Journal of the Operational Research Society (2017) 68, 1161–1176 © 2016 The Operational Research Society. All rights reserved. 0160-5682/17

www.palgrave.com/journals

Solving the deterministic and stochastic uncapacitated facility location problem: from a heuristic to a simheuristic

Jesica de Armas[1]*, Angel A. Juan[1], Joan M. Marquès[1] and João Pedro Pedroso[2]

[1]IN3 - Computer Science Department, Open University of Catalonia, Barcelona, Spain; and [2] Computer Science Department, University of Porto, Porto, Portugal

The uncapacitated facility location problem (UFLP) is a popular combinatorial optimization problem with practical applications in different areas, from logistics to telecommunication networks. While most of the existing work in the literature focuses on minimizing total cost for the deterministic version of the problem, some degree of uncertainty (*e.g.*, in the customers' demands or in the service costs) should be expected in real-life applications. Accordingly, this paper proposes a simheuristic algorithm for solving the stochastic UFLP (SUFLP), where optimization goals other than the minimum expected cost can be considered. The development of this simheuristic is structured in three stages: *(i)* first, an extremely fast savings-based heuristic is introduced; *(ii)* next, the heuristic is integrated into a metaheuristic framework, and the resulting algorithm is tested against the optimal values for the UFLP; and *(iii)* finally, the algorithm is extended by integrating it with simulation techniques, and the resulting simheuristic is employed to solve the SUFLP. Some numerical experiments contribute to illustrate the potential uses of each of these solving methods, depending on the version of the problem (deterministic or stochastic) as well as on whether or not a real-time solution is required.

Journal of the Operational Research Society (2017) **68(10),** 1161–1176. doi:10.1057/s41274-016-0155-6;
published online 14 December 2016

Keywords: uncapacitated facility location problem; stochastic combinatorial optimization problems; metaheuristics; simheuristics

1. Introduction

The facility location problem (FLP), originally introduced by Stollsteimer (1961) and Balinski (1966), involves deciding the position of an undetermined number of facilities (each associated with a decision variable) to minimize the sum of: *(i)* the setup cost of these facilities and *(ii)* the cost related to serving the customers from them. Most versions of the problem assume that the alternative sites where the facilities can be located are predetermined, and also that all inputs (*e.g.*, the demand associated with each customer and the service costs) are deterministic in the sense that they are known in advance. Usually, decisions on facility location are difficult to reverse due to the fixed costs associated with opening a facility. In this regard, the FLP is useful to model allocation problems 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.). A simple example of a FLP instance is shown in Figure 1, where each customer (circle) is assigned via an active connection to its closest open facility (dark square).

The uncapacitated version of the FLP (UFLP) assumes that the capacity of each facility is virtually unlimited or, at least, far beyond the expected demand. According to Verter (2011), the UFLP variant is considered to be the simplest version of the FLP. Nevertheless, even without the capacity constraint, the FLP has been proved to be NP-hard (Cornuejols *et al*, 1990).

More formally, the UFLP is defined over an undirected graph $G = (F, C, E)$, where F is a non-empty subset of facilities (each of them with unlimited capacity), C is a non-empty set of customers to be served, and E is a set of edges connecting each customer $j \in C$ with some of the facilities in F (Figure 1). Delivering a customer $j \in C$ throughout a facility $i \in F$ has a service cost $c_{ij} \geq 0$. Also, each facility $i \in F$ has a fixed opening cost $f_i \geq 0$. Let X be the decision variable denoting the set of open facilities, with $\emptyset \subset X \subseteq F$. Let $\sigma : C \rightarrow F$ be a function assigning to each customer $j \in C$ a facility $\sigma(j) \in F$ satisfying $c_{\sigma(j),j} = \min_{i \in X}\{c_{ij}\}$. Under these circumstances, the UFLP consists in minimizing the total cost of providing service to all customers, *i.e.*, minimize $\sum_{i \in X} f_i + \sum_{j \in C} c_{\sigma(j),j}$.

In the literature, the UFLP is also called the simple facility location problem, the simple warehouse location problem, or the simple plant location problem. Realistic applications of the

**Correspondence: Jesica de Armas, IN3 - Computer Science Department, Open University of Catalonia, Barcelona, Spain.*
E-mail: jde_armasa@uoc.edu

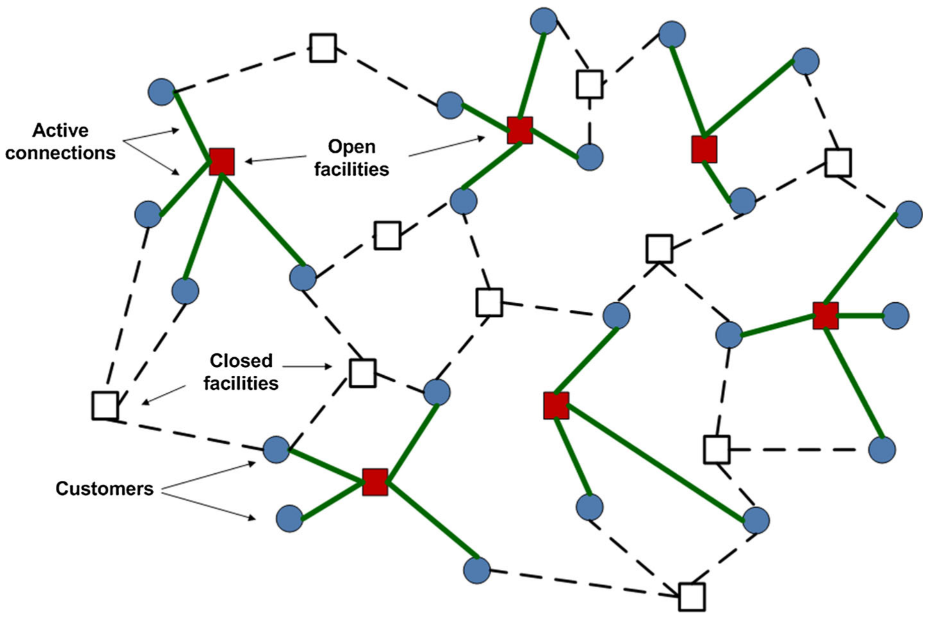

Figure 1 Illustrative example of the facility location problem.

UFLP can be found, for instance, in the lube oil industry (Brahimi and Khan, 2014), in the bank account location (Cornuejols *et al*, 1977), in the self-configuration of wireless sensor networks (Frank and Römer, 2007), in computer vision (Lazic *et al*, 2009), or in health care (Maric *et al*, 2015). Having a wide range of applications, the UFLP has been studied for several decades now (Mallette and Francis, 1972; Hansen, 1976). However, to the best of our knowledge, methods that provide robust solutions taking into account uncertainty conditions are not reported in the literature. This stochastic environment appears, for example, when inputs such as the customers' demands or the service costs are random variables instead of constant values.

To partially close this gap, we introduce here a novel simheuristic algorithm (Juan *et al*, 2015) designed to deal with the stochastic version of the UFLP (SUFLP). First, the optimal values for all the classical UFLP benchmark instances are obtained using the Gurobi commercial solver. This is achieved after implementing the mixed integer programming (MIP) model of the UFLP into a Python script. As far as we know, this is the first time that these optimal values are reported. Then, since obtaining the optimal solution for a large-sized instance might take several hours in a standard computer, we propose a savings-based heuristic for the deterministic UFLP. This heuristic is able to generate reasonably good solutions in milliseconds. Next, we integrate this constructive heuristic inside an iterated local search (ILS) metaheuristic framework. The resulting algorithm allows generating near-optimal solutions for the UFLP in short time. Finally, once the quality of this approach has been tested in the deterministic version of the UFLP, we extend it to a simheuristic algorithm in order to deal with the SUFLP, when optimization goals other than the minimum expected cost should be considered. In effect, notice that in a stochastic environment one could be interested in solutions that offer a good trade-off between total expected cost and risk or variability.

In this regard, our simheuristic approach is able to generate several alternative solutions, each of them offering different values for each of the parameters considered. In particular, for any given instance of the SUFLP we focus on finding: *(i)* the solution with minimum expected cost; *(ii)* the solution which minimizes the third quartile of the cost values obtained when it is run multiple times; and *(iii)* solutions offering a good trade-off between expected cost and standard deviation of the cost values obtained when they are run multiple times (*i.e.*, among those solutions with low expected costs, we are interested in identifying the ones showing a relatively low variability or risk).

Accordingly, the remainder of this paper is structured as follows: Section 2 reviews the literature on the UFLP, enumerating the solution approaches available to this problem. Then, Section 3 reviews the literature on the SUFLP, detailing previous works that have dealt with this particular problem. Section 4 explains our implementation of the Erlenkotter (1978) model for the UFLP and discusses the need for heuristic and metaheuristic approaches to shorten computing times in some real-life applications. Section 5 describes our savings-based heuristic for the UFLP. Section 6 explains how this heuristic can be integrated into a metaheuristic framework to generate an efficient algorithm. Section 7 discusses the quality of the proposed approaches for solving the UFLP. Section 8 describes how the metaheuristic algorithm can be extended into a simheuristic one for the SUFLP, and analyzes some numerical examples of application. Finally, Section 9 highlights the main conclusions of this paper and points out some open research lines.

2. Literature review on the UFLP

This section presents a review of the different approaches to the UFLP. For a more extensive literature review on the FLP, the reader is addressed to Drezner (1995), Snyder (2006), and Fotakis (2011). The FLP was introduced by Stollsteimer (1961) and Balinski (1966), originally referred to as the plant location problem. In general, the FLP has been studied from the perspectives of worst-case analysis, probabilistic analysis, polyhedral combinatory and empirical heuristics (Cornuejols *et al*, 1990). In the existing literature, we can also find exact algorithms for the problem, but its NP-hard nature makes heuristics a more suitable tool to quickly obtain solutions for larger and realistic instances.

One of the first works on the FLP was a branch-and-bound algorithm developed by Efroymson and Ray (1966). They used a compact formulation of the FLP to take advantage of the fact that its linear programming relaxation can be solved by inspection. However, this linear programming relaxation is known to be weak and therefore does not provide tight lower bounds. Another of the early approaches proposed for the problem is the direct search or implicit enumeration method proposed by Spielberg (1969). The author defined two different algorithms based on the same directed search, one considering the facilities initially open and another one considering the facilities initially closed. Later, Schrage (1975) presented a tight linear programming formulation for the FLP different from the one defined by Efroymson and Ray (1966). Schrage applied to this formulation a specialized linear programming algorithm for variable upper bound constraints. Beginning with this tight linear programming formulation, Erlenkotter (1978) presented a dual-based procedure, differing from previous approaches by considering a dual objective function. An improved version of this original algorithm was presented by Körkel (1989). The main drawback of exact approaches is the difficulties solving large real instances in short times.

Regarding the use of approximation algorithms for this problem, one of the earliest was the greedy algorithm proposed by Hochbaum (1982). In the 1990s, the first constant factor approximation emerged by the hand of Shmoys *et al* (1997), and it was later improved by Chudak (1998), being both of these algorithms based on LP-rounding and therefore having high running times. In order to reduce these running times, Jain and Vazirani (1999) proposed a primal-dual algorithm adapted for solving several related problems. This same algorithm was later enhanced in Jain *et al* (2003), obtaining better results. More recently, Li (2013) outperformed the former results by reducing the approximation ratio of the previous approximation algorithms. Approximation algorithms are very valuable for a theoretical analysis of the problem. However, these algorithms are outperformed in practice by more straightforward heuristics with no performance guarantees.

Constructive algorithms and local search methods have been used for decades, starting with Kuehn and Hamburger (1963). The authors presented one of the earliest models for the problem and a heuristic procedure for solving it. Their heuristic comprised two main phases: first a constructive phase is considered as the main program, followed by a second, improvement phase which evaluates the profit implications of dropping individual warehouses or of shifting them from one location to another. Following this work, more sophisticated heuristics have been applied to the FLP. Alves and Almeida (1992) proposed a simulated annealing algorithm. Kratica *et al* (2001) presented a genetic algorithm outperforming previous works. Ghosh (2003) presented a neighborhood search heuristic for the problem, using tabu search as local search and obtaining competitive solutions in very low computational times, compared to exact algorithms. Michel and Van Hentenryck (2004) defined a simple tabu search algorithm, which demonstrated to be robust, efficient, and competitive when compared with previous genetic algorithms applied to the problem. The tabu search algorithm used a linear neighborhood, which flipped a single facility at each iteration. Resende and Werneck (2006) proposed an algorithm based on the GRASP metaheuristic. This algorithm combined a greedy construction phase with a local search procedure and path relinking. It obtained results very close to the lower bound values for a wide range of different instance sets. More recently, Lai *et al* (2010) presented a hybrid algorithm based on the Benders' decomposition algorithm (Benders, 1962) and using a genetic algorithm instead of the costly branch-and-bound method, to obtain good-quality solutions. According to the computational results reported, the algorithm seems to be efficient. However, the author only compared its performance with the Benders' original algorithm.

Concerning the use of parallel computing techniques, Wang *et al* (2008) presented an adaptive version of a parallel multi-population particle swarm optimization algorithm. The implementation obtained an important improvement in terms of execution times, while obtaining competitive results using a standard computer. However, they only solved small- and medium-sized instances of the problem. To the best of our knowledge, there are not approaches in the literature that optimally solve large instances (between 500 and 3000 facilities) under a real-time setting, *i.e.*, a few seconds using a standard computer.

Regardless of the solution method, many variants of the basic FLP have been studied in the literature. An important source of variations is due to considering uncertainty in the problem (Snyder, 2006). This is typically taken into consideration by introducing a wide variance on any of the parameters of the problem (cost, demands, or distances). The goal in these problems is to find a solution that performs well under any possible realization of the random variables, *i.e.*, a robust solution. The random variables can be either continuous or discrete. As an example, Balachandran and Jain (1976) presented a FLP model with piecewise linear production costs that need be neither concave nor convex. Demands are random and continuous, described by some joint probability distributions. In this kind of problems, only the first-stage decisions are available, so there are no recourse decisions. Thus, once the locations are set, they cannot be changed after the uncertainty is resolved. The objectives therefore include the expected recourse costs. In this paper, we consider the UFLP with random customers' demands, which affect the assignment costs of customers to facilities. Therefore, it can be considered to involve stochastic costs. An analysis of this uncertainty is provided as explained in the next sections.

In reference to applications of the FLP in real-time environments, some examples can be found in the literature. These examples are usually related to environments with high mobility, dynamism, or uncertainty. For instance, Gendreau *et al* (2001) described a system for the real-time re-allocation of ambulances to maintain cover after an ambulance responds to a call. Similarly, Kolesar and Walker (1974) considered the redeployment of fire companies in New York City while some of them are responding to a call. An example in digital network design problems is the equipment allocation for video on demand (VoD) network deployments (Thouin and Coates, 2008). VoD services are complex and resource demanding, so deployments involve careful design of many mechanisms where content attributes and usage should be taken into account. The high bandwidth requirements motivate distributed architectures with replication of content. An important and complicated task during the network-planning phase of these distributed architectures is resource allocation. An example of such a distributed infrastructure is the Guifi Net network (www.guifi.net), an open, free, and neutral telecommunication network whose infrastructure is completely supported by its users. In fact, the growth of peer-to-peer networks and the use of mobile devices for accessing the contents have made the problem even more complex.

Another example can be found in Lee and Murray (2010). These authors introduce an approach for survivable network design of citywide wireless broadband based on the FLP model. They address two issues: how to locate the wi-fi equipment to maximally cover the given demand and how to connect wi-fi equipment to ensure survivable networking on a real case scenario in the city of Dublin (Ohio, USA). An online FLP (Meyerson, 2001) is encountered when modeling a network design problem in which several servers need to be purchased and each client has to be connected to one of the servers. Once the network has been constructed, it may be necessary to add additional clients to the network. In this case, additional costs will be incurred; *e.g.*, the cost of connecting a new customer to the cluster, or the cost of installing additional servers whenever the current capacity cannot accommodate the increase in demand. Finally, in the telecommunication sector, a real-time environment that could require the quick solution of UFLP concerns the future 5G cellular networks. The research community, together with standardization organizations, has posed the need for the densification of the radio access network, so that the current set of deployed base stations will be complemented with a tier of small cells to provide high capacity (Bartelt *et al*, 2015; Wang *et al*, 2015). The right location and activation of these small cells according to the customers behavior could involve large energy and monetary savings.

3. Literature review on stochastic location problems

This section presents a review of some approaches to the different variants of the SUFLP. Usually the values that can take the inputs of combinatorial optimization problems are not deterministic in real life. In the case of the UFLP, a variety of sources of uncertainty may appear. In this regard, Snyder (2006) reviewed the FLP with stochasticity, illustrating both the rich variety of approaches for optimization under uncertainty that have appeared in the literature and their application to facility location problems. Later, Arabani and Farahani (2012) reported some aspects and characteristics of the dynamics of FLPs, dedicating a section to the probabilistic, fuzzy, and stochastic versions of the problem in the literature. Recently, Correia and da Gama (2015) discussed different modeling frameworks for the facility location under uncertainty, distinguishing between robust optimization, stochastic programming, and chance-constrained models. None of these three surveys are specifically devoted to review the UFLP, but the different variants of the FLP, dealing with the UFLP in some marginal parts. Here, we will focus on the stochasticity in the UFLP and some of its most interesting variants.

Our proposal solves the UFLP where service costs are considered stochastic. A similar stochastic variant of the problem is tackled by Verma *et al* (2010), who adopt fuzzy theory for dealing with the uncertainty. However, only very small instances of the problem can be solved with their approach due to the complexity of the problem. Although other approaches in the literature dealt with the UFLP under stochastic environments, they work with other features and variants of the uncapacitated version. Thus, for instance, Ravi and Sinha (2004) formulated the problem in the framework of two-stage stochastic optimization so that the demand of each customer in the UFLP is not known at the first stage. In each scenario, a customer has a demand which may be zero. Each facility has a first-stage opening cost and recourse costs in other scenarios. These may be infinity, reflecting the unavailability of the facilities in various scenarios. These authors provided a nearly tight approximation algorithm to solve it.

Disruptions are other features incorporated in some UFLP. Drezner (1987) introduced them in some FLPs for the first time. One interesting recent research in this line was developed in Lu *et al* (2015). In their work, authors used a model that allows disruptions to be correlated with an uncertain joint distribution, and they applied distributionally robust optimization to minimize the expected cost under the worst-case distribution with given marginal disruption probabilities.

Another variant of the problem with uncertainty is the competitive FLP, used for commercial facilities, *e.g.*, shops and stores. In this case, the objective of a decision maker is mainly to obtain as many demands for his/her facilities as possible. Uno *et al* (2010) presented a fuzzy method to address this problem with stochastic demands. The demands for facilities are represented as fuzzy random variables. For a review of competitive location models, see the book by Drezner (1995), and the review by Eiselt *et al* (1993).

In Contreras *et al* (2011), the uncapacitated hub location problem is addressed. In this variant of the problem, it is assumed that flows originating at the same node but having different destination points can be routed through different sets of hub nodes, *i.e.*, a multiple assignment pattern applies. This way the objective is to minimize the sum of the hub fixed costs and demand routing costs. These authors demonstrate that the stochastic problems with uncertain demands or dependent transportation costs are equivalent to their associated deterministic expected value problem in which random variables are replaced by their expectations. In the case of uncertain independent transportation costs, the corresponding stochastic problem is not equivalent and specific solution methods need to be developed.

In order to incorporate the level of risk aversion into the decision-making process, some works use a mean–variance objective function. Thus, Jucker and Carlson (1976) used it in a stochastic formulation of the UFLP in which selling price (and hence demand) may be random. Later, Hodder and Jucker (1985) extended this model to allow for correlation among random prices. Wagner *et al* (2009) provided an exact method for a problem where demands are probabilistic and correlated. It is a mean–variance approach that balances two often conflicting objectives: profit and associated uncertainty. The objective of locating the facilities is to maximize the lower limit of future earnings based on a stated confidence level.

Finally, some papers place constraints on the maximum regret that may be attained by the solution (Kouvelis *et al*, 1992). The term of p-robustness to measure it was introduced by Snyder and Daskin (2006). In Snyder *et al* (2007), they combined the minimum-expected-cost and p-robustness measures for the UFLP with the aim of finding the minimum-expected-cost solution that is p-robust. They solved their models using Lagrangian decomposition.

Notice that none of these works in the literature dealing with variants of the stochastic UFLP applied the combination of simulation and metaheuristics proposed in this work, which means that our approach is novel in this field.

4. Optimal solutions for the UFLP

In order to evaluate the quality of the results provided by heuristic algorithms, different authors have used lower bound values of the optimal values as a reference. However, to the best of our knowledge, the optimal values for some of the largest instances in the literature have not been reported so far. Therefore, another contribution of this paper is the publication of these optimal values, for all the classical benchmarks, through the application of a MIP model which uses the *Gurobi* state-of-the-art solver (see Table 1). In order to use this solver, we have implemented the MIP model into a Python script. By comparing against optimal values instead of lower bounds, we can assess more accurately the quality of the solutions obtained via approximate methods.

A mathematical model for the UFLP was provided by Erlenkotter (1978). The set of clients is I, and the set of possible locations for the facilities is J. Costs f_j are the fixed costs for installing facility j, and c_{ij} are the costs incurred if customer i is served from facility j. Binary variables y_j will be 1 if location j is used, and 0 otherwise, for $j \in J$. Variables x_{ij}, also binary, take the value of 1 if customer i is served from facility j, and 0 otherwise, for $i \in I, j \in J$. The objective (1a) is to minimize the sum of both fixed and variable costs. Constraints (1b) state that all the customers must be satisfied by exactly one facility. Constraints (1c) impose that if a facility is used to serve some customer, then it must be opened.

$$\text{minimize} \quad \sum_{j\in J} f_j y_j + \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} \tag{1a}$$

$$\text{subject to:} \quad \sum_{i\in I} x_{ij} = 1 \quad \forall i \in I \tag{1b}$$

$$x_{ij} \le y_j \quad \forall i \in I, j \in J \tag{1c}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in I, j \in J \tag{1d}$$

$$y_j \in \{0,1\} \quad \forall j \in J \tag{1e}$$

We have developed a Python-based implementation of this mathematical model ready to be used in the *Gurobi* commercial solver. As it will be shown in the experimental section, for some of the largest instances several hours of computation are required in order to obtain the optimal value. In some real-life UFLP applications (*e.g.*, in the telecommunication or computing sectors), a solution might be required after a few milliseconds or seconds. These applications justify the need for fast heuristics. Additionally, these heuristics can be further extended to simheuristics in order to account for the uncertainty behavior that characterizes most real-life applications.

5. A savings-based heuristic for the UFLP

As the initial stage in the process of developing a simheuristic algorithm is able to deal with the SUFLP, we have designed a novel constructive heuristic for the deterministic UFLP. Unlike the few heuristics available in the UFLP literature (Hoefer, 2014), our heuristic is relatively easy to understand. Also, as shown in Table 1, it offers an outstanding performance in terms of computing times, providing quite competitive results even for the largest instances in the literature (average gap around 3% with respect to the optimal solutions) in just a few milliseconds. Thus, this heuristic constitutes a good alternative for real-life UFLP applications in which real-time solutions might be required.

Table 1 Efficiency of the proposed heuristic and ILS-based approach for the UFLP

Instance	*Gurobi (optimal)*		*Savings-based heuristic*			*ILS-based approach*					*Resende and Werneck (2006)*		
	Cost (1)	*Time (s)*	*Cost (2)*	*Time (s)*	*Gap (%) (2)-(1)*	*Avg cost (3)*	*Best cost (4)*	*Time (s)*	*Gap (%) (3)-(1)*	*Gap (%) (4)-(1)*	*Avg cost (5)*	*Time (s)*	*Gap (%) (5)-(1)*
500-10	798577	15.54	816911	0.01	2.30	799077.1	798577	46.89	0.06	0.00	798577	33.20	0.00
500-100	326790	12.69	334182	0.01	2.26	327007.6	326894	18.54	0.07	0.03	326805.4	32.90	0.00
500-1000	99169	15.12	99630	0.00	0.46	99172.6	99169	14.57	0.00	0.00	99169	23.60	0.00
1000-10	1434154	516.45	1496913	0.03	4.38	1435918.6	1434154	54.94	0.12	0.00	1434185.4	173.90	0.00
1000-100	607878	117.6	623351	0.03	2.55	608202.3	607939	160.76	0.05	0.01	607880.4	148.80	0.00
1000-1000	220560	84.18	225232	0.02	2.12	221260.9	220959	175.71	0.32	0.18	220560.9	141.70	0.00
1500-10	2000801	12109.82	2081386	0.06	4.03	2002874.5	2001920	178.66	0.10	0.06	2001121.7	347.80	0.02
1500-100	866454	286.26	900077	0.06	3.88	867654.7	866769	385.46	0.14	0.04	866493.2	378.70	0.00
1500-1000	334962	211.74	344279	0.05	2.78	338046.1	337452	381.83	0.92	0.74	334973.2	387.20	0.00
2000-10	2558118	2407.97	2683346	0.12	4.90	2559611.3	2558125	224.77	0.06	0.00	2558120.8	717.50	0.00
2000-100	1122748	488.35	1166994	0.12	3.94	1125471.9	1124572	980.89	0.24	0.16	1122861.9	650.80	0.01
2000-1000	437686	419.54	450549	0.11	2.94	443025.7	442136	997.64	1.22	1.02	437690.7	760.00	0.00
2500-10	3099907	240588.51	3223279	0.22	3.98	3107032.5	3102719	710.01	0.23	0.09	3100224.7	1419.50	0.01
2500-100	1347516	1534.58	1398526	0.20	3.79	1350446.6	1349510	1903.79	0.22	0.15	1347577.6	1128.20	0.00
2500-1000	534405	758.51	547825	0.18	2.51	540365.2	539485	1940.58	1.12	0.95	534426.6	1309.40	0.00
3000-10	3570766	2960.9	3714590	0.31	4.03	3579295.7	3570930	2605.87	0.24	0.00	3570818.8	1621.10	0.00
3000-100	1602154	8873.8	1653616	0.29	3.21	1607502.6	1606424	2987.58	0.33	0.27	1602530.9	1977.60	0.02
3000-1000	643463	1692.01	660541	0.28	2.65	652092.7	650777	2996.92	1.34	1.14	643541.8	2081.40	0.01
Avg		15171.86		0.12	3.15			931.41	0.37	0.26		740.73	0.01

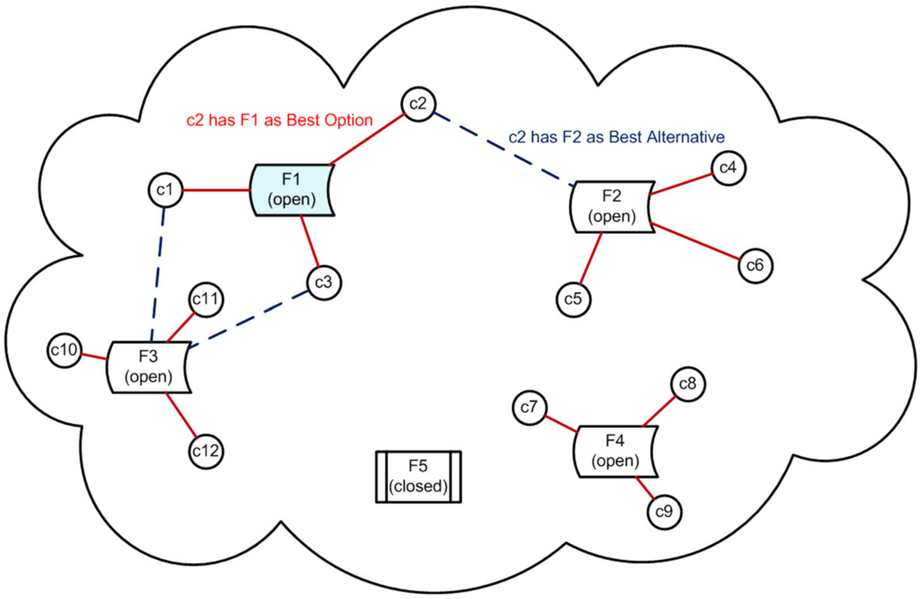

Figure 2 Best and best alternative assignments for a customer.

Notice that, in general, closing a facility is computationally less expensive than opening a facility. Indeed, if an open facility is closed, only the customers that were assigned to it need to be re-allocated to alternative open facilities. On the contrary, if a closed facility is open, then all customers in the system need to be evaluated in order to decide which of them should be assigned to it. Accordingly, the proposed heuristic is based on the concept of cost savings associated with closing a given facility in the current configuration of open facilities. The savings of closing a given facility can be either a positive or a negative value. In particular, the savings related to closing a facility in the current configuration are computed as follows: the cost of opening the facility, plus the assignment cost of its customers, minus the re-allocation cost of its customers to alternative facilities. For each customer, its best alternative facility is the open facility with a minimum re-allocation cost in case its current facility is closed. Thus, for example, in Figure 2, customers $c1$, $c2$, and $c3$ are assigned to facility $F1$, but if $F1$ is closed they will be assigned to $F2$ and $F3$, respectively. In this case, we would save the opening cost of $F1$, as well as the assignment cost of $c1$, $c2$, and $c3$ to $F1$; however, we would have to account for the re-allocation cost of $c1$ and $c3$ to $F3$, as well as the re-allocation cost of $c2$ to $F2$.

Our constructive heuristic works as follows (Figure 3). Given an UFLP instance, we consider an initial system configuration in which all the facilities are open. Then, using this initial configuration as a reference, we compute the cost savings associated with closing each individual facility while keeping all the others open. This way, we obtain a list of possible closures, which is then sorted by decreasing savings value. This is henceforth called the *savings list*. Afterward, starting from the initial system configuration (the one with all facilities open), the savings list is iteratively traversed from the beginning and until

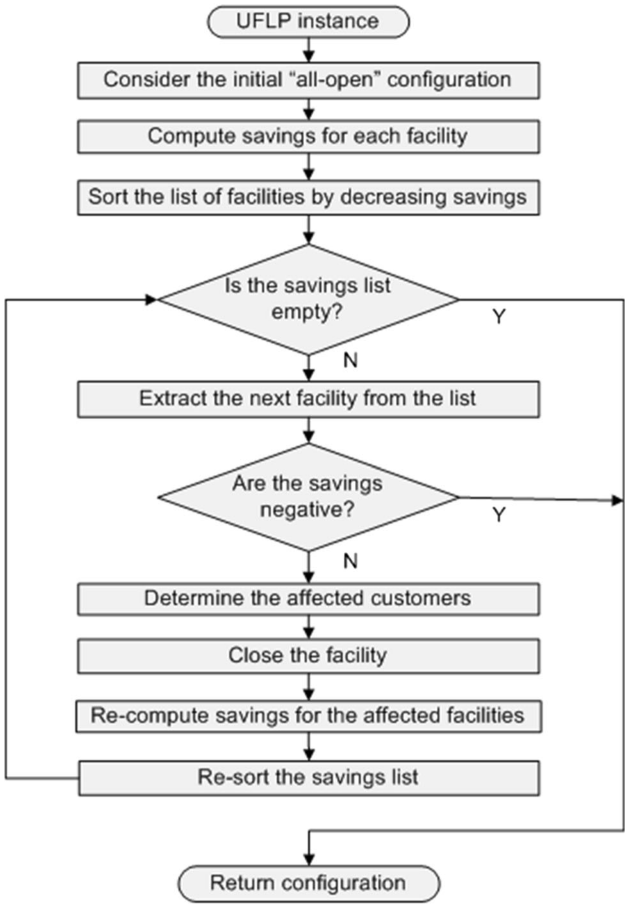
UFLP instance
Consider the initial "all-open" configuration
Compute savings for each facility
Sort the list of facilities by decreasing savings
Is the savings list empty?
Y
N
Extract the next facility from the list
Are the savings negative?
Y
N
Determine the affected customers
Close the facility
Re-compute savings for the affected facilities
Re-sort the savings list
Return configuration

Figure 3 Flow chart of the proposed heuristic approach.

no more positive savings are obtained. At each iteration, the corresponding facility is closed and the savings list is updated according to the new configuration of open facilities. In order to make the savings re-computation process efficient, every customer keeps updated information about its best alternative facility in case the currently assigned one gets closed. Thus, every time an open facility F is closed, the customers directly affected (*i.e.*, the customers that were assigned to F) are quickly re-allocated to their best alternative facilities. Additionally, the customers indirectly affected (*i.e.*, those customers that had F as their best alternative facility) update the information about their best alternative open facility. This way, the savings only need to be re-computed for a reduced set of facilities, the ones associated with the affected customers, which speeds up the computation.

6. Combination with a metaheuristic framework

In this section, we describe how the previous heuristic is integrated into a metaheuristic framework. This allows improving the quality of the generated solution whenever more computing time is allowed. In this paper, the ILS metaheuristic has been chosen. This selection is based on the following criteria: *(i)* it offers a well-balanced combination of efficiency and relative simplicity (Juan *et al*, 2014; Dominguez *et al*, 2016) and *(ii)* it can be easily extended to a simheuristic (Grasas *et al*, 2016). Being an algorithm with few parameters, our ILS-based approach represents an interesting alternative to other state-of-the-art approaches.

The proposed ILS-based approach is detailed in Algorithm 1. It receives the following input parameters: *(i)* the facilities and customers of the instance; *(ii)* the limits for the percentage of facilities to be deleted from the current base solution during the perturbation phase; and *(iii)* a stopping criterion (maximum number of iterations to run). The algorithm works as follows: first, the savings list is created and sorted (line 1). Then, an initial solution is generated using our savings-based heuristic (line 2) or, alternatively, a biased-randomized version of it as described in Juan *et al* (2011b) (this option might be preferred if parallel computing strategies are required). A local search procedure is applied to refine this initial solution (line 3). This procedure combines closing and opening movements. The solution generated will be used by the ILS framework as the initial base solution. During the ILS iterative process (lines 7–19), the current base solution is perturbated in order to generate a new feasible solution. After a local search process, this new solution is compared against the current base solution and if the former is better or an acceptance criterion is met, then the latter is updated to resume the search from a more promising point in the solution space. At any time, the best solution found so far is also saved, since this is the solution that will be returned at the end. Notice that during the perturbation stage, a random percentage of the current base solution is destroyed (*i.e.*, a random percentage of closed facilities are opened) and then reconstructed (by closing some facilities) to generate a new solution. Also, notice that the acceptance criterion is based on the concept of `credit`. Whenever the current base solution is improved, a credit is assigned by the algorithm. This credit has the same value of the improvement, and it poses a limit on the quantity that the current base solution could be worsened during the next iteration.

Algorithm 1: ILS-UFLP(facilities, customers, minOpenPer, maxOpenPer, maxIter)

```
1  savingsList ← createSavingsList(facilities, customers)
2  initSol ← genInitSol(savingsList)
3  baseSol ← localSearch(initSol)
4  bestSol ← baseSol
5  nIter ← 0
6  credit ← 0
7  while (nIter ≤ maxIter) do
8      newSol ← perturbate(baseSol, minOpenPer, maxOpenPer)
9      newSol ← localSearch(newSol)
10     delta ← cost(newSol) − cost(baseSol)
11     if (delta < 0) then
12         credit ← -(delta)
13         baseSol ← newSol
14         if (cost(newSol) < cost(bestSol)) then
15             bestSol ← newSol
           end
       else
16         if (delta > 0) and (credit ≥ delta) then
17             credit ← 0
18             baseSol ← newSol
           end
       end
19     nIter ← nIter + 1
   end
20 return bestSol
```

Algorithm 2: createSavingsList(facilities, customers)

```
1 sol ← getSolutionAllFacilitiesOpen()
2 nFacilities ← size(facilities)
3 for (i in 1 ... nFacilities) do
4     fac1 ← getFacility(facilities, i)
5     savings ← calcSavingAfterClosing(sol, fac1)
6     addToList(savingsList, savings)
  end
7 decreasingSortBySavings(savingsList)
8 return savingsList
```

For implementation purposes, the pseudocodes of the auxiliary methods employed by the approach are included and briefly explained next. The `createSavingsList` method (Algorithm 2) is used to create and sort the list of savings. The `genInitSol` method (Algorithm 3) is responsible for generating the initial solution, which serves as a starting point for the algorithm. The `perturbate` method (Algorithm 4) implements the perturbation operator used within the main loop of the algorithm. First, it starts a destruction phase, so that a set of new facilities are

Algorithm 3: genInitSol(savingsList)

```
1  sol ← getSolutionAllFacilititesOpen()
2  while (notEmpty(savingsList)) do
3      savingsElem ← removeFirstElem(savingsList)
4      facility ← getFacility(savingsElem)
5      savings ← getSaving(savingsElem)
6      if (savings > 0) then
7          directAffectedCustomers ← getDirectAffectedCustomers()
8          indirectAffectedCustomers ← getIndirectAffectedCustomers()
9          sol ← removeFacility(sol, facility)
10         updateSavingsDirect(directAffectedCustomers)
11         updateSavingsIndirect(indirectAffectedCustomers)
12         decreasingSortBySavings(savingsList)
       end
   end
13 return sol
```

Algorithm 4: perturbate(baseSol, minOpenPer, maxOpenPer)

```
1 sol ← baseSol
  // destruction phase
2 minFacilToOpen ← rand(0, minOpenPer * size(getOpenFacilities(sol)))
3 maxFacilToOpen ← rand(0, maxOpenPer * size(getOpenFacilities(sol)))
4 nFacilToOpen ← rand(minFacilToOpen, maxFacilToOpen)
5 for (i in 1 ... nFacilToOpen) do
6     facilToOpen ← randPick(getClosedFacilities(sol))
7     sol ← constructSol(sol, facilToOpen)
  end
  // construction phase
8 sol ← pathRelinking(baseSol, sol)
9 return sol
```

randomly chosen to be included in the current base solution. Their number is determined by the minimum destruction percentage and the maximum destruction percentage. Finally, a reconstruction phase is applied, using the `pathRelinking` procedure. Path relinking was originally proposed in the context of the tabu search metaheuristic (Glover, 1997), and it is based on the generation of new solutions by exploring trajectories that connect high-quality solutions. Here, the `pathRelinking` method (Algorithm 5) applies a simple form of path relinking between two solutions. The `localSearch` method (Algorithm 6) relies on a loop structured in two different blocks that are kept running consecutively until a non-improving iteration is reached. In the first block, an open facility is tentatively removed from the solution to search for an improvement in total cost. In the second block, a closed facility is tentatively incorporated to the configuration of open facilities.

Algorithm 5: pathRelinking(aSol, bSol)

```
toInsertList ← setDiv(bSol, aSol)
toRemoveList ← setDiv(aSol, bSol)
do
    hasImproved ← false
    max ← 0
    bestInsertFacility ← null
    // Insert facility involving highest savings
    for (facility in toInsertList) do
        savings ← calcSavingsAfterOpening(facility)
        if ($savings > max$) then
            max ← savings
            bestInsertFacility ← facility
        end
    end
    if ($bestInsertFacility \neq null$) then
        aSol ← constructSol(aSol, bestInsertFacility)
        hasImproved ← true
    end
    removeElem(toInsertList, bestInsertFacility)
    max ← 0
    bestRemoveFacility ← null
    // Remove facility involving highest savings
    for (facility in toRemoveList) do
        savings ← calcSavingsAfterClosing(facility)
        if ($savings > max$) then
            max ← savings
            bestRemoveFacility ← facility
        end
    end
    if ($bestRemoveFacility \neq null$) then
        aSol ← removeFacility(aSol, bestRemoveFacility)
        hasImproved ← true
    end
    removeElem(toRemoveList, bestRemoveFacility)
while (hasImproved)
return aSol
```

Algorithm 6: localSearch(sol)

```
1 do
2     accumulatedbalance ← 0
      // Try to remove facilities
3     do
4         balance ← tryRemoveFacility(sol)
5         accumulatedBalance ← accumulatedBalance + balance
      while (balance > 0)
      // Try to add facilities
6     do
7         balance ← tryAddFacility(sol)
8         accumulatedBalance ← accumulatedBalance + balance
      while (balance > 0)
  while (accumulateBalance > 0)
9 return sol
```

7. Computational experiments for the UFLP

To evaluate and assess the performance of the proposed algorithms, several computational experiments were performed. The proposed algorithms have been implemented as Java® 7SE applications. All tests have been executed on a standard desktop computer with an Intel® Core™ i5 at 2.4 GHz and 8 GB RAM running on Windows 7. The largest instances in the UFLP literature have been used, since they are the most challenging ones. To the best of our knowledge, the results of these instances have not been improved from 2006 (Resende and Werneck, 2006). Therefore, they are the perfect benchmarks to test the quality of our algorithm. This set of instances is called MED. They were originally proposed for the *p*-median problem by Ahn *et al* (1988) and later used in the context of the UFLP by Barahona and Chudak (1999). Each instance is a set of n points picked uniformly at random in the unit square. A point represents both a user and a facility, and the corresponding Euclidean distance determines connection costs. The set 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 for each subset ($\sqrt{n}/10$, $\sqrt{n}/100$, and $\sqrt{n}/1000$ corresponding to 10, 100, and 1000 instance suffixes, respectively). In order to introduce customer demands in these instances (since they will be used later during the stochastic experiments), we have divided the assignment costs of each customer to each facility by the expected customer demand, so that the assignment cost per unit is obtained.

As mentioned before, our MIP model implementation has been run in order to generate the optimal solutions for this set of instances. Afterward, we have executed the proposed savings-based heuristic for each problem instance. Finally, our ILS-based approach has been run. In particular, each instance has been run 30 times, each time employing a different seed for the pseudorandom number generator. Both the best and average solutions found by our algorithm are reported jointly with the average time for the 30 runs. Table 1 depicts all these results using the benchmarks mentioned above. Each row corresponds to a single instance. The first two columns show the Gurobi results, the next three columns depict the savings-based heuristic results, five columns are used for the ILS-based approach results, and finally Resende and Werneck (2006) results are in the last three columns. Times are shown in seconds and gaps are calculated as follows:

$$Gap(Cost, Cost_{Optimal}) = 100\left(\frac{Cost - Cost_{Optimal}}{Cost_{Optimal}}\right) \qquad (2)$$

Notice that our savings-based heuristic approach provides reasonably good solutions in just a few milliseconds, with and average gap of 3.15% with respect to the optimal values. Additionally, our ILS-based approach provides near-optimal solutions (average gap of 0.26% with respect to the optimal values). Also, notice that the best and the average costs provided are quite similar, which supports the idea that our algorithm is quite robust in that sense.

Thus, although the performance of the algorithm proposed by Resende and Werneck (2006) is still somewhat superior, the difference of performance with respect to our ILS-based approach seems to be small, which allows us to use it as a base for our simheuristic approach.

8. Extending to a simheuristic for the SUFLP

Once it has been verified that our ILS-based approach is able to provide competitive solutions in short computing times (average gap below 1% after a few seconds in a standard computer), it can be extended to a simheuristic algorithm for solving the SUFLP, when solutions offering a good trade-off between total expected cost and variability or risk might be desirable. Now, instead of considering a deterministic service cost d_i for a customer $i (\forall i \in \{1, 2, \ldots, |C|\})$, this service cost will be modeled as a random variable D_i with $E[D_i] = d_i$.

As discussed in Juan *et al* (2015), one natural way to extend heuristic algorithms, so they can deal with stochastic versions of combinatorial optimization problems, is by integrating simulation inside the metaheuristic framework. In the case of an ILS metaheuristic framework, a detailed discussion of this hybridization process can be found in Grasas *et al* (2016). Accordingly, in this paper we have integrated our ILS-based approach with Monte Carlo simulation (MCS). The MCS does not only provide estimates to the expected cost associated with the solutions generated by the approach, but it also reports feedback to the stochastic search process. Indeed, the selection of the current base solution is driven by the results of the MCS.

Algorithm 7 depicts our simheuristic proposal. On the one hand, the *detCost* routine returns the deterministic cost of a solution, i.e., the solution cost using the deterministic service costs specified by the UFLP instance. On the other hand, the *stochCost* routine returns the average cost of the simulations performed over that solution (i.e., an estimate of its expected cost). Observe that, after considering that a new solution is better than the current base solution in the deterministic environment (lines 11–12 with *deltaDet* storing the difference), a fast simulation is performed (line 13) in order to check its goodness in the stochastic environment (lines 14–15 with *deltaStoch* storing the difference).

Algorithm 7: SimILS-UFLP(facilities, customers, minOpenPer, maxOpenPer, maxIter)

```
1  savingsList ← createSavingsList(facilities, customers)
2  initSol ← genInitSol(savingsList)
3  baseSol ← localSearch(initSol)
4  bestSol ← baseSol
5  fastSimulation(bestSol)
6  nIter ← 0
7  credit ← 0
8  while (nIter ≤ maxIter) do
9      newSol ← perturbate(baseSol, minOpenPer, maxOpenPer)
10     newSol ← localSearch(newSol)
11     deltaDet ← detCost(newSol) – detCost(baseSol)
12     if (deltaDet < 0) then                                   // newSol is promising
13         fastSimulation(newSol)
14         deltaStoch ← stochCost(newSol) - stochCost(baseSol)
15         if (deltaStoch < 0) then                             // newSol improves baseSol
16             baseSol ← newSol
17             credit ← -(deltaStoch)
18             if (stochCost(newSol) < stochCost(bestSol)) then // newSol improves bestSol
19                 bestSol ← newSol
20                 insert(poolBestSol, bestSol)
               end
           end
       end
21     else if (credit ≥ deltaDet) then          // newSol is expected to be within the credit
22         baseSol ← newSol                                     // degrade baseSol
23         credit ← 0
       end
24     nIter ← nIter + 1
   end
25 for (sol ∈ poolBestSol) do
26     deepSimulation(sol)
   end
27 return poolBestSol
```

In case the new solution is better in the stochastic environment, the current base solution is updated (line 16) and a credit is fixed (line 17). The best-found solution is also updated if the new solution improves its performance in the stochastic environment (lines 18–19). A pool of best-found solutions (*poolBestSol*) is kept in order to analyze them later (line 20). If the new solution is not better than the current base solution in terms of deterministic cost, we can still consider the degradation of the current base solution. This allows the algorithm to escape from stagnation (line 21). In order to avoid a degradation after another, the credit is reestablished to zero each time it is used (line 23). Finally, larger simulations are performed over a reduced set of the best-found solutions. This allows to obtain more accurate estimates of the different parameters associated with the stochastic costs generated by each of these solutions, *e.g.*, standard deviation, quartiles, etc. (lines 25–26).

In order to test our simheuristic algorithm in the SUFLP environment, we have extended the deterministic benchmark instances by employing the log-normal probability distribution for modeling the stochastic service costs. In a real-world application, historical data could be used to model each service cost by a different probability distribution. As discussed in Juan *et al* (2011a), the log-normal distribution is a more natural choice than the normal distribution when modeling nonnegative random variables. The log-normal has two parameters, namely: the location parameter, μ_i, and the scale parameter, σ_i. According to the properties of the log-normal distribution, these parameters will be given by the following expressions:

$$\mu_i = \ln(E[D_i]) - \frac{1}{2}\ln\left(1 + \frac{Var[D_i]}{E[D_i]^2}\right)\sigma_i = \left|\sqrt{\ln\left(1 + \frac{Var[D_i]}{E[D_i]^2}\right)}\right|$$

As stated before, $E[D_i] = d_i, \forall i \in \{1, 2, \ldots, |C|\}$. For computational experiments, we have considered a variance level (uncertainty) $Var[D_i] = 50d_i$. Observe that the original UFLP instances are a particular case of these new instances when $Var[D_i] = 0, \forall i \in \{1, 2, \ldots, |C|\}$.

During the stochastic searching process, only simulations with a reduced number of runs (5000 in our experiments) are employed. This way, we avoid the simulation to jeopardize the computing time of the ILS-based approach. Once a reduced set of promising solutions for the SUFLP has been selected, a simulation with more runs (100,000 in our case) is employed to obtain more accurate estimates of the different parameters associated with the stochastic costs generated by each of these solutions. Actually, simulation is not only used here to generate estimates of the expected cost associated with each solution and to guide the searching process of the heuristic component, but it is also employed to generate observations on the stochastic behavior of each solution. As suggested in Juan *et al* (2015), these observations can then be analyzed to perform a risk analysis on each of these promising solutions.

Notice that, in the basic SUFLP with a linear objective function, the solution minimizing 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 SUFLP objective function to account for facilities with a total demand higher than a threshold, etc.). Nevertheless, even in this basic SUFLP variant, the decision maker will be interested not only in the solution that minimizes the total expected cost but also in other solutions that might offer a better trade-off between total expected cost and ‘robustness’ (in terms of assuming a reasonably low variability or risk). For this purpose, our approach is able to generate several alternative solutions, each of them offering different values for each of the parameters considered. In particular, for any given instance of the SUFLP we are interested in obtaining: *(i) det*, the optimal solution for the deterministic UFLP when applied to the stochastic environment (as discussed before, for the basic linear version of the SUFLP this solution will be also the one with the minimum expected cost); *(ii) min-avg*, the solution with minimum expected cost found by the simheuristic algorithm (for validation purposes, it might be useful to compare *min-avg* with *det*); *(iii) min-q3*, the solution with minimum third quartile, where the third quartile refers to the observed costs after multiple executions of a solution; and *(iv) min-dev*, a solution offering a good trade-off between expected cost and standard deviation, i.e., among those solutions with a low expected cost, we are interested in identifying the ones also offering low variability or risk.

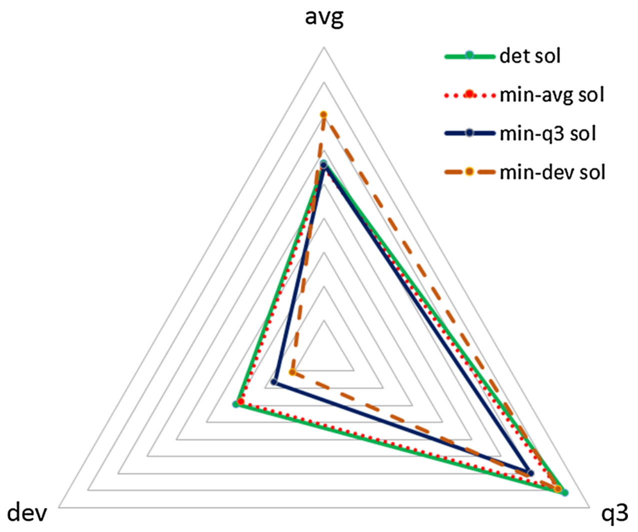

Figure 4 Comparison of solutions for instance 1000_10.

For the *1000_10* instance, Figure 4 depicts the aforementioned four solutions in a radar-like graph. Notice that each solution offers different values in each of the three dimensions considered. In particular, *(i)* the *min-avg* solution and the *det* solution are overlapping, which contributes to validate our approach; and *(ii)* both the *min-avg* and the *det* solutions offer a poor performance, in terms of standard deviation and third quartile, when compared with *min-dev* and *min-q3*, respectively.

As mentioned before, once the simheuristic approach has found the solutions with best performance for each dimension, simulation with more runs (100,000 in our case) is employed in the last step to obtain more accurate values. Therefore, we can develop a multiple boxplot comparison of the cost obtained through this large simulation for the aforementioned four solutions. Figure 5 shows this multiple boxplot comparison for the *1500-10* instance. For each boxplot, it also includes the number of facilities open in the corresponding solution.

Again, it can be observed that solution *min-avg* is quite similar to solution *det*, both in average cost and variability as well as in number of open facilities. However, solution *min-q3* seems completely different: by accepting a somewhat higher expected cost, it offers less variability (risk). In particular, 75% of the times it is applied will provide cost levels that are considerably lower than the third quartile associated with the *det* solution. Notice that this reduction in risk is achieved by increasing the number of open facilities: by increasing this number, the network becomes more ’resilient’ to high demands that could significantly increase service costs if only a few facilities were open. Finally, the *min-dev* solution illustrates how it is possible to find alternative solutions combining different cost variability and expected cost levels. According to the results, it seems that in order to reduce the cost variability more facilities are needed, thus reducing the

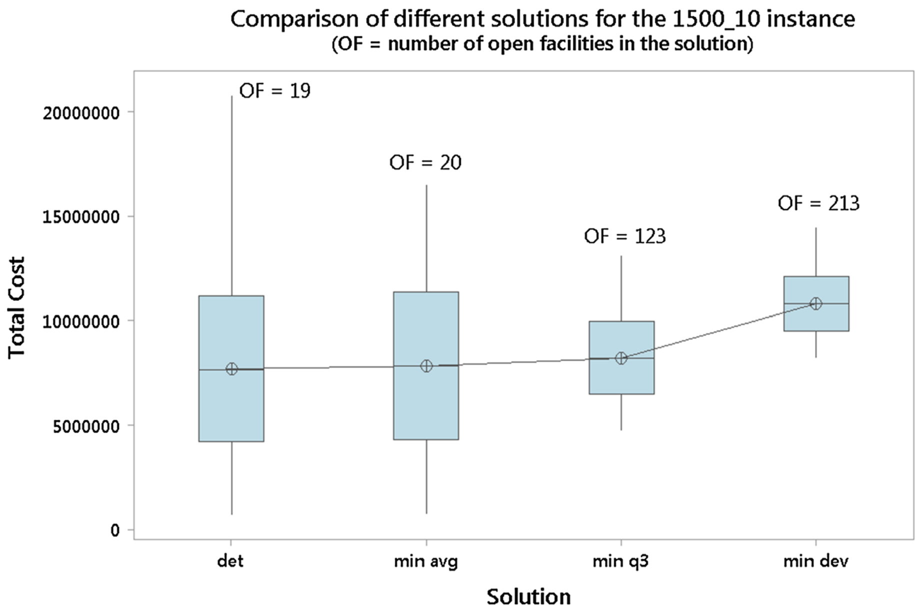

Figure 5 Comparison of the solutions obtained by the simheuristic approach.

expected service cost (although increasing somewhat the opening cost and, eventually, the total expected cost).

9. Conclusions

In this paper, a simheuristic algorithm for solving the stochastic uncapacitated facility location problem (SUFLP) has been developed. In the SUFLP, service costs are assumed to be random variables. The simheuristic development process is based on the following logic: first, we have obtained optimal solutions for the large-sized instances of the deterministic version of the problem (UFLP). To the best of our knowledge, these optimal values had never been reported before in the UFLP literature. Unfortunately, obtaining optimal values for large-sized instances requires computing times that are beyond the available times in some practical applications of the UFLP. For that reason, we have also developed a fast savings-based heuristic able to provide reasonably good solutions in a few milliseconds. This heuristic might be useful in some real-life applications whenever instantaneous solutions are required. Then, this heuristic has been extended to a competitive ILS-based approach, which is able to provide near-optimal solutions in short time. Finally, we have built a simheuristic algorithm for the SUFLP by integrating the metaheuristic with Monte Carlo simulation techniques. The simulation stage is not only used to estimate the stochastic behavior of the solutions generated by the ILS-based approach, but its feedback is used by the approach to drive the stochastic search process.

As illustrated with several examples, even in the case of basic versions of the SUFLP with linear objective functions, our simheuristic approach can be used to provide alternative solutions to the one with the minimum expected cost, *e.g.*, the solution that minimizes a given percentile, or solutions with different trade-off levels of expected cost and variability. The methodology can also be applied to more advanced versions of the SUFLP with nonlinear penalty costs and/or probabilistic constraints, where the optimal solution to the deterministic UFLP might provide a suboptimal value for the minimum expected cost.

This work can be extended in several directions. In first place, other more advanced variants of the SUFLP could be analyzed. Also, similar approaches to the one introduced in this paper could be used to solve stochastic versions of: *(i)* the capacitated facility location problem; *(ii)* the *p*-median problem; or *(iii)* the location routing problem, where also vehicle routing plans need to be accounted for.

Acknowledgements—This work has been partially supported by the Spanish Ministry of Economy and Competitiveness (TRA2013-48180-C3-P, TRA2015-71883-REDT), and FEDER. Likewise, we want to acknowledge the support received by the Department of Universities, Research & Information Society of the Catalan Government (2014-CTP-00001).

References

Ahn S, Cooper C, Cornuejols G and Frieze A (1988). Probabilistic analysis of a relaxation for the k-median problem. *Mathematics of Operations Research* **13**(1):1–31.

Alves M and Almeida M (1992). Simulated annealing algorithm for the simple plant location problem. *Revista Investigação Operacional* **12**(2):145–157.

Arabani AB and Farahani RZ (2012). Facility location dynamics: An overview of classifications and applications. *Computers & Industrial Engineering* **62**(1):408–420.

Balachandran V and Jain S (1976). Optimal facility location under random demand with general cost structure. *Naval Research Logistics Quarterly* **23**(3):421–436.

Balinski M (1966). On finding integer solutions to linear programs. In: *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems*, pp. 225–248.

Barahona F and Chudak F (1999). Near-optimal solutions to large scale facility location problem. Technical Report RC21606. IBM, Yorktown Heights, NY, USA.

Bartelt J, Rost P, Wubben D, Lessmann J, Melis B and Fettweis G (2015). Fronthaul and backhaul requirements of flexibly centralized radio access networks. *IEEE Wireless Communications* **22**(5):105–111.

Benders J (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* **4**(1):238–252.

Brahimi N and Khan SA (2014). Warehouse location with production, inventory, and distribution decisions: A case study in the lube oil industry. *4OR* **12**(2):175–197.

Chudak FA (1998). Improved approximation algorithms for uncapacitated facility location. In: Bixby R, Boyd E, Ríos-Mercado R (Eds.) *Integer Programming and Combinatorial Optimization*. Springer, Berlin, Heidelberg. volume 1412 of *Lecture Notes in Computer Science*, pp. 180–194.

Contreras I, Cordeau JF and Laporte G (2011). Stochastic uncapacitated hub location. *European Journal of Operational Research* **212**(3):518–528.

Cornuejols G, Fisher ML and Nemhauser GL (1977). Exceptional paper—location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. *Management Science* **23**(8):789–810.

Cornuejols G, Nemhauser G and Wolsey L (1990). The uncapacitated facility location problem. In: Mirchandani PB, Francis RL (Eds.) *Discrete Location Theory*. Wiley, New York, pp. 119–171.

Correia I and da Gama FS (2015). Facility Location Under Uncertainty. Springer International Publishing, Cham. pp. 177–203.

Dominguez O, Juan AA, Barrios B, Faulin J and Agustin A (2016). Using biased randomization for solving the two-dimensional loading vehicle routing problem with heterogeneous fleet. *Annals of Operations Research* **236**(2):383–404.

Drezner Z (1987). Heuristic solution methods for two location problems with unreliable facilities. *Journal of the Operational Research Society* **38**(6):509–514.

Drezner Z (Ed.) (1995). Facility location: a survey of applications and methods. Springer series in Operations Research and Financial Engineering. Springer, New York.

Efroymson MA and Ray TL (1966). A branch-bound algorithm for plant location. *Operations Research* **14**(3):361–368.

Eiselt HA, Laporte G and Thisse JF (1993). Competitive location models: A framework and bibliography. *Transportation Science* **27**(1):44–54.

Erlenkotter D (1978). A dual-based procedure for uncapacitated facility location. *Operations Research* **26**(6):992–1009.

Fotakis D (2011). Online and incremental algorithms for facility location. *SIGACT News* **42**(1):97–131.

Frank C and Römer K (2007). Distributed Facility Location Algorithms for Flexible Configuration of Wireless Sensor Networks. In: *Proceedings of Third IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS 2007*, Santa Fe, NM, USA, June 18–20, 2007. Springer, Berlin, Heidelberg, pp. 124–141.

Gendreau M, Laporte G, Semet F (2001). A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. Applications of parallel computing in transportation. *Parallel Computing* **27**(12):1641–1653.

Ghosh D (2003). Neighborhood search heuristics for the uncapacitated facility location problem. *European Journal of Operational Research* **150**(1):150–162.

Glover F (1997). Tabu search and adaptive memory programming—advances, applications and challenges. In: Barr RS, Helgason RV, Kennington JL (Eds.) *Interfaces in Computer Science and Operations Research*. Springer, Boston, pp. 1–75.

Grasas A, Juan AA and Ramalhinho H (2016). Simils: A simulation-based extension of the iterated local search metaheuristic for stochastic combinatorial optimization. *Journal of Simulation* **10**(1):69–77.

Hansen P (1976). The simple plant location problem. *Omega* **4**(3):347–349.

Hochbaum DS (1982). Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing* **11**(3):555–556.

Hodder JE and Jucker JV (1985). A simple plant-location model for quantity-setting firms subject to price uncertainty. *European Journal of Operational Research* **21**(1):39–46.

Hoefer M (2014). Max planck institut informatik benchmarks. http://resources.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/.

Jain K, Mahdian M, Markakis E, Saberi A and Vazirani VV (2003). Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *Journal of the ACM (JACM)* **50**(6):795–824.

Jain K and Vazirani VV (1999). Primal-dual approximation algorithms for metric facility location and k-median problems. In: *40th Annual Symposium on Foundations of Computer Science*, 1999, pp. 2–13.

Juan AA, Faulin J, Grasman SE, Rabe M and Figueira G (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives* **2**(1):62–72.

Juan AA, Faulin J, Grasman SE, Riera D, Marull J and Mendez C (2011a). Using safety stocks and simulation to solve the vehicle routing problem with stochastic demands. *Transportation Research Part C: Emerging Technologies* **19**(5):751–765.

Juan AA, Faulin J, Jorba J, Riera D, Masip D and Barrios B (2011b). On the use of Monte Carlo simulation, cache and splitting techniques to improve the Clarke and Wright savings heuristics. *Operational Research Society* **62**(6):1085–1097.

Juan AA, Lourenço HR, Mateo M, Luo R and Castella Q (2014). Using iterated local search for solving the flow-shop problem: Parallelization, parametrization, and randomization issues. *International Transactions in Operational Research* **21**(1):103–126.

Jucker JV and Carlson RC (1976). The simple plant-location problem under uncertainty. *Operations Research* **24**(6):1045–1055.

Kolesar P and Walker WE (1974). An algorithm for the dynamic relocation of fire companies. *Operations Research* **22**(2):249–274.

Körkel M (1989). On the exact solution of large-scale simple plant location problems. *European Journal of Operational Research* **39**(2):157–173.

Kouvelis P, Kurawarwala AA and Gutiérrez GJ (1992). Algorithms for robust single and multiple period layout planning for manufacturing systems. *European Journal of Operational Research* **63**(2):287–303.

Kratica J, Tošic D, Filipović V and Ljubić I (2001). Solving the simple plant location problem by genetic algorithm. *RAIRO-Operations Research* **35**(1):127–142.

Kuehn AA and Hamburger MJ (1963). A heuristic program for locating warehouses. *Management Science* **9**(4):643–666.

Lai MC, Sohn HS, Tseng TLB and Chiang C (2010). A hybrid algorithm for capacitated plant location problem. *Expert Systems with Applications* **37**(12):8599–8605.

Lazic N, Givoni IE, Frey BJ and Aarabi P (2009). Floss: Facility location for subspace segmentation. In: *IEEE 12th International Conference on Computer Vision, ICCV 2009*, Kyoto, Japan, September 27–October 4, 2009, pp. 825–832.

Lee G and Murray AT (2010). Maximal covering with network survivability requirements in wireless mesh networks. *Environment and Urban Systems* **34**(1):49–57.

Li S (2013). A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation* **222**: 45–58.

Lu M, Ran L and Shen ZJ (2015). Reliable facility location design under uncertain correlated disruptions. *Manufacturing and Service Operations Management* **17**(4):445–455.

Mallette AJ and Francis RL (1972). A generalized assignment approach to optimal facility layout. *AIIE Transactions* **4**(2): 144–147.

Maric M, Stanimirovic Z and Bozovic S (2015). Hybrid metaheuristic method for determining locations for long-term health care facilities. *Annals OR* **227**(1):3–23.

Meyerson A (2001). Online facility location. In: *Proceedings of 42nd IEEE Symposium on Foundations of Computer Science*, 2001, pp. 426–431.

Michel L and Van Hentenryck P (2004). A simple tabu search for warehouse location. *European Journal of Operational Research* **157**(3):576–591.

Ravi R and Sinha A (2004). *Hedging Uncertainty: Approximation Algorithms for Stochastic Optimization Problems*. Springer, Berlin, Heidelberg, pp. 101–115.

Resende MG and Werneck RF (2006). A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operational Research* **174**(1):54–68.

Schrage L (1975). Implicit representation of variable upper bounds in linear programming. In: Balinski ML, Hellerman E (Eds.) *Computational Practice in Mathematical Programming*. Springer, Berlin, Heidelberg, pp. 118–132.

Shmoys DB, Tardos É and Aardal K (1997). Approximation algorithms for facility location problems. In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, ACM*, pp. 265–274.

Snyder L and Daskin M (2006). Stochastic p-robust location problems. *IIE Transactions* (Institute of Industrial Engineers) **38**(11):971–985.

Snyder LV (2006). Facility location under uncertainty: A review. *IIE Transactions* **38**(7):547–564.

Snyder LV, Daskin MS and Teo CP (2007). The stochastic location model with risk pooling. *European Journal of Operational Research* **179**(3):1221–1238.

Spielberg K (1969). Algorithms for the simple plant-location problem with some side conditions. *Operations Research* **17**(1):85–111.

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. Ph.D. thesis. University of California at Berkeley.

Thouin F and Coates M (2008). Equipment allocation in video on demand network deployments. *ACM Transaction on Multimedia Computing, Communications and Applications* **5**(1):1–24.

Uno T, Katagiri H and Kato K (2010). A facility location for fuzzy random demands in a competitive environment. *International Journal of Applied Mathematics* **40**(3):172–177.

Verma A, Verma R and Mahanti N (2010). A new approach to fuzzy uncapacitated facility location problem. *International Journal of Soft Computing* **5**(3):149–154.

Verter V (2011). Uncapacitated and capacitated facility location problems. In: Eiselt HA, Marianov V (Eds.) *Foundations of Location Analysis*. Springer, Boston.

Wagner MR, Bhadury J and Peng S (2009). Risk management in uncapacitated facility location models with random demands. *Computers & Operations Research* **36**(4):1002–1011.

Wang D, Wu CH, Ip A, Wang D and Yan Y (2008). Parallel multi-population particle swarm optimization algorithm for the uncapacitated facility location problem using OpenMP. In: *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 1214–1218.

Wang N, Hossain E and Bhargava VK (2015). Backhauling 5g small cells: A radio resource management perspective. *IEEE Wireless Communications* **22**(5):41–49.

Received 14 December 2015;
accepted 16 November 2016