	237.43
1500-1000	334,962.0	75.07	335,113.0	20.52	334,973.2	387.20	337,452.0	235.04	335,002.0	530.93
2000-10	2,558,118.0	652.92	2,558,118.0	9.44	2,558,120.8	717.50	2,558,125.0	138.36	2,558,118.0	209.67
2000-100	1,122,748.0	225.24	1,122,872.0	13.96	1,122,861.9	650.80	1,124,572.0	603.79	1,122,805.0	499.11
2000-1000	437,686.0	200.49	437,833.0	37.95	437,690.7	760.00	442,136.0	614.10	437,693.0	907.61
2500-10	3,099,907.0	30,033.03	3,099,907.0	14.47	3,100,224.7	1,419.50	3,102,719.0	437.05	3,099,907.0	401.29
2500-100	1,347,516.0	138.17	1,347,898.0	18.94	1,347,577.6	1,128.20	1,349,510.0	1,171.89	1,347,516.0	783.48
2500-1000	534,405.0	95.88	534,923.0	58.37	534,426.6	1,309.40	539,485.0	1,194.53	534,506.0	2,068.08
3000-10	3,570,766.0	497.78	3,570,766.0	19.21	3,570,818.8	1,624.10	3,570,930.0	1,604.06	3,570,766.0	662.45
3000-100	1,602,154.0	1,283.93	1,603,821.0	27.38	1,602,530.9	1,977.60	1,606,424.0	1,839.02	1,602,345.0	1,292.78
3000-1000	643,463.0	366.48	644,003.0	75.02	643,541.8	2,081.40	650,777.0	1,884.77	643,797.0	2,561.09
Average:	1,200,339.3	1,926.76		18.25	740.73	0.01	573.34	0.27		0.01

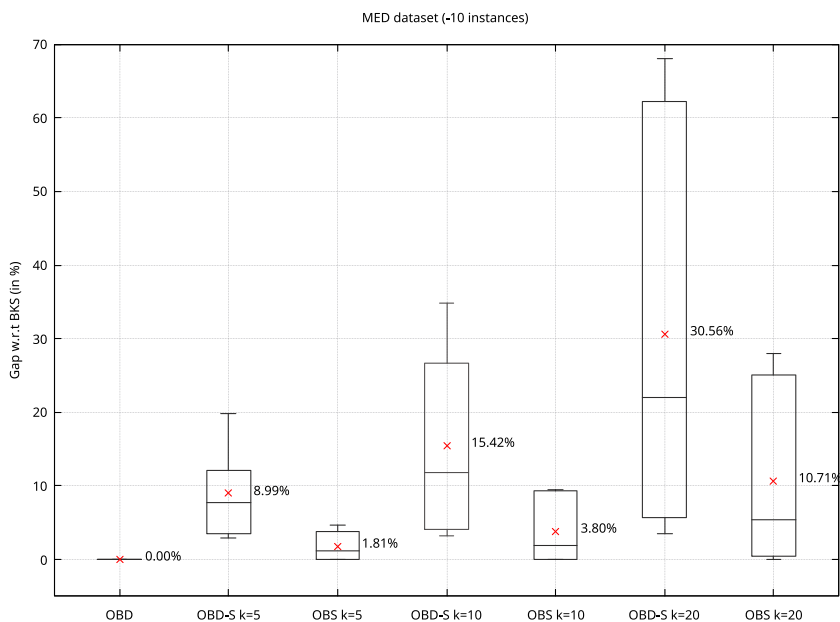
Table 6 Computational results of the different uncertainty levels

Instance	BKS	OBD	GAP (%)	k = 5		k = 10		k = 20	
				OBD-S	OBS	OBD-S	OBS	OBD-S	OBS
500-10	798,577.0	798,577.0	0.00	856,084.4	808,276.8	874,968.9	818,489.1	914,993.1	848,529.5
500-100	326,790.0	326,852.0	0.02	331,215.7	328,815.4	331,889.7	329,509.3	334,291.3	331,812.6
500-1000	99,169.0	99,169.0	0.00	102,761.5	102,346.9	103,311.4	102,892.1	103,699.4	103,160.0
1000-10	1,434,154.0	1,434,154.0	0.00	1,475,848.9	1,434,355.9	1,479,961.3	1,434,311.1	1,485,347.4	1,434,977.9
1000-100	607,878.0	607,886.0	0.00	611,337.3	609,586.0	613,050.2	611,478.9	618,716.5	616,703.2
1000-1000	220,560.0	220,579.0	0.01	225,001.1	224,507.4	227,146.0	226,630.5	230,140.9	229,422.3
1500-10	2,000,801.0	2,000,801.0	0.00	2,070,767.5	2,002,292.9	2,083,208.2	2,002,651.3	2,113,361.2	2,008,947.8
1500-100	866,454.0	866,648.0	0.02	874,809.4	872,746.3	884,531.1	880,075.6	911,279.7	900,923.2
1500-1000	334,962.0	335,113.0	0.05	338,613.9	338,498.6	341,336.3	341,235.3	346,378.0	346,299.9
2000-10	2,558,118.0	2,558,118.0	0.00	3,065,319.6	2,654,437.6	3,450,705.7	2,797,598.0	4,299,087.3	3,273,881.8
2000-100	1,122,748.0	1,122,872.0	0.01	1,126,631.0	1,125,099.6	1,126,673.3	1,125,657.7	1,128,593.1	1,128,267.6
2000-1000	437,686.0	437,833.0	0.03	439,507.6	439,321.4	441,122.3	440,905.9	445,427.7	445,151.5
2500-10	3,099,907.0	3,099,907.0	0.00	3,358,824.3	3,134,747.7	3,539,050.0	3,144,055.0	4,009,604.5	3,241,657.1
2500-100	1,347,516.0	1,347,898.0	0.03	1,361,228.9	1,357,077.0	1,372,497.1	1,362,192.5	1,407,027.5	1,386,424.2
2500-1000	534,405.0	534,923.0	0.10	538,670.6	538,140.7	542,609.6	542,050.9	551,992.1	551,284.0
3000-10	3,570,766.0	3,570,766.0	0.00	4,004,050.6	3,736,788.7	4,520,871.5	3,907,568.4	5,791,495.3	4,463,877.3
3000-100	1,602,154.0	1,603,821.0	0.10	1,608,936.2	1,605,330.7	1,610,552.4	1,606,826.6	1,621,815.9	1,618,208.9
3000-1000	643,463.0	644,003.0	0.08	644,947.5	644,914.0	646,667.0	646,574.6	653,135.5	652,956.0
Average:	1,200,339.3	1,200,551.1	0.03	1,279,697.6	1,219,849.1	1,343,897.3	1,240,039.0	1,498,132.6	1,310,138.0

Table 7 Additional details of the computational results

Instance	OPEN COST	PENALTY	THRESHOLD	OPEN	k = 5			k = 10			k = 20		
					OBD-S GAP (%)	OBS GAP (%)	OPEN	OBD-S GAP (%)	OBS GAP (%)	OPEN	OBD-S GAP (%)	OBS GAP (%)	OPEN
500-10	22,361.0	44,722.0	2,229.0	13	7.20	1.21	15	9.57	2.49	15	14.58	6.26	15
500-100	2,236.0	4,472.0	1,123.0	59	1.35	0.62	64	1.56	0.83	64	2.30	1.54	64
500-1000	224.0	448.0	223.0	334	3.62	3.20	344	4.18	3.75	340	4.57	4.02	348
1000-10	31,623.0	63,246.0	2,452.0	15	2.91	0.01	16	3.19	0.01	16	3.57	0.06	16
1000-100	3,162.0	6,324.0	1,051.0	78	0.57	0.28	78	0.85	0.59	78	1.78	1.45	78
1000-1000	316.0	632.0	316.0	382	2.01	1.79	392	2.99	2.75	391	4.34	4.02	391
1500-10	38,730.0	77,460.0	2,205.0	18	3.50	0.07	19	4.12	0.09	19	5.63	0.41	19
1500-100	3,873.0	7,746.0	934.0	84	0.96	0.73	90	2.09	1.57	90	5.17	3.98	90
1500-1000	387.0	774.0	364.0	414	1.09	1.06	416	1.90	1.87	416	3.41	3.38	423
2000-10	44,721.0	89,442.0	1,600.0	21	19.83	3.77	24	34.89	9.36	23	68.06	27.98	24
2000-100	4,472.0	8,944.0	1,123.0	93	0.35	0.21	94	0.35	0.26	94	0.52	0.49	94
2000-1000	447.0	894.0	412.0	455	0.42	0.37	459	0.79	0.74	459	1.77	1.71	459
2500-10	50,000.0	100,000.0	1,750.0	22	8.35	1.12	26	14.17	1.42	26	29.35	4.57	26
2500-100	5,000.0	10,000.0	919.0	101	1.02	0.71	106	1.85	1.09	106	4.42	2.89	106
2500-1000	500.0	1,000.0	380.0	491	0.80	0.70	494	1.54	1.43	493	3.29	3.16	494
3000-10	54,772.0	109,544.0	1,562.0	23	12.13	4.65	26	26.61	9.43	26	62.19	25.01	26
3000-100	5,477.0	10,954.0	992.0	111	0.42	0.20	111	0.52	0.29	111	1.23	1.00	111
3000-1000	548.0	1,096.0	427.0	517	0.23	0.23	520	0.50	0.48	520	1.50	1.48	520
Avg					3.71	1.16		6.20	2.14		12.09	5.19	

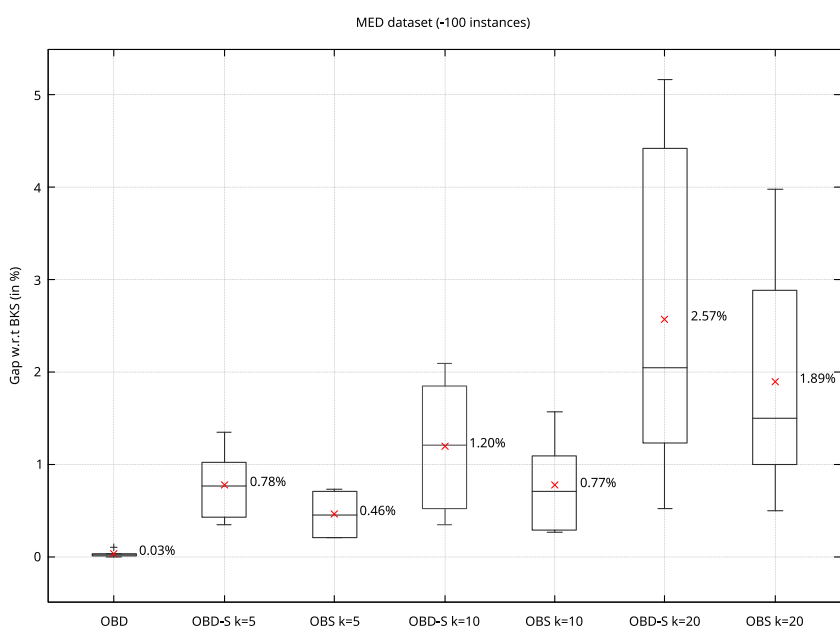
Fig. 7 Gaps of $\sqrt{n}/10$ opening cost scheme instances w.r.t the BKS



tively. These results highlight the quality of our algorithm because our approach provides highly competitive solutions. Regarding the stochastic version of the UFLP, which is the main contribution of this paper, the obtained results show that the solutions provided by our heuristic approach for the three different uncertainty levels clearly outperform the solutions for the deterministic UFLP when they are simulated at the corresponding uncertainty level. In other words, our best-found solutions for the deterministic version of the problem (*OBD*) might be suboptimal when uncertainty is considered. Hence the importance of integrating simulation methods when dealing with optimization problems with

uncertainty. Note also that the *OBD* can be seen as a reference lower bound in an ideal scenario with perfect information (i.e., without uncertainty) for the expected cost under uncertainty conditions. Similarly, *OBD-S* can be seen as an upper bound for the expected cost at the different uncertainty levels. As expected, the gaps for all three opening cost schemes worsen as the uncertainty level of k increases. For the specific case of the $\sqrt{n}/10$ opening cost scheme, gaps increase more than for others. As shown in Table 7, this is because the penalty cost (opening cost) is higher in this case. The penalty amounts to 3-5% of the total costs, but is less than 1.5% for the $\sqrt{n}/100$ opening cost scheme and less than 0.5% for the

Fig. 8 Gaps of $\sqrt{n}/100$ opening cost scheme instances w.r.t the BKS



remaining one. In addition, the number of open facilities is much smaller. These are cases in which a few facilities with high capacity serve many customers. The average number of open facilities for this opening cost scheme is around 18 in the best-found deterministic solutions, but rises to over 430 for the instances with suffix 1000. In this high opening cost scheme, where there are few viable facilities to open and with a high cost penalty, the *OBD-S* result is that which obtains the worst values on average, with a gap of 8.99%, 15.42% and 30.56% for all three uncertainty levels. In this environment, our simheuristic approach is able to achieve a 3 to 5 times better improvement (1.81%, 3.80%, and 10.71%). For the remaining cases (suffixes 100 and 1000) where the penalty costs are lower and the number of open installations is bigger, and the gaps shown in Table 7 and Figs. 7, 8 and 9 are smaller. It is not surprising that the *OBD* solutions better perform under uncertainty when the penalty cost is lower. Our *OBS* solutions improve the deterministic model's performance for all the different k uncertainty levels, and also for all the opening cost schemes, which demonstrates the validity of the proposed approach.

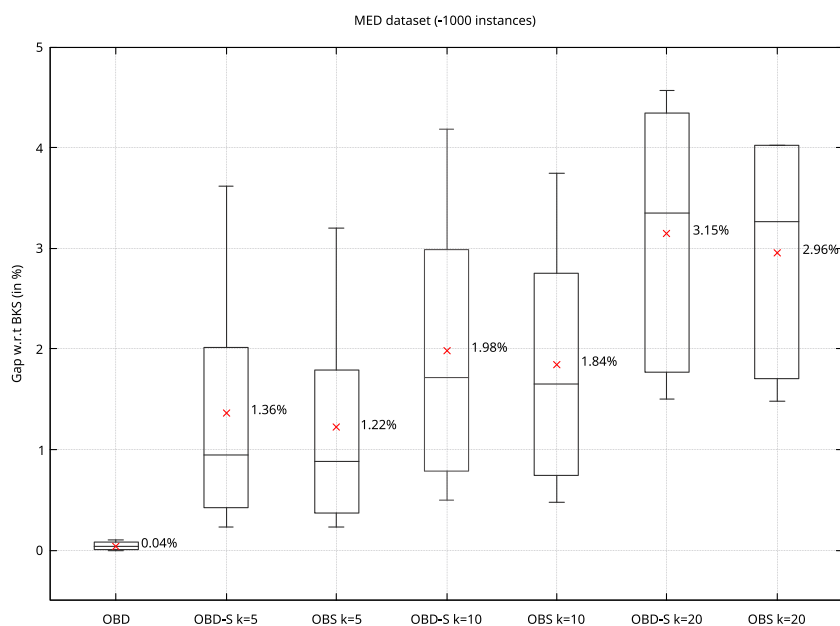
When taking everything into account, the findings validate the significance of factoring in uncertainty while searching for a solution because it can greatly affect the quality of our best-found solutions. For instance, when our best deterministic solution is simulated in a stochastic scenario, it yields a much higher expected cost. Conversely, our best-found stochastic solution (*OBS*) provides a lower expected cost. Therefore, incorporating uncertainty aspects during the search process produces better results than when simulating uncertainty elements after finding a solution for the deterministic version of the problem.

6 Conclusions

In this paper, we analyze the UFLP under uncertainty, which occurs in several real-life systems. This uncertainty arises when inputs, such as customer demands or service costs, are random variables instead of deterministic values. To solve this optimization problem, we propose a novel solving methodology that combines a tabu search metaheuristic with path-relinking. The proposed algorithm is tested on the largest instances from the literature because they are the most challenging ones. The results show that our approach is capable of obtaining near-optimal solutions in short computational times. Additionally, the results are compared to the *BKS* from the literature, along with other competitive approaches, to validate our algorithm's effectiveness in the deterministic scenario. Then the algorithm is converted into a simheuristic by integrating it with a simulation component to solve the UFLP with random service costs. The log-normal probability distribution is employed to model the random service costs. The obtained results show our simheuristic approach outperforms the simulated deterministic solutions when uncertainty is considered. Thus our simheuristic approach constitutes a general methodology that can be employed to efficiently solve several FLP variants under uncertainty.

One of the limitations of the present work is that it considers only uncertainty of a stochastic nature. Thus extending the simheuristic algorithm to address the UFLP in a more general setting, including stochastic and nonstochastic uncertainty elements like *type-2* fuzzy systems [58] and rough sets [59], is a future research line. Another limitation is that simulations are performed on a single CPU core. As a future

Fig. 9 Gaps of $\sqrt{n}/1000$ opening cost scheme instances w.r.t the BKS



research line, it would be interesting to consider a parallel version of the simheuristic algorithm to obtain optimal solutions in an even shorter time, which may allow fast decision making in time-sensitive scenarios. Finally, another possible extension for this paper is to develop a similar combination of a tabu search metaheuristic with a path-relinking simheuristic to obtain near-optimal solutions in short computational times for other optimization problems under uncertainty, for example, vehicle routing problems, arc routing problems and scheduling problems.

Acknowledgements This work has been partially supported by the Spanish Ministry of Science (PDC2022-133957-I00, PID2022-138860 NB-I00 and RED2022-134703-T) and by the Generalitat Valenciana (PROMETEO/2021/065).

Author Contributions Conceptualization: Angel A. Juan, David Peidro; Methodology: Angel A. Juan, Xabier A. Martin, Javier Panadero; Software: David Peidro; Validation: Angel A. Juan, Xabier A. Martin, David Peidro, Javier Panadero; Software: David Peidro; Writing-original draft preparation: Xabier A. Martin, David Peidro, Javier Panadero; Writing-review and editing: Angel A. Juan, David Peidro. All authors have read and agreed to the published version of the manuscript.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Data availability statement Data sharing is not applicable to this article because no datasets were generated or analyzed during the present study.

Declarations

Ethics approval This article does not contain any studies with human participants or animals performed by any of the authors

Conflict of interest The authors declare no conflict of interest

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Stollsteimer JF (1961) The Effect of Technical Change and Output Expansion on the Optimum Number, Size, and Location of Pear Marketing Facilities in a California Pear Producing Region. University of California, Berkeley
2. Kuehn AA, Hamburger MJ (1963) A heuristic program for locating warehouses. *Manage Sci* 9(4):643–666
3. Manne AS (1964) Plant location under economies-of-scale-decentralization and computation. *Manage Sci* 11(2):213–235
4. Balinski ML (1966) On finding integer solutions to linear programs. In: *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems*, pp. 225–248. Yortown Heights, NY
5. Verter V (2011) Uncapacitated and capacitated facility location problems. *Foundations of Location Analysis*. 155:25–37
6. Cornuejol G, Nemhauser GL, Wolsey LA (1990) The uncapacitated facility location problem. In: Mirchandani, P.B., Francis, R.L. (eds.) *Discrete Location Theory*. Wiley Series in Discrete Mathematics and Optimization, pp. 119–171. John Wiley & Sons, New York
7. Silva FJF, Figuera DS (2007) A capacitated facility location problem with constrained backlogging probabilities. *Int J Prod Res* 45(21):5117–5134
8. Klose A, Drexl A (2005) Facility location models for distribution system design. *Eur J Oper Res* 162(1):4–29
9. Yang Z, Chu F, Chen H (2012) A cut-and-solve based algorithm for the single-source capacitated facility location problem. *Eur J Oper Res* 221(3):521–532
10. Correia I, Saldanha-da-Gama F (2019) Facility location under uncertainty. *Location Science*, 185–213
11. Figueira G, Almada-Lobo B (2014) Hybrid simulation-optimization methods: A taxonomy and discussion. *Simul Model Pract Theory* 46:118–134
12. Amaran S, Sahinidis NV, Sharda B, Bury SJ (2016) Simulation optimization: a review of algorithms and applications. *Ann Oper Res* 240(1):351–380
13. Castaneda J, Martin XA, Ammouriova M, Panadero J, Juan AA (2022) A fuzzy simheuristic for the permutation flow shop problem under stochastic and fuzzy uncertainty. *Mathematics* 10(10)
14. Hatami S, Calvet L, Fernandez-Viagas V, Framinan JM, Juan AA (2018) A simheuristic algorithm to set up starting times in the stochastic parallel flowshop problem. *Simul Model Pract Theory* 86:55–71
15. Glover F (1986) Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*. 13(5):533–549
16. Al-Sultan KS, Al-Fawzan MA (1999) A tabu search approach to the uncapacitated facility location problem. *Ann Oper Res* 86:91–103
17. Sun M (2006) Solving the uncapacitated facility location problem using tabu search. *Computers & Operations Research*. 33(9):2563–2589
18. Erlenkotter D (1978) A dual-based procedure for uncapacitated facility location. *Oper Res* 26(6):992–1009
19. Efraymson M, Ray T (1966) A branch-bound algorithm for plant location. *Oper Res* 14(3):361–368
20. Spielberg K (1969) Algorithms for the simple plant-location problem with some side conditions. *Oper Res* 17(1):85–111
21. Körkel M (1989) On the exact solution of large-scale simple plant location problems. *Eur J Oper Res* 39(2):157–173
22. Hochbaum DS (1982) Approximation algorithms for the set covering and vertex cover problems. *SIAM J Comput* 11(3):555–556
23. Ghosh D (2003) Neighborhood search heuristics for the uncapacitated facility location problem. *Eur J Oper Res* 150(1):150–162
24. Michel L, Van Hentenryck P (2004) A simple tabu search for warehouse location. *Eur J Oper Res* 157(3):576–591
25. Resende MG, Werneck RF (2006) A hybrid multistart heuristic for the uncapacitated facility location problem. *Eur J Oper Res* 174(1):54–68
26. C Martins L Hirsch P, Juan AA (2021) Agile optimization of a two-echelon vehicle routing problem with pickup and delivery. *Int Trans Oper Res* 28(1):201–221

27. Martins LDC, Tarchi D, Juan AA, Fusco A (2022) Agile optimization for a real-time facility location problem in internet of vehicles networks. *Networks* 79(4):501–514
28. Holmberg K, Rönnqvist M, Yuan D (1999) An exact algorithm for the capacitated facility location problems with single sourcing. *Eur J Oper Res* 113(3):544–559
29. Díaz JA, Fernández E (2002) A branch-and-price algorithm for the single source capacitated plant location problem. *Journal of the Operational Research Society*. 53(7):728–740
30. Chen C-H, Ting C-J (2008) Combining Lagrangian heuristic and ant colony system to solve the single source capacitated facility location problem. *Transportation Research Part E: Logistics and Transportation Review*. 44(6):1099–1122
31. Ahuja RK, Orlin JB, Pallottino S, Scaparra MP, Scutellà MG (2004) A multi-exchange heuristic for the single-source capacitated facility location problem. *Manage Sci* 50(6):749–760
32. Filho VJMF, Galvão RD (1998) A tabu search heuristic for the concentrator location problem. *Locat Sci* 6(1):189–209
33. Delmaire H, Díaz JA, Fernández E, Ortega M (1999) Reactive GRASP and tabu search based heuristics for the single source capacitated plant location problem. *INFOR: Information Systems and Operational Research* 37(3):194–225
34. Estrada-Moreno A, Ferrer A, Juan AA, Bagirov A, Panadero J (2020) A biased-randomised algorithm for the capacitated facility location problem with soft constraints. *J Oper Res Soc* 71(11):1799–1815
35. Fotakis D (2011) Online and incremental algorithms for facility location. *ACM SIGACT News* 42(1):97–131
36. Eiselt HA, Marianov V (eds) (2011) *Foundations of Location Analysis*. International Series in Operations Research & Management Science. Springer, Boston, MA
37. Snyder LV, Daskin MS (2006) Stochastic p-robust location problems. *IIE Trans* 38(11):971–985
38. Balachandran V, Jain S (1976) Optimal facility location under random demand with general cost structure. *Naval Research Logistics Quarterly*. 23(3):421–436
39. Reyes-Rubiano L, Ferone D, Juan AA, Faulin J (2019) A simheuristic for routing electric vehicles with limited driving ranges and stochastic travel times. *SORT-Statistics and Operations Research Transactions*. 43(1):0003–0024
40. De Armas J, Juan AA, Marquès JM, Pedroso JP (2017) Solving the deterministic and stochastic uncapacitated facility location problem: From a heuristic to a simheuristic. *Journal of the Operational Research Society*. 68(10):1161–1176
41. Quintero-Araujo CL, Guimarães D, Juan AA (2021) A simheuristic algorithm for the capacitated location routing problem with stochastic demands. *Journal of Simulation*. 15(3):217–234
42. Bayliss C, Panadero J (2023) Simheuristic and learnheuristic algorithms for the temporary-facility location and queuing problem during population treatment or testing events. *J Simul* 0(0):1–20
43. Juan AA, Keenan P, Martí R, McGarraghy S, Panadero J, Carroll P, Oliva D (2023) A review of the role of heuristics in stochastic optimisation: From metaheuristics to learnheuristics. *Ann Oper Res* 320(2):831–861
44. Marques MDC, Dias JM (2018) Dynamic location problem under uncertainty with a regret-based measure of robustness. *Int Trans Oper Res* 25(4):1361–1381
45. Zhang J, Li M, Wang Y, Wu C, Xu D (2019) Approximation algorithm for squared metric two-stage stochastic facility location problem. *J Comb Optim* 38:618–634
46. Ramshani M, Ostrowski J, Zhang K, Li X (2019) Two level uncapacitated facility location problem with disruptions. *Computers & Industrial Engineering*. 137:106089
47. Koca E, Noyan N, Yaman H (2021) Two-stage facility location problems with restricted recourse. *IIE Transactions*. 53(12):1369–1381
48. Gruler A, Quintero-Araújo CL, Calvet L, Juan AA (2017) Waste collection under uncertainty: A simheuristic based on variable neighbourhood search. *European Journal of Industrial Engineering*. 11(2):228–255
49. Fu MC (2003) Feature article: Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing* 14(3):192–215
50. Ho SC, Gendreau M (2006) Path relinking for the vehicle routing problem. *Journal of Heuristics*. 12:55–72
51. Peng B, Lü Z, Cheng TCE (2015) A tabu search/path relinking algorithm to solve the job shop scheduling problem. *Computers & Operations Research*. 53:154–164
52. Chica M, Juan AA, Bayliss C, Cordón O, Kelton WD (2020) Why simheuristics? benefits, limitations, and best practices when combining metaheuristics with simulation. *SORT-Statistics and Operations Research Transactions*. 44(2):311–334
53. Ahn S, Cooper C, Cornuejols G, Frieze A (1988) Probabilistic analysis of a relaxation for the k-median problem. *Math Oper Res* 13(1):1–31
54. Barahona F, Chudak FA (2000) In: Pardalos, P.M. (ed.) *Solving Large Scale Uncapacitated Facility Location Problems*, pp. 48–62. Springer, Boston, MA
55. Kim JS, Yum B-J (2008) Selection between weibull and lognormal distributions: A comparative simulation study. *Computational Statistics & Data Analysis*. 53(2):477–485
56. Montgomery DC (2017) *Design and Analysis of Experiments*. John Wiley & Sons, Hoboken, New Jersey
57. Glover F, Hanafi S, Guemri O, Crevits I (2018) A simple multi-wave algorithm for the uncapacitated facility location problem. *Frontiers of Engineering Management*. 5(4):451–465
58. Castillo O, Melin P, Kacprzyk J, Pedrycz W (2007) Type-2 fuzzy logic: theory and applications. In: 2007 IEEE International Conference on Granular Computing (GRC 2007), pp. 145–145. IEEE
59. Pawlak Z (1982) Rough sets. *International Journal of Computer & Information Sciences*. 11(5):341–356

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.