Computers & Industrial Engineering 188 (2024) 109858

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Multi-neighborhood simulated annealing for the capacitated facility location problem with customer incompatibilities

Sara Ceschia, Andrea Schaerf *

DPIA, University of Udine, Via delle Scienze 206, 33100 Udine, Italy

A R T I C L E I N F O

Dataset link: https://github.com/iolab-uniud/ms-cflp-ci

Keywords:
Facility location
Local search
Simulated annealing

A B S T R A C T

We consider the Capacitated Facility Location Problem with Customer Incompatibilities, which is a recently -proposed variant of the classic facility location problem whose distinctive feature is to take into account incompatibilities between customers.

We tackle this problem using local search and we propose a combination of neighborhoods and ad hoc techniques to reduce the size of the search space, in order to effectively deal with large instances. The resulting multi-neighborhood approach is guided by a simulated annealing procedure. Our method, suitably tuned in a statistically-principled way, has been able to outperform all previous techniques on the publicly available dataset, on both short and long running times.

1. Introduction

The Facility Location Problem (FLP) is a classic optimization problem that consists in selecting the locations where new facilities have to be established (among a finite set of available candidate locations) and allocating customers to facilities such that all customer requests are served and the total cost is minimized. It is a strategic issue in logistics and systems design, since it has multitude of applications both in private and public sectors, and many variants of the problem have been proposed and investigated in the literature (Celik Turkoglu & Erol Genevois, 2020; Drezner & Hamacher, 2004; Laporte et al., 2015).

In the simplest version, called Uncapacitated (UFLP), it is assumed that each facility has infinite capacity; as a consequence, once the locations for the facilities have been chosen, each customer is entirely supplied by the open facility with the minimum transportation cost. Despite its simplicity, the UFLP has been proven to be NP-hard by Cornuéjols et al. (1983) and a broad literature exists on this problem: for an overview of the main contributions, see Klose and Drexl (2005), Krarup and Pruzan (1983), Revelle and Laporte (1996) and Verter (2011). The current leading method by Letchford and Miller (2014) is a branch and bound algorithm embedding sophisticated problem reduction procedures, that is able to solve to proven optimality instances with up to 18000 facilities.

From a practical point of view, the most interesting variant of the problem is the capacitated one (CFLP) in which facilities have a fixed maximum capacity that must not be exceeded when supplying customers. In this case, a customer can be supplied by a unique facility (single-source, SS) or by multiple facilities (multi-source, MS).

The literature on the CFLP is extensive, we thus refer the interested readers to the surveys of Fernández and Landete (2015) and Klose and Drexl (2005) for a comprehensive overview. To the best of our knowledge, the state-of-the-art computational results for the CFLP are those reported by Avella et al. (2021), Fischetti et al. (2016) and Görtz and Klose (2012) among exact methods, and Guastaroba and Speranza (2012, 2014) and Caserta and Voß (2020), among heuristics. Görtz and Klose (2012) present a branch-and-bound algorithm based on Lagrangian relaxation and subgradient optimization, (Fischetti et al., 2016) design a Bender decomposition approach and Avella et al. (2021) devise a new class of valid inequalities that are used in a branch-and-cut framework. Top heuristic methods also exploit mathematical programming techniques. In detail, Guastaroba and Speranza implement a Kernel search framework, while Caserta and Voß use a Corridor method. Other relevant works include Avella and Boccia (2009) and Avella et al. (2009) who introduced new datasets with instances up to 2000 facilities, now commonly used as benchmark for comparison. Recently, Weninger and Wolsey (2023) investigated the performance of different Benders based branch-and-cut algorithms for the CFLP with partial single sourcing, which occurs when a subset of customers requires a single supplier.

Approximation results for the CFLP have been obtained by Korupolu et al. (2000), who proved that a *steepest descent* local search heuristic yields a solution of value no more than $(8+\epsilon)$ times the optimal one; this result has been improved to an $6(1+\epsilon)$ approximation by Chudak and Williamson (2005). The neighborhood relations implemented in the

* Corresponding author.
E-mail addresses: sara.ceschia@uniud.it (S. Ceschia), andrea.schaerf@uniud.it (A. Schaerf).

https://doi.org/10.1016/j.cie.2023.109858
Received 3 August 2023; Received in revised form 16 November 2023; Accepted 22 December 2023
Available online 30 December 2023

local search are: opening a new facility, closing a facility, or changing a facility. Notice that these moves modify only the set of open facilities; indeed, for the MS-CFLP, given a set of open facilities, an optimal assignment of customers to facilities can be computed in polynomial time by solving the corresponding instance of the transportation problem. Thus, for this variant of the problem, any solution is completely characterized by the set of open facilities.

Moving to metaheuristic techniques, their application to facility location problems has been surveyed by Basu et al. (2015). Examples of metaheuristic approaches applied to the SS-CFLP are provided by Ahuja et al. (2004), Cortinhal and Captivo (2003) and Chen and Ting (2008), who implemented Tabu Search, Very Large Scale Neighborhood Search and Ant Colony Optimization algorithms, respectively. The neighborhood relations used by Chen and Ting (2008) and Cortinhal and Captivo (2003) are the traditional ones, i.e. the assignment of a customer to a different facility and the swap of the facilities between two customers. Conversely, Ahuja et al. (2004) defined a more sophisticated large neighborhood that exchanges customers among facilities in a cyclic manner; these moves are detected on a *facility improvement graph* dynamically built through the use of a greedy scheme. In addition, Ahuja et al. also implemented the moves that open a new facility, close an existing facility, and transfer a facility to a different location. Finally, Lai et al. (2010) proposed a hybrid approach: a genetic algorithm is embedded in a Benders' decomposition framework to solve the master problem.

Among the different variants of the CFLP, recently Maia et al. (2023) introduced the Multi-Source Capacitated Facility Location Problem with Customer Incompatibilities (MS-CFLP-CI), where there are additional constraints stating that specific pairs of customers cannot be served by the same facility.

These constraints are used to model two main practical situations: *(i)*. rival customers that prevent their suppliers serving also their competitors, assuming the competitive nature of the market; *(ii)*. incompatibility between different types of product required by customers (for example the siting of collection centers for household hazardous waste (Rabbani et al., 2018; Revelle & Laporte, 1996) or perishable products). In addition, consider a *multi-product* setting in which each facility can only store one type of product, and it must be decided which facilities to open, which product store in each facility, and which customers assign to which facilities. This problem can be modeled by introducing incompatibilities between customers who require products of different types.

Customer incompatibilities were investigated from the theoretical point of view by Marín and Pelegrín (2019) in the contest of the uncapacitated problem. In particular, they studied the facial structure of the *set-packing* formulation of the problem, and demonstrated that it generalizes other variants of the UFLP, such as the Fault-Tolerant Facility Location problem, (see e.g. Swamy & Shmoys, 2008) in which each customer has to be assigned to several facilities.

Notice that for the MS-CFLP-CI a search approach that works on the level of deciding which facilities to open, and then assigns the suppliers using a dedicated subprocedure, would not be as promising as for the MS-CFLP. In fact, the presence of incompatibilities introduces disjunctive constraints that make the underlying transportation problem NP-hard, as proven by Goossens and Spieksma (2009).

For the solution of the MS-CFLP-CI Maia et al. (2023) propose a portfolio of different (meta)heuristic techniques which are compared on a new dataset composed of 30 instances, with up to 3000 facilities. All instances and source code of the solution methods are publicly available at https://github.com/MESS-2020-1. Subsequently, two other problem-specific heuristics for the solution of the MS-CFLP-CI have been proposed by Pandey et al. (2023).

The main goal of this work is to explore a new search method to solve the MS-CFLP-CI, which could improve the state-of-the-art results on the available benchmarks. To this aim, we propose a local search algorithm driven by a Simulated Annealing metaheuristic. It starts from an initial solution generated by the greedy algorithm proposed by Maia et al. (2023), properly revised, and then implements some innovative neighborhood structures that, up to our knowledge, have never been applied to the CFLP. In particular, along with the neighborhood that changes a supplier and its supplied quantity of a customer and the neighborhood that swaps the suppliers of two distinct customers, we propose a more complex neighborhood that simultaneously closes a facility and opens a new one. In this case, firstly the facility to be closed is emptied by transferring customers in a greedy way to the cheapest open facility with enough space, then the new open facility attracts new customers, if their shipping cost is reduced. In addition, our solver makes use of some techniques of *reduction* of the search space that allow us to also tackle effectively large instances. In particular, it focuses on solutions with at most two suppliers for each customer, that are selected among a short list of *preferred* facilities. Our search method has a large number of parameters, so it turned out to be necessary to tune it using an automatic tool. In particular, we employed F-Race, which resorts to statistical tests for removing inferior parameter configurations until the best one emerges.

Besides the local search method, we implemented the mathematical models of the different variants of the problem in CPLEX, so as to obtain optimal values and lower bounds for the smaller instances to be used for comparison.

Finally, we propose a novel dataset, that we call `CFLP-CI`, which could possibly become an additional benchmark for the future. The `CFLP-CI` dataset, the results for both datasets, and the source code of our method are available online at https://github.com/iolab-uniud/ms-cflp-ci.

The outcome is that our local search solver, properly tuned, is able to outperform on almost all instances all methods of Maia et al. (2023), with an average improvement of more than 6% with respect to the best one, which is MineReduce, a hybrid approach based on data mining and iterated local search. It is worse than MineReduce only on the two smallest instances, which are however less interesting, as they are solved to optimality by our implementation of the mathematical model in CPLEX.

The outline of the paper is as follows: in Section 2 we first informally describe the problem also using a toy example, then we provide the mathematical models, we introduce the datasets used and we discuss results obtained from the implementation of the models. Next, in Section 3 we present the basic ideas of our local search method, describing the search space, the initial solution strategies, the neighborhood relations and the Simulated Annealing algorithm. Section 4 presents the parameter tuning and computational results on benchmark instances from the literature and on the new dataset `CFLP-CI`. Finally, in Section 5, conclusions are drawn and future work is proposed.

2. Problem formulations, models, and dataset

In this section, we first introduce informally the problem under consideration. Secondly, we formally define it by means of mathematical models for its different versions. Finally, we discuss the features of the available dataset and the bounds obtained by the implementation of our models for this dataset, and we introduce our new dataset `CFLP-CI`.

2.1. Problem formulations

In the Capacitated Facility Location Problem (CFLP) there are n customers to be served, each one characterized by a demand d_i ($i = 1,\dots,n$) to be fully satisfied. There are m potential locations where a facility can be opened, each one with fixed cost f_j and maximum capacity s_j ($j = 1,\dots,m$). The unitary shipping cost from facility j to customer i is denoted by c_{ij}.

The problem consists in selecting the facilities to be opened and quantities supplied by each open facility to each customer, such that

```
Facilities = 3;
Costumers = 6;

Capacity = [40, 70, 60];
FixedCost = [720, 1250, 830];
Demand = [17, 8, 16, 18, 9, 11];
ShippingCost = [|39, 80, 50
                |39, 24, 88
                |50, 89, 49
                |34, 78, 62
                |75, 51, 57
                |8, 45, 52|];

Incompatibilities = 2;
IncompatiblePairs = [| 1, 5 | 4, 5 |];
```

Fig. 1. A toy instance.

the capacity of the facilities is not exceeded and the demands of the customers are fully satisfied.

This formulation corresponds to the *multi source* version of the problem (MS-CFLP). If we add the constraint that a customer must be served by only one facility, we have the *single source* problem (SS-CFLP).

These two basic versions of the CLP can be extended by the notion of *customer incompatibilities* (CI): We are given a set of pairs of customers that cannot be served by the same facility, regardless of the quantity. The resulting problems are called MS-CFLP-CI and SS-CFLP-CI, for multi-source and single-source, respectively.

A toy instance is shown in Fig. 1 in the input file format proposed by Maia et al. (2023), which is based on the language MiniZinc (Nethercote et al., 2007). The last two lines of the file describe the incompatibilities: The second to last line provides the number of pairs of incompatible customers, and the last line enumerates the pairs (starting from 1), stating that the fifth customer is incompatible with the first and the fourth customers.

The optimal solution of this instance for the MS-CFLP is shown in Fig. 2(a), where the open facilities are shown in black and the closed ones in gray. In the format proposed by Maia et al. (2023) as a set of triples (customer, facility, quantity), it is represented by the following values:

`{(1,1,3),(1,3,14),(2,1,8),(3,3,16),(4,1,18),(5,3,9),(6,1,11)}.`

The total cost is 4676 divided into 3126 supply cost ($3 \times 39 + 14 \times 50 + 8 \times 39 + 16 \times 49 + 18 \times 34 + 9 \times 57 + 11 \times 8$) and 1550 opening cost (720 + 830).

We notice that two facilities are open (1 and 3) and that customer 1 is the only one supplied by two distinct facilities, while all other customers are single-source.

If we consider the SS-CFLP formulation, the optimal solution (see Fig. 2(b)) is:

`{(1,3,17),(2,1,8),(3,3,16),(4,1,18),(5,3,9),(6,1,11)}.`

It is similar to the previous one, but the full demand of customer 1 is now taken by facility 3, with an increase of cost of $3 \times (50 - 39)$ thus resulting in a total cost of 4709.

We also notice that both solutions do not satisfy the incompatibility constraints as incompatible customers 1 and 5 are both supplied by facility 3.

The optimal solutions of the MS-CFLP-CI (see Fig. 2(c)) is

`{(1,1,17),(2,1,5),(2,3,3),(3,3,16),(4,1,18),(5,3,9),(6,3,11)},`

with a total cost of 5153. The optimal solutions of the SS-CFLP-CI (see Fig. 2(d)) is

`{(1,3,17),(2,1,8),(3,3,16),(4,3,18),(5,1,9),(6,1,11)}`

with a total cost of 5373.

With respect to the formulations without incompatibilities, in both cases the fixed costs are not changed since the same facilities are open, however the shipping has undergone a substantial change (shown in red), which corresponds to an increase of cost from 3126 to 3603 for the multi-source, and from 3150 to 3823 for the single-source.

2.2. Mathematical models

We now introduce the mathematical models of different formulations of the Capacitated Facility Location Problem. The general problem can be formulated as follows:

$$\text{(CFLP): } \min z = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} d_i + \sum_{j=1}^{m} f_j y_j \tag{1}$$

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

$$\sum_{i=1}^{n} d_i x_{ij} \le s_j y_j, \qquad j = 1, \dots, m \tag{3}$$

$$y_j \in \{0, 1\}, \qquad j = 1, \dots, m \tag{4}$$

$$x_{ij} \in \mathcal{X}, \qquad i = 1, \dots, n, j = 1, \dots, m \tag{5}$$

The objective function (Eq. (1)) minimizes both the total shipping cost of serving customers and the total fixed cost of opening facilities. Constraints (Eq. (2)) guarantee that each customer is fully served, whereas inequalities (Eq. (3)) are the capacity constraints. The decision variable y_j takes value 1 if a facility is opened in location j, and 0 otherwise, while the decision variable x_{ij} represents the fraction of the demand of customer i supplied by facility j. The domain $\mathcal{X}$ of x_{ij} variables depends on the specific formulation of the CFLP, in particular if

$$\mathcal{X} = \{0, 1\}^{n \times m} \tag{6}$$

the problem is the SS-CFLP, meaning that the entire demand of each customer must be covered by a unique facility. On the contrary, if

$$\mathcal{X} = [0, 1]^{n \times m} \tag{7}$$

then the demand of a customer can be satisfied by multiple facilities, thus the problem is the MS-CFLP. For all formulations, we assume that all input values are non negative and the capacity is strictly positive: $c_{ij} \ge 0, \forall i, j$; $f_j \ge 0$, $s_j > 0\ \forall j$; $d_i \ge 0\ \forall i$. We also assume that $\sum_{j=1}^{m} s_j \ge \sum_{i=1}^{n} d_i$, otherwise the instance would be infeasible.

In addition, it is customary to add the redundant constraints

$$x_{ij} \le y_j, \quad i = 1, \dots, n, j = 1, \dots, m \tag{8}$$

to the model (Eqs. (1)–(5)) that strengthen the continuous relaxation of CFLP models.

We now introduce the additional constraints for customer incompatibilities proposed by Maia et al. (2023). In detail, we call $\mathcal{I}$ the set of pairs of incompatible customers, such that for each $\langle i_1, i_2 \rangle \in \mathcal{I}$, i_1 and i_2 cannot be served by the same facility.

The mathematical model for the single source (SS-CFLP-CI) can be obtained by simply adding constraints (Eq. (9)) to the SS-CFLP formulation (Eqs. (1)–(6), Eq. (8)).

$$x_{i_1 j} + x_{i_2 j} \le 1, \quad \langle i_1, i_2 \rangle \in \mathcal{I}, j = 1, \dots, m \tag{9}$$

Given that x_{ij} variables are binary in the single-source case, this constraint is sufficient to ensure that the two variables involved are never both different from 0.

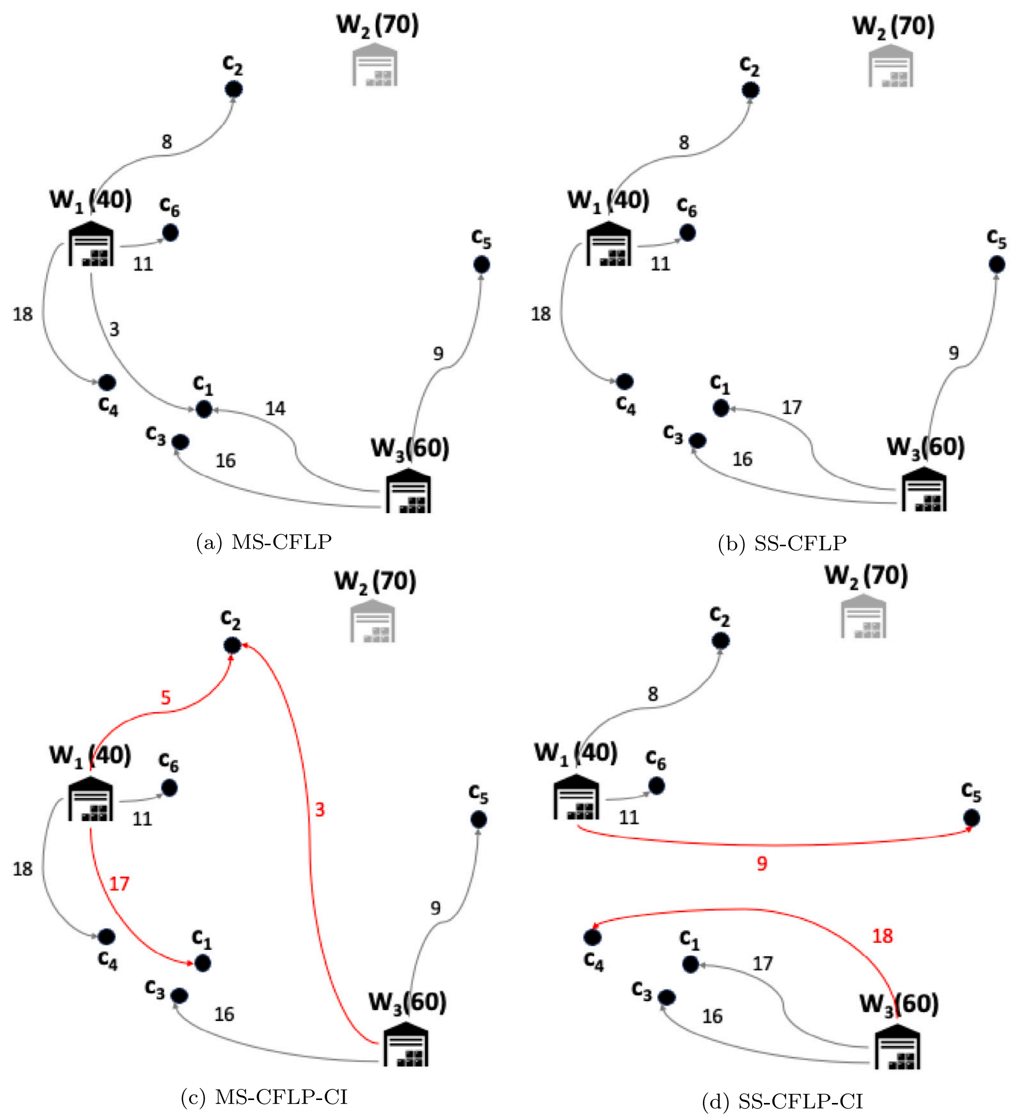


Fig. 2. The optimal solution of the toy instance for different problem formulations.

To add incompatibility constraints to the multi-source formulation, it is necessary to introduce the binary decision variables w_{ij}, so that w_{ij} is equal to 1 if the customer i is supplied by facility j (even partially), 0 otherwise. These binary variables are linked with the flow variables x_{ij} by inequalities (Eq. (10)):

$$x_{ij} \leq w_{ij}, \quad i = 1, \ldots, n, j = 1, \ldots, m. \tag{10}$$

Similarly to Eq. (9), the incompatibility constraints for the multi-source can be formulated as

$$w_{i_1 j} + w_{i_2 j} \leq 1, \quad \langle i_1, i_2 \rangle \in \mathcal{I}, j = 1, \ldots, m. \tag{11}$$

In conclusion, the complete formulation for the MS-CFLP-CI consists of Eqs. (1)–(5), Eqs. (7)–(8), and Eqs. ((10)–(11)).

2.3. Datasets and bounds

The mathematical models have been implemented in CPLEX (v. 22.1) and executed with a timeout of 7200 s on an AMD Ryzen Threadripper PRO 3975WX with 32 physical cores (3.50 GHz), hyper-threaded to 64 virtual cores, with 64 GB of memory and running Ubuntu Linux 22.4.

We tested the models on the dataset of 30 artificial instances introduced by Maia et al. (2023) and available at https://github.com/MESS-2020-1/Instances. The dataset is divided into two groups of 20 and 10 instances respectively, with the idea that the first group (wlp01–wlp20) is for training and tuning, whereas the second one (wlp21–wlp30) is meant for validation and comparison.

Table 1 shows upper and lower bounds obtained on this dataset for the MS-CFLP-CI model, along with the size of the instances in terms of facilities (m) and customers (n). Optimal solutions are underlined, non-optimal ones are followed by the lower bound (which is omitted for the optimal ones, as they coincide). For comparison, we also report the results for the multi-source formulation without incompatibilities (MS-CFLP) and single source formulation with incompatibilities (SS-CFLP-CI). The symbol — means that the execution exhausted the memory of the PC and was automatically aborted without delivering any solution. Notice that in some cases, the running time is lower than the timeout of 7200 s but the value is not optimal, because the execution was prematurely aborted; we thus reported the value of the best integer solution found so far captured from the CPLEX log.

Unsurprisingly, only relatively small instances (up to 150 facilities) were successfully solved to optimality for the MS-CFLP-CI model. For the medium-size ones (up to 450 facilities) the solver produced a feasible solution with a maximum optimality gap of 5.7%. Large instances are clearly out of reach for this model, due to memory consumption.

Table 1
Features of the training and validation instances, results of the IP solver for different problem formulations.

Instance	m	n	MS-CFLP-CI				MS-CFLP		SS-CFLP-CI	
			z	t [s]	LB	gap	z	t [s]	z	t [s]
wlp01	50	115	28716	3			27971	0	29397	4
wlp02	100	253	52952	423			51955	6	54 653	7200
wlp03	150	345	64296	6954			63077	25	66 558	7200
wlp04	200	479	84 633	7200	84 567.8	0.08%	83600	231	87 382	7200
wlp05	250	601	107 323	478	103 073.6	4.12%	102427	2268	109 135	7200
wlp06	300	705	115 295	361	110 614.2	4.23%	110 291	7200	117 426	4008
wlp07	400	1012	170 100	129	161 292.3	5.46%	160 567	3649	171 209	7200
wlp08	500	1277	–				185 942	3243	–	
wlp09	600	1483	–				216 297	801	–	
wlp10	700	1733	–				242 814	218	–	
wlp11	800	2020	–				287 213	212	–	
wlp12	900	2159	–				–		–	
wlp13	1000	2305	–				313 838	3621	–	
wlp14	1200	2927	–				–		–	
wlp15	1400	3445	–				–		–	
wlp16	1600	4067	–				–		–	
wlp17	1800	4373	–				–		–	
wlp18	2000	4908	–				–		–	
wlp19	2500	5882	–				–		–	
wlp20	3000	7800	–				–		–	
wlp21	75	172	38067	24			37560	2	39413	54
wlp22	175	428	74 473	5534	74 269.4	0.27%	73300	268	76 886	7200
wlp23	275	694	124 991	287	118 262.9	5.69%	117 619	4098	124 091	3663
wlp24	450	1128	176 721	346	167 591.1	5.45%	166 781	2249	–	
wlp25	650	1619	–				228 342	474	–	
wlp26	850	2007	–				288 183	306	–	
wlp27	1100	2847	–				–		–	
wlp28	1500	3474	–				–		–	
wlp29	1900	4522	–				–		–	
wlp30	2750	6965	–				–		–	

Looking at the results of the MS-CFLP model, we see that the incompatibility constraints play a relevant role in terms of both running time and solution quality. In fact, on the one hand the solutions without incompatibilities are obtained in shorter computational times and for larger instances, and on the other hand the quality of the solutions (for both optimal and non-optimal ones) improves up to 5.9% for the available results.

We notice that the single-source problem (SS-CFLP-CI) is more difficult than the multi-source one, due to the presence of the integrality constraints (Eq. (6)); indeed, only instances up to 75 facilities were solved to optimality. We also observe that there is an increase of the value of the objective function which, however, is quite limited. In fact, as further investigated in Section 3.1, most of the customers are served by a single facility even in the multi-source setting. Only for case wlp23 the (non-optimal) value of the single-source formulation is better than the one of the MS-CFLP-CI, but this is due to the interruption of the multi-source model by the system after only 287 s, while the single-source model was aborted after 3663 s.

Finally, we acquired the generator of Maia et al. (2023) and we tuned up some of its parameters in order to generate additional instances with different feature values. We generated a new dataset, namely CFLP-CI, composed of 50 instances, with new values for the ratio between facilities and customers, the number of incompatibilities, and the ratio between opening and supply costs. The CFLP-CI dataset is publicly available at our repository. As shown in Table 5, these instances exhibit a large number of facilities and customers, thus the optimal solution for the MS-CFLP-CI model was found only for instance cflp-ci-11 corresponding to a value of the objective function equal to 30,728. For all other instances the solver was not able to deliver any solution within the time limit.

3. Local search

In this section, we illustrate our metaheuristic search method, which is based on local search. We proceed in stages, starting from the search space (Section 3.1), then moving to the initial solution procedure (Section 3.2), then to the neighborhood relation (Section 3.3), and finally we discuss the metaheuristic that guides the search (Section 3.4).

3.1. Search space

Given that we want to address large instances, we have to deal with the fact that the straightforward choice for the search space based on the flow variables x_{ij} of Section 2 would be extremely large. Therefore, we decided to take some actions that could reduce the size of the search space, without missing the best possible solutions.

To this aim, we inspected both the optimal solutions for the small instances obtained by the mathematical model for the MS-CFLP-CI and the solutions delivered by the methods by Maia et al. (2023). We realized that in all solutions most of the customers are supplied by one single facility, less than 20% are supplied by two facilities, and only a few are supplied by three or more facilities.

In particular, for the optimal solutions of the MS-CFLP-CI model there are on average 82.2% single source customers, 16.4% customers served by two facilities, and 1.4% served by three facilities and none served by more than three. For the best solver of Maia et al. (2023), we obtained[1] the following average distribution of customers: 75.7% single source, 18.8% two sources, 4.3% three sources, 0.9% four sources, and 0.2% five or more sources. To further investigate this issue, we modified the MS-CFLP-CI model in order to have at most two suppliers for each customer by adding the following constraints

$$\sum_{j=1}^{m} w_{ij} \leq 2, \quad i = 1, \dots, n. \tag{12}$$

Optimal results for the two-source model were obtained only for small instances (up to 100 facilities), however it was interesting to

[1] We compiled and rerun the source code available online at https://github.com/MESS-2020-1/MR-MS-ILS-solver.

observe that in these cases the maximum gap between the optimal values of the multi-source and two-source models is only 0.008%, thus the loss due to this restriction is very small.

Based on these observations, we decided to limit the search space by considering only solutions with at most two suppliers per customer. With this choice, our search space is a vector Φ of size n of quadruples $\langle f_1^c, f_2^c, q_1^c, q_2^c \rangle$, such that f_1^c and f_2^c represent the two facilities that supply customer c and q_1^c and q_2^c represent their supplied quantities (with $q_1^c + q_2^c = d_c$). If a customer c is supplied entirely by one facility, then f_2^c assumes the conventional value -1 and $q_2^c = 0$. Furthermore, in order to break the symmetry, the two suppliers are ordered in terms of supply cost, so that the supply cost of f_1^c is always lower than or equal to the supply cost of f_2^c.

This choice reduces the search space very significantly, dropping its size from quadratic to linear, with respect to the number of customers ad facilities. Analogously, it allows us to keep to quadratic the size of all the neighborhoods discussed in Section 3.3, which would have been problematic with the general search space.

Nonetheless, in order to search for better results, we decided to limit and refine it further. The second action we took is based on the observation that it is very unlikely that a customer is served by a facility with a high supply cost. Therefore, we define for each customer the list of its *preferred* facilities, which is composed of its "cheapest" suppliers. The search space is then restricted to preferred assignments (i.e., assignments to preferred facilities).

The length of the list of preferred suppliers is clearly a crucial parameter of the search, as too high a value would result in a waste of time by searching for inferior assignments, and too low a value would exclude good solutions that might need one or a few "bad" assignments to be completed.

Given that we have no evidence on which would be the best choice, we decided to make it parametric and to tune it experimentally. Intuitively, the length should depend on both the size of the instance and on the distribution of the supply costs. Therefore, the length is governed by two parameters, one related to the number of facilities m and one to the supply costs.

Regarding the number of facilities, our first idea was to define the length of the list as a fraction of m. However, preliminary experiments showed that the list was too short for small instances and too long for large ones. Therefore, we decided to resort to the square root of m. In detail, the length of the list is $\beta \cdot \lfloor \sqrt{m} \rfloor$ for all customers, where β is a parameter of the search method.

The distribution of costs comes into play separately for each customer. That is, for each customer c the list is augmented by all excluded facilities that have a difference in supply cost with respect to the cheapest potential supplier of c of at most λ, which is the second parameter related to the preferred facilities.

Note that the creation of the lists depends only on the input data (and on β and λ) and can be computed once for all in a preprocessing stage, while reading the instance from the file.

Besides the two above-described choices, we made two other more straightforward limitations of the search space. First, we exclude the possibility that a facility serves two incompatible customers. This is enforced by avoiding such assignments in the initial solution and excluding the moves that would violate this rule. In order to check this situation, we maintain in the state object an integer-valued auxiliary $n \times m$ matrix Ξ that stores for each customer c the number of customers incompatible with c served by facility f.

Finally, we exclude from the space the possibility that a facility is overloaded. Similarly to incompatibilities, we provide against this occurrence in the initial solution construction and in the selection of the moves. To this aim, we store the current load of each facility in an integer-valued vector Λ of size m that is kept updated during the search. We do not explicitly store the decision to open a facility: A facility f is considered open if and only if $\Lambda_f > 0$.

Φ

	f_1	f_2	q_1	q_2
1	1	-1	17	0
2	1	3	5	3
3	3	-1	16	0
4	1	-1	18	0
5	3	-1	9	0
6	3	-1	11	0

Ξ

	1	2	3
1	0	0	1
2	0	0	0
3	0	0	0
4	0	0	1
5	2	0	0
6	0	0	0

Λ

1	2	3
40	0	39

Ψ

1	1 2 4
2	—
3	2 3 5 6

Fig. 3. Representation of solution σ_O.

With these design choices, all visited states are feasible, so that the cost function that guides the local search coincides with the objective function (Eq. (1)) of the problem, which is composed of two components: the fixed opening costs and the variable supply costs.

Besides the already-mentioned vectors Φ, Ξ and Λ, we introduced a further data structure, which is a vector Ψ of size m, that stores for each facility the list of its customers. This is useful to accelerate the computation of the difference of cost between two neighbor states (*delta costs*).

Let us call σ_O the optimal solution of the MS-CFLP-CI previously introduced in Section 2.1 and represented by the following set of triples

`{(1,1,17),(2,1,5),(2,3,3),(3,3,16),(4,1,18),(5,3,9),(6,3,11)},`

its full representation is shown in Fig. 3.

3.2. Initial solution strategies

For the initial solution, we tested two alternative strategies. The first one is *random* and works as follows. For each customer c, first we make a draw to select whether is should be served by one or two facilities (75% one, 25% two), and subsequently we select the corresponding one or two distinct preferred facilities uniformly, only checking that they are not overloaded and they do not serve incompatible customers.

In many problems the random strategy is suitable for quickly obtaining solutions which are unbiased and sufficiently diverse form each other; however, given the large size of some instances, a more heuristic strategy is presumably more promising in our context. For this reason, we resort to a *greedy* strategy, and in particular we adapted the one denoted as *Multi-start greedy algorithm* in the article by Maia et al. (2023) (Section 4.4). The method selects at each stage the pair $\langle c, f \rangle$ with the "best" cost, also considering the fixed opening cost in an amortized way, and computing the maximum quantity that can be supplied from f to c.

The adaptation consists in selecting for each customer at most two suppliers, and selecting them among the preferred ones. The second one is chosen among those that can fulfill the residual request completely. In addition, we run the procedure just once, and not in a multi-start loop.

For the toy instance, examples of solutions (called σ_R and σ_G) obtained by the random and greedy procedures, respectively, are

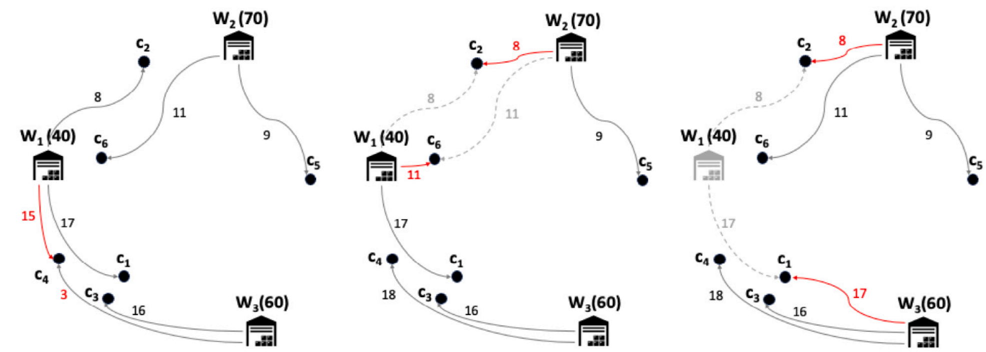
`{(1,1,17),(2,1,8),(3,3,16),(4,3,18),(5,2,9),(6,2,11)}`

with cost 6629 (3829+2800) and

`{(1,1,11),(1,3,6),(2,2,8),(3,3,16),(4,1,18),(5,2,9),(6,1,11)}`

with cost 5664 (2864+2800).

The two strategies have been compared experimentally on available benchmarks, as discussed in Section 4.1.



(a) CS⟨4, 1, 2⟩ (b) SS⟨2, 6, 1, 1⟩ (c) CF⟨1, −1⟩

Fig. 4. Examples of moves.

3.3. Neighborhoods

The neighborhood relation is indisputably the most important aspect of local search. To this regard, we considered three different atomic neighborhoods: ChangeSupplier, SwapSuppliers and ClopenFacilities.

3.3.1. ChangeSupplier neighborhood

Our first atomic neighborhood is the most straightforward one, which consists in replacing one of the suppliers of a customer. We call it ChangeSupplier(CS) and it is identified by three attributes: the customer $c \in \{1, \dots, n\}$, the new supplier $f \in \{1, \dots, m\}$, and the position $p \in \{1, 2\}$ where it is inserted in Φ. That is, the move $CS\langle c, f, p\rangle$ replaces the first ($p = 1$) or second ($p = 2$) supplier of c with f. The new supplier f is selected among the preferred ones of c.

There are three cases to be taken care of depending on the current state and the attribute p of the move.

- c has one supplier and $p = 1$: the amount of goods of c is entirely passed from the single supplier to the new one f.
- c has one supplier and $p = 2$: the amount of goods of c is split among f_1^c and f in the way that minimizes the supply cost, respecting the capacity of f (the capacity of f_1^c is always satisfied, as it initially has all the load, and so it can only decrease); the order of the two suppliers is possibly updated.
- c has two suppliers and $p = 2$: like the case above, the amount of goods of c is redistributed among f_1^c and f in order to minimize the supply cost, respecting the capacity of f and f_1^c.

The case in which c has two suppliers and $p = 1$ is not listed above because we decided to forbid it, based on the idea to focus only on promising moves. This is because, given that suppliers are ordered by supply cost, it is always more effective to replace the second supplier than the first one.

As an example, starting from the random initial solution σ_R above, the best ChangeSupplier move is CS⟨4, 1, 2⟩ that replaces the dummy facility as the second supplier of customer 4 with facility 1 (middle case above). Given that facility 1 has a lower supply cost then the current first supplier, it gets the maximum load (15, up to its capacity) and becomes the first supplier. The solution obtained (see Fig. 4(a), with changes highlighted in red) is

```
{(1,1,17),(2,1,8),(3,3,16),(4,1,15),(4,3,3),(5,2,9),(6,2,11)}
```

and the decrease of supply cost is 420 (equal to $(62 - 34) \times 15$). The opening cost remains unchanged.

3.3.2. SwapSuppliers neighborhood

The second atomic neighborhood is called SwapSuppliers (SS) and it swaps two suppliers of two distinct customers. The SS neighborhood is particularly useful in situations in which the facilities are fully loaded, and also in presence of incompatible customers, so that the state can be changed in one move only by swapping the suppliers.

It is identified by two customers $c_1, c_2 \in \{1, \dots, n\}$, and two positions $p_1, p_2 \in \{1, 2\}$. The move SS$\langle c_1, c_2, p_1, p_2\rangle$, swaps the first or second supplier (depending on p_1) of c_1 with the first or second supplier of c_2 (depending on p_2).

If c_1 (resp. c_2) has only one supplier, then p_1 (resp. p_2) is always equal to 1, as the dummy supplier cannot be swapped. The current quantities are passed to the new supplier, without any rebalancing with the other supplier, as done instead for the ChangeSupplier neighborhood.

Obviously, a move is forbidden if it overloads one of the facilities (this can happen only for the facility that receives the customer with the larger quantity) or if it creates incompatibilities.

Whenever a customer is supplied by two facilities, it is more promising to swap the second one, as they are ordered by cost. On the other hand, we do not want to exclude the possibility that the first one is also selected to be swapped. To balance this trade-off, we introduce a new parameter that we call b_{SS} (SS bias) such that with probability b_{SS} the second one is selected and with probability $1 - b_{SS}$ there is uniform random selection. In the case of customers served by a single facility, there is no option and the first (and only) one is always selected.

Starting again from σ_R, the best SwapSuppliers move is SS⟨2, 6, 1, 1⟩ that swaps the first (only) suppliers of customers 2 and 6 (which are 1 and 2, respectively). The solution obtained (see Fig. 4(b), with new flows highlighted in red and removed ones in dotted gray) is

```
{(1,1,17),(2,2,8),(3,3,16),(4,3,18),(5,2,9),(6,1,11)}
```

and the decrease of supply cost is 527 (equal to $(39 - 24) \times 8 + (45 - 8) \times 11$). The opening cost remains unchanged, as it is never affected by SwapSuppliers moves.

3.3.3. ClopenFacilities neighborhood

Our last neighborhood specifically addresses the issue of closing facilities and opening new ones. In fact, we observed in preliminary experiments that the search guided by the above two neighborhoods tends to prematurely fix the facilities to be open and rarely changes such decisions in the later part of the execution. This is because the closing of a facility would require a sequence of moves that iteratively remove clients, and is it unlikely that all members of such a sequence are iteratively selected.

We call this neighborhood ClopenFacilities (CF) (clopen: close + open) and it is identified by two attributes: the facility f_c to be closed and the facility f_o to be opened. The facility f_c is selected among those currently open, and f_o among those that are currently closed. The neighborhood also includes the option of an opening alone with no closing and vice versa. In these cases, the attribute f_c or f_o gets the conventional value -1, corresponding to the dummy facility which has no clients and no costs.

The execution of the move $\mathsf{CF}\langle f_c, f_o\rangle$ works in two steps. First, all clients of f_c are transferred one by one in a greedy way to the cheapest facility with enough space, so that f_c gets closed. In this search for the cheapest transfers from f_c, the facility f_o is assumed open, so as to encourage its use. Afterwards, all preferred clients of f_o (not assigned to it in the first step) are checked in turn to see if they can reduce their supply cost by being transferred from one or both their suppliers to f_o (up to its capacity).

It is worth mentioning that all the above-mentioned transfers (from f_c and to f_o) are not immediately executed, but rather stored in a list that complements the attributes of the move. Only when the move is accepted, all transfers are actually executed in the current state.

The execution of a ClopenFacilities move may lead to also opening other facilities besides f_o, in case some of the best transfers from f_c are towards closed facilities different from f_o. Similarly, it is possible that some other facility gets closed by the transfers towards f_o. The change of the fixed cost due to these extra openings and closings is obviously considered in the evaluation of the move.

Starting again for σ_R, the best ClopenFacilities move is $\mathsf{CF}\langle 1, -1\rangle$, that closes facility 1 and opens nothing. The best relocations of its customers (1 and 2) is towards facilities 3 and 2, respectively. The fixed cost decreases by 720 and the supply cost increases by 67, and the resulting solution (see Fig. 4(c)) is

`{(1,3,17),(2,2,8),(3,3,16),(4,3,18),(5,2,9),(6,2,11)}.`

This is actually the only feasible ClopenFacilities move in this state because no openings are possible given that all facilities are currently open. Furthermore, facility 2 cannot be closed because customer 5 cannot be relocated anywhere due to incompatibilities; and facility 3 cannot be closed because customer 4 cannot be relocated to facility 1 due to capacity limits and to facility 2 due to incompatibilities.

As for the SwapSuppliers neighborhood, the random selection is not uniform. In fact, the neighborhood is actually composed of three types of moves: open-only ones ($f_c = -1$), close-only ones ($f_o = -1$), and regular ones with both real facilities involved. In the last case, we select the pair of facilities $\langle f_c, f_o\rangle$ such that they have at least one preferred client in common, so that there is some synergy between the two sets of transfers. The selection between these three options is driven by internal probabilities. We introduce two additional parameters, called p_{Op} and p_{Cl}, such that open-only moves, close-only moves, and regular moves are selected with probability p_{Op}, p_{Cl}, and $1 - p_{\mathsf{Op}} - p_{\mathsf{Cl}}$, respectively.

It is easy to understand that the ClopenFacilities neighborhood is more complex than the previous two, in the sense that the construction and the evaluation of a ClopenFacilities move is computationally much more expensive than the others. This fact must be taken into account when selecting the rate of drawing ClopenFacilities moves with respect to the other ones, in order to design the best configuration with equal running time.

3.4. Simulated annealing

The metaheuristic that guides the local search is Simulated Annealing (SA), proposed by Kirkpatrick et al. (1983). We believe that Simulated Annealing is more promising than other local search metaheuristics, such as Tabu Search and Variable Neighborhood Search, to address problems with large instances, due to its stochastic move selection. Indeed, the methods that perform a complete exploration of the neighborhood at every move might experience a performance loss for large neighborhoods.

There are many different versions of SA, we describe here the one used in this work, and we refer to the comprehensive work by Franzin and Stützle (2019) for the description of all the other variants. Similar approaches have been successfully used by Bellio et al. (2021) and Ceschia et al. (2021) in the context of examination timetabling and frequency assignment, respectively.

The SA procedure starts from the initial solution built using one of the methods described in Section 3.2. Which of the two should be used is a parameter to be selected experimentally.

At each iteration, the SA procedure selects a random move in the composite neighborhood $\mathsf{CS} \cup \mathsf{SS} \cup \mathsf{CF}$. The move selection is done in two steps: the first step is to select one of the three atomic neighborhoods and the second one is to draw the specific move inside the neighborhood. The selection of the first step is biased based on given probabilities. That is, we have two real-valued parameters called p_{SS} and p_{CF}, such that neighborhoods SS and CF are selected with probability p_{SS}, p_{CF}. Consequently, neighborhood CS is selected with probability $1 - p_{SS} - p_{CF}$.

As customary for SA, the move is evaluated and always accepted if its cost difference Δ is negative or null (that is, the value of the objective function improves or remains the same). Conversely, if $\Delta > 0$ it is accepted according to the so-called *Metropolis* criterion, i.e., with probability $e^{-\Delta/T}$, where T is a control factor called *temperature*.

The temperature is set to its initial value T_0, and then it is decreased according to the geometric cooling scheme ($T_i = \alpha \cdot T_{i-1}$), after a fixed number of samples N_s. In order to accelerate the early stages of the search, we add the so-called *cut-off* mechanism according to which the temperature also decreases if a maximum number of moves is accepted. This threshold is expressed as a fraction ρ of the number of samples N_s (with $0 \le \rho \le 1$). The iterations "saved" by the cut-off mechanism are redeployed uniformly at the following temperature levels.

In order to guarantee that all the configurations of SA have the same running time, we use the total number of iterations I as the stop criterion. To keep I fixed, we recalculate N_s using the formula $N_s = I \bigg/ \left(\frac{\log(T_f/T_0)}{\log \alpha}\right)$, where T_f is the final temperature.

4. Experimental analysis

Our local search method has been coded in C++ and compiled with GNU g++ (v. 11.3) under Ubuntu Linux 22.4. Experiments have been run using the PC already described in Section 2.3, using one single virtual core for each experiment. The source code is available online at https://github.com/iolab-uniud/ms-cflp-ci.

4.1. Parameter tuning

The tuning procedure was carried out sampling the parameter configurations known as the *Hammersley point set* (Hammersley & Handscomb, 1964) and using of the F-Race procedure (Birattari et al., 2010) for identifying the best one. F-Race is based on the statistical tests of Friedman and Wilcoxon for removing configurations as soon as they are recognized as inferior with the given confidence.

The set of parameters is quite large and heterogeneous, therefore we preferred to divide the parameter tuning into stages, assuming that the interaction of the parameters involved in the different stages is minimal and can be neglected. In each stage, the parameters belonging to a subsequent stage were fixed to values obtained from preliminary experiments. Parameters of the preceding stages were obviously fixed to the value suggested by the previous executions of the F-Race procedure.

Table 2 summarizes the parameters, distributed in the six stages of the tuning procedure, together with their initial range and the value finally selected.

As mentioned above, the stop criterion is based on the total number of iterations and the number of samples at each temperature is computed so that the search reaches exactly the final temperature T_f.

Table 2
Parameters' configuration.

Name	Description	Tuning stage	Initial range	Value
β	square root multiplier	I	[1.0, 2.0]	1.375
λ	cost difference	I	[5, 20]	8
IS	initial solution strategy	II	{random, greedy}	greedy
T_0	initial temperature	III	[10, 50]	16.42
T_f	final temperature	III	[0.05, 0.2]	0.183
α	cooling rate	III	[0.985, 0.995]	0.994
ρ	accepted moves ratio	III	[0.05, 0.15]	0.13
p_{SS}	probability of SS moves	IV	[0.3, 0.8]	0.580
p_{CF}	probability of CF moves	IV	[0.0, 0.2]	0.044
b_{SS}	bias of SS moves	V	[0.0, 1.0]	0.45
p_{Op}	probability of Open CF moves	VI	[0.0, 0.2]	0.160
p_{Cl}	probability of Close CF moves	VI	[0.0, 0.2]	0.019

Table 3
Computational results for a timeout equal to $10\sqrt{m}$ seconds.

		MR-MS-ILS		GRASP		PcEA		MG		SA		
Inst.	LB	min	avg	min	avg	min	avg	min	avg	min	avg	gap
wlp01	*28716	28 978	**29 092.3**	30 040	30 195.4	29 006	29 736.8	34 377	34 377.0	29 025	29 249.7	0.54%
wlp02	*52952	54 493	54 882.4	56 492	56 821.7	56 925	57 789.6	59 933	60 247.3	54 061	**54 207.5**	−1.23%
wlp03	*64296	66 927	67 683.9	69 027	69 450.5	73 027	74 116.9	73 349	73 680.5	65 562	**65 986.2**	−2.51%
wlp04	84 633	89 857	90 207.1	92 455	93 042.2	98 576	100 104.5	98 367	98 615.2	85 894	**86 391.7**	−4.23%
wlp05	107 323	111 627	112 236.1	113 694	114 304.7	123 293	126 425.7	115 846	117 354.7	106 079	**106 336**	−5.26%
wlp06	115 295	118 681	119 549.1	120 797	121 942.1	133 108	137 503.5	126 265	127 688.3	113 976	**114 255.3**	−4.43%
wlp07	170 100	176 631	177 599.6	180 191	181 591.0	199 563	202 793.1	183 670	184 205.8	165 647	**166 063.1**	−6.50%
wlp08		204 862	206 574.3	212 179	214 415.6	233 202	236 736.1	211 878	212 843.8	191 822	**192 275.7**	−6.92%
wlp09		240 299	241 249.7	250 934	252 634.8	273 294	277 149.3	246 270	246 917.2	222 979	**223 537.7**	−7.34%
wlp10		264 252	265 507.4	282 518	285 182.7	308 317	314 341.1	275 649	276 605.0	249 762	**250 453.9**	−5.67%
wlp11		315 760	317 057.7	329 788	332 845.5	362 415	366 547.8	322 269	323 449.5	294 315	**294 877**	−7.00%
wlp12		323 993	326 631.9	344 734	347 702.0	374 867	379 300.5	332 749	333 909.7	302 834	**303 594.2**	−7.05%
wlp13		343 793	345 926.4	371 068	372 925.6	396 380	401 975.9	349 964	351 697.2	320 652	**321 310.4**	−7.12%
wlp14		431 981	432 921.9	468 742	470 702.7	498 599	502 115.0	436 872	438 916.3	402 477	**403 560**	−6.78%
wlp15		499 596	505 033.1	540 567	544 737.7	576 254	579 838.0	501 671	505 401.4	466 848	**467 414.7**	−7.45%
wlp16		579 364	583 798.1	615 768	619 139.1	658 583	669 751.4	581 757	583 453.1	541 385	**542 326.1**	−7.10%
wlp17		605 310	606 812.2	655 090	662 262.5	702 080	710 160.3	617 832	619 390.0	573 244	**574 713**	−5.29%
wlp18		677 396	680 067.6	736 207	740 016.9	774 778	783 993.2	687 117	688 304.8	638 976	**640 471.3**	−5.82%
wlp19		807 447	815 478.9	876 105	880 930.1	937 785	946 733.5	807 573	809 986.9	757 538	**758 421.8**	−7.00%
wlp20		1 043 150	1048531.0	1 140 090	1142525.0	1 192 820	1198344.0	1 050 020	1053036.0	994 310	**995 029.6**	−5.10%
wlp21	*38067	38 474	**38 653.0**	39 892	40 282.8	39 233	40 466.1	44 298	44 443.7	38 872	39 147.1	1.28%
wlp22	74 473	79 378	79 746.2	79 348	80 078.4	84 301	86 434.6	85 732	86 462.4	75 860	**76 043.7**	−4.64%
wlp23	124 991	127 873	128 307.5	129 719	130 669.6	143 077	144 204.9	134 271	135 725.6	121 275	**121 540.2**	−5.27%
wlp24	176 721	182 248	182 922.0	191 882	193 015.9	209 596	212 898.2	191 017	192 262.3	172 252	**172 672.6**	−5.60%
wlp25		247 975	251 053.7	264 454	267 853.5	289 981	292 921.6	257 904	258 974.2	234 307	**235 019.8**	−6.39%
wlp26		322 663	324 775.3	334 346	336 539.9	364 831	369 576.6	322 085	323 866.0	295 774	**296 392.6**	−8.74%
wlp27		421 190	422 517.2	457 577	460 811.3	481 565	488 519.6	430 559	431 642.5	397 231	**397 998**	−5.80%
wlp28		493 849	497 937.1	536 825	543 859.7	566 881	573 423.6	502 019	504 837.7	465 282	**466 056**	−6.40%
wlp29		658 764	665 442.3	689 560	697 430.5	743 620	754 246.0	648 531	650 840.2	603 826	**605 266.3**	−9.04%
wlp30		943 009	949 442.8	1 014 500	1017282.0	1 072 730	1076453.0	940 479	942 920.2	888 124	**888 870.3**	−6.38%
Avg		349 994.0	352 254.6	374 153.0	376 706.4	399 956.2	404 486.7	355 677.4	357 068.5	329 006.3	329 649.4	−6.42%

This choice guarantees that the running time is approximately the same for all settings of the parameters. The exception to this setting is Stage IV, given that the rates p_* of the neighborhoods have a great influence on the running times (in particular for the CF neighborhood). As a consequence, a fair comparison of configurations for Stage IV cannot be done based on the same number of iterations. Therefore, for Stage IV alone, we use a time-based stop criterion, so that cooling (i.e., multiplying by α) is applied when the time assigned to each temperature is expired, regardless of the number of iterations.

4.2. Comparison results

In this section, we position the results of our SA method in the best configuration found by the tuning procedure and reported in Table 2 within the ones in the literature. In detail, Tables 3 and 4 compare the average and best results out of 10 runs to those obtained by Maia et al. (2023) for the two timeouts proposed by Maia et al., which depend on the size of the instance and correspond to m and $10\sqrt{m}$ seconds, respectively.

The time-based stop criterion of SA has been used for this comparison. Given that the PC used by Maia et al. is basically equivalent to ours, we granted the same running time with no rescaling.

The column LB gives lower bound (marked with $*$ if it corresponds to the optimum) obtained by running the MS-CFLP-CI mathematical model described in Section 2.3.

The approaches developed by Maia et al. (2023) are: MineReduce-based Multi-Start Iterated Local Search (MR-MS-ILS), Greedy Randomized Adaptive Search Procedure (GRASP), Permutation-Coded Evolutionary Algorithm (PcEA), and Multi-start Greedy (MG). We do not

Table 4
Computational results for a timeout equal to m seconds.

Inst.	LB	MR-MS-ILS min	MR-MS-ILS avg	GRASP min	GRASP avg	PcEA min	PcEA avg	MG min	MG avg	SA min	SA avg	gap
wlp01	*28716	28 913	**29 115.8**	30 041	30 149.5	29 455	29 754	34 377	34 377	29 002	29 122.3	0.02%
wlp02	*52952	54 337	54 802.9	56 043	56 725.3	56 703	57 847.9	60 055	60 347.9	53 838	**54 117.5**	−1.25%
wlp03	*64296	67 266	67 540.8	68 609	69 476.1	70 796	72 662.8	73 064	73 677.8	65 570	**65 804.1**	−2.57%
wlp04	84 633	89 544	90 060.3	92 301	92 797.8	97 317	99 675.1	97 624	98 247.5	85 933	**86 219.4**	−4.26%
wlp05	107 323	111 640	112 175.6	112 795	113 542.8	120 270	123 091.1	116 841	117 319.3	105 814	**106 185.8**	−5.34%
wlp06	115 295	119 049	119 613.1	120 892	121 759.5	134 270	135 687.4	126 802	127 334.7	113 499	**113 934.0**	−4.75%
wlp07	170 100	176 044	177 044.3	178 863	180 693.6	195 751	198 910.9	183 445	183 797.8	165 105	**165 623.4**	−6.45%
wlp08		203 358	205 710.1	212 670	213 795.5	227 613	232 665.3	211 885	212 515.1	191 002	**191 537.5**	−6.89%
wlp09		239 174	240 312.5	248 099	251 159.7	271 341	273 808.6	245 345	245 922.1	222 498	**222 838.3**	−7.27%
wlp10		263 295	264 925	281 576	283 672.7	301 321	309 782.7	274 494	275 508.5	249 199	**249 629.1**	−5.77%
wlp11		313 229	315 728.7	327 830	331 875.2	355 126	360 683.7	321 833	322 804.4	293 349	**294 125.8**	−6.84%
wlp12		324 766	325 507.1	341 030	344 437	369 333	375 447.1	332 666	333 413.6	301 602	**302 619.9**	−7.03%
wlp13		341 823	345 307.9	366 602	368 888.9	390 328	396 320.3	349 548	350 811.3	319 647	**320 238.1**	−7.26%
wlp14		429 168	430 919.1	462 335	468 817.1	491 340	496 191.2	436 583	438 183.4	400 871	**401 929.2**	−6.73%
wlp15		497 061	501 122.4	538 535	541 858.4	570 659	575 833.8	502 617	504 380.3	464 710	**465 208.8**	−7.17%
wlp16		578 979	581 756.9	613 451	617 507.9	658 618	664 376	580 481	581 807.7	539 320	**540 173.0**	−7.15%
wlp17		603 288	604 889	654 713	660 503.1	697 541	704 478.4	616 041	617 354.1	571 361	**571 954.6**	−5.44%
wlp18		674 421	677 531.8	734 574	739 435.7	778 266	782 867.4	683 724	686 295.7	636 129	**637 451.0**	−5.92%
wlp19		805 663	808 405.2	873 218	878 551.5	920 158	941 345.1	*805365*	807 075.4	754 102	**754 760.4**	−6.64%
wlp20		1 039 400	1 041 971	1 137 770	1 142 545	1 173 020	1 197 136	1 044 990	1 047 811	986 397	**987 448.3**	−5.23%
wlp21	*38067	38 420	**38 567.3**	39 978	40 214.7	39 781	40 600.8	44 175	44 391.6	38 920	39 124.3	1.44%
wlp22	74 473	78 813	79 336.6	79 599	80 138.5	83 092	84 975.6	85 964	86 273.8	75 888	**76 088.7**	−4.09%
wlp23	124 991	127 076	128 028.2	129 527	130 163.8	139 348	142 287.9	133 931	135 290.9	121 250	**121 468.4**	−5.12%
wlp24	176 721	181 199	182 220.6	189 718	191 675	207 508	209 439	190 960	192 024.5	171 887	**172 168.9**	−5.52%
wlp25		249 547	250 928.1	265 817	267 378.8	283 119	288 489.6	257 796	258 596.2	234 000	**234 580.6**	−6.51%
wlp26		321 935	323 134	331 780	333 906.5	358 093	365 582.5	322 082	323 315.8	294 942	**295 352.6**	−8.60%
wlp27		418 930	420 459.1	451 413	455 969.9	477 195	481 836.7	428 214	430 329	395 901	**396 510.9**	−5.70%
wlp28		493 746	495 644.8	535 637	540 882.2	568 640	573 048.9	502 483	503 812.5	463 263	**464 045.4**	−6.38%
wlp29		656 512	659 751.1	694 579	697 376.3	734 162	744 444.2	*646626*	648 973.2	602 169	**602 607.9**	−8.66%
wlp30		940 179	942 409.2	1 012 550	1 016 481	1 066 990	1 072 834	*936728*	939 816.9	882 060	**882 912.5**	−6.31%
Avg		348 892.5	350 497.3	372 751.5	375 412.6	395 571.8	401 070.1	354 891.3	356 060.3	327 640.9	328 192.7	−6.36%

consider the results by Pandey et al. (2023) as they regard only instances wlp01–wlp15 and they are consistently worse than the ones by Maia et al.

We see that SA (whose solutions were validated with the solution checker provided by Maia et al.) outperforms all four previous methods on all instances except for the smallest two (wlp01 and wlp21) for both timeout settings, in terms of average and best solution values.

Considering the instances whose proven optimal solution is known, the optimality gap is at most 2.2% using the linear timeout. The average improvement with respect to MR-MS-ILS, which was the leading algorithm developed in Maia et al. (2023), is respectively 6.36% and 6.42% for linear and $10\sqrt{m}$ timeouts, and it is more substantial for large instances.

The fact that we do not improve on the smallest instances can be explained considering the fact that our technique was specifically tailored to deal with large instances. An ad hoc setting of the parameters for small instances could have improved the results on them. However, we consider them less interesting given that they are solved to optimality by the mathematical model.

4.3. Runs with shorter timeouts

In order to further investigate the effectiveness of SA in different computational settings, we performed an analysis of its performance on three selected instances (wlp05, wlp08 and wlp13) representative of the different sizes of the dataset, with shorter running times. In detail, we considered a timeout equal to $k\sqrt{m}$ with $k \in [1, 10]$ and we computed the average value of the objective function for 10 repetitions for each instance.

Fig. 5 plots the average percentage increase of the objective function with respect to the value obtained with $k = 10$, for different values of the parameter k, corresponding to a growing timeout from $\sqrt{m}$ to $10\sqrt{m}$.

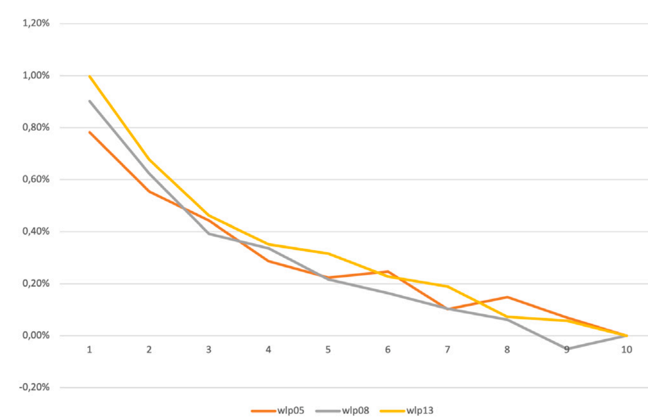


Fig. 5. Average percentage deviation as a function of k.

The outcome is that reducing the running time by a factor of ten makes the results deteriorate by at most 1%. In addition, we see that, unsurprisingly, the decrease of performances happens smoothly with the running time, demonstrating that SA obtains good quality results for short runs as well.

4.4. Results on the CFLP−CI dataset

Finally, we report the results for the new CFLP−CI dataset. Table 5 shows the number of facilities (m) and customers (n) of each instance, along with the results of our solver for both timeouts, on the configuration reported in Table 2. This dataset and these results could be used for future comparisons with new search methods.

Table 5
Computational results on the CFLP-CI dataset for different timeouts.

Instance	m	n	SA ($10\sqrt{m}$ timeout)			SA (m timeout)		
			min	avg	avg time	min	avg	avg time
cflp-ci-00	852	2102	424 519	425 264.1	290.7	423 418	423 997.0	851.0
cflp-ci-01	1981	5709	573 983	575 397.2	391.9	568 919	570 251.9	1916.9
cflp-ci-02	2224	5970	448 326	449 302.0	387.2	442 916	443 314.1	2136.3
cflp-ci-03	2826	7701	1 508 473	1509804.8	369.0	1 495 309	1496823.6	2669.5
cflp-ci-04	2143	6143	1 220 618	1222512.9	391.5	1 216 587	1217446.0	2080.2
cflp-ci-05	1788	4829	368 456	368 914.3	385.1	364 036	364 548.5	1750.5
cflp-ci-06	633	1705	155 149	155 376.4	250.9	154 811	155 013.3	632.2
cflp-ci-07	2353	5319	452 561	453 707.1	414.3	447 859	448 501.9	2287.3
cflp-ci-08	1479	4396	702 000	703 657.0	357.5	699 317	700 112.5	1450.1
cflp-ci-09	2161	4510	334 128	334 818.7	432.8	331 089	331 702.6	2130.2
cflp-ci-10	1583	4609	612 854	614 279.3	370.1	610 333	611 297.3	1553.7
cflp-ci-11	69	156	30 894	30 975.9	82.8	30 882	31 059.8	68.6
cflp-ci-12	618	1832	461 171	462 177.8	247.8	460 229	461 474.0	617.4
cflp-ci-13	684	1558	388 264	388 879.6	260.8	386 948	388 202.4	683.5
cflp-ci-14	2128	5893	1 031 054	1032521.9	403.6	1 027 028	1028186.6	2070.6
cflp-ci-15	2226	5225	565 607	566 459.6	413.9	561 099	562 128.0	2169.4
cflp-ci-16	1485	3606	891 160	892 011.8	370.7	887 877	889 296.2	1471.2
cflp-ci-17	2721	5840	1 231 972	1233547.8	426.3	1 225 297	1226184.4	2628.8
cflp-ci-18	1773	4757	903 653	905 310.9	385.9	900 438	902 206.0	1738.6
cflp-ci-19	520	1305	152 349	152 740.9	227.4	151 907	152 255.5	519.7
cflp-ci-20	2629	6587	1 280 618	1283255.5	412.6	1 273 308	1275338.5	2524.2
cflp-ci-21	1121	2374	460 143	461 168.5	331.8	459 288	459 903.4	1118.1
cflp-ci-22	722	2138	313 629	314 280.3	267.6	313 381	313 663.0	721.2
cflp-ci-23	1471	3344	414 336	415 852.6	371.8	413 761	414 130.3	1457.9
cflp-ci-24	2142	5518	473 660	474 247.5	411.5	468 618	469 362.6	2092.0
cflp-ci-25	2757	7474	803 729	805 523.5	374.7	796 370	797 221.8	2587.1
cflp-ci-26	2442	5073	698 430	700 168.0	438.4	695 323	696 126.2	2387.2
cflp-ci-27	2866	6689	1 433 262	1434453.2	416.3	1 420 856	1423304.2	2747.6
cflp-ci-28	855	2141	469 428	470 734.6	291.1	469 035	469 843.7	854.0
cflp-ci-29	2787	6825	1 381 347	1383402.3	423.9	1 372 452	1374401.2	2681.0
cflp-ci-30	1748	4095	580 934	581 995.5	394.9	579 263	580 050.7	1724.7
cflp-ci-31	2946	7622	1 216 734	1218218.8	400.2	1 206 339	1207307.3	2804.2
cflp-ci-32	2028	4946	492 232	493 294.5	403.6	487 236	488 433.6	1982.2
cflp-ci-33	2685	5974	1 082 543	1083618.0	428.2	1 075 666	1076818.6	2605.9
cflp-ci-34	2048	5004	1 156 148	1158156.4	407.1	1 151 861	1153176.1	2004.2
cflp-ci-35	2618	6408	549 152	550 971.0	425.9	541 626	542 213.1	2537.7
cflp-ci-36	1276	3697	556 058	557 097.0	343.7	554 259	555 083.6	1262.0
cflp-ci-37	2213	5823	1 290 114	1291258.3	411.6	1 284 102	1285221.9	2156.2
cflp-ci-38	1745	5050	1 058 405	1059980.9	381.4	1 054 933	1055679.1	1710.6
cflp-ci-39	383	1131	192 846	193 320.9	195.3	192 693	193 001.8	382.5
cflp-ci-40	1731	3466	638 996	639 737.2	397.0	636 674	637 300.3	1712.4
cflp-ci-41	2704	7893	1 309 439	1311166.0	397.1	1 297 401	1298835.6	2584.7
cflp-ci-42	937	2133	422 885	424 160.1	304.6	422 267	422 889.1	935.6
cflp-ci-43	1770	4181	508 871	509 989.0	385.6	505 548	506 465.7	1736.3
cflp-ci-44	2074	5752	766 352	767 928.0	383.0	759 920	761 332.5	2005.6
cflp-ci-45	2288	4585	724 761	725 291.0	434.2	721 010	721 857.7	2240.9
cflp-ci-46	2431	5397	637 155	638 433.8	424.8	633 722	634 707.6	2358.5
cflp-ci-47	843	2090	410 913	411 767.5	289.2	410 205	410 729.5	841.7
cflp-ci-48	1223	3442	778 044	779 471.1	339.9	775 605	777 095.8	1213.1
cflp-ci-49	892	2135	305 726	306 143.6	297.3	304 209	304 719.1	890.4
Avg			697 281.6	698 454.9	360.9	693 264.6	694 204.3	1725.7

5. Conclusions and future work

We have designed a local search approach for the MS-CFLP-CI problem, recently proposed by Maia et al. (2023). Our method has been able to outperform all previous methods on almost all instances independently of the running time. Additional experiments show that our approach is also suitable for shorter runs, allowing for more interactive solution sessions. We believe that the key ingredients of the method are the reduced (two-suppliers) search space and the sharp tailoring and engineering of the neighborhoods.

In addition, we have provided a mathematical model that obtains optimal solutions for relatively small instances. In addition, our model has been helpful, through the inspection of its solutions, to guide our successful choice of using the reduced search space.

Finally, we have designed and made publicly available a new dataset that could complement the current one as benchmark for comparisons on this problem.

For the future, we plan to refine the neighborhoods so as to try to further improve our current results. For example, the dynamic rebalancing of the quantities of the two suppliers applied in the ChangeSupplier neighborhood could be profitably applied to SwapSuppliers and CloseOpenFacilities as well.

In addition, we plan to adapt our solution technique to different versions of the CFLP and compare our results with the corresponding state-of-the-art contributions on the available benchmarks.

Finally, we would like to design some "auto-tuning" mechanism that could control some of the parameters online based on reinforcement learning techniques, in order to reduce the computational cost of the tuning procedure.

CRediT authorship contribution statement

Sara Ceschia: Conceptualization, Data curation, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Andrea Schaerf:** Conceptualization, Investigation, Methodology, Software, Supervision, Validation, Writing – original draft, Writing – review & editing.

Data availability

All data and source code is available at https://github.com/iolab-uniud/ms-cflp-ci.

Acknowledgments

We thank the organizers of the MESS-2020+1 summer school, Mario Pavone in particular, and the participants to the MESS-2020+1 competition.

References

Ahuja, R. K., Orlin, J. B., Pallottino, S., Scaparra, M. P., & Scutellà, M. G. (2004). A multi-exchange heuristic for the single-source capacitated facility location problem. *Management Science*, *50*(6), 749–760.

Avella, P., & Boccia, M. (2009). A cutting plane algorithm for the capacitated facility location problem. *Computational Optimization and Applications*, *43*(1), 39–65.

Avella, P., Boccia, M., Mattia, S., & Rossi, F. (2021). Weak flow cover inequalities for the capacitated facility location problem. *European Journal of Operational Research*, *289*(2), 485–494.

Avella, P., Boccia, M., Sforza, A., & Vasil'ev, I. (2009). An effective heuristic for large-scale capacitated facility location problems. *Journal of Heuristics*, *15*(6), 597.

Basu, S., Sharma, M., & Ghosh, P. S. (2015). Metaheuristic applications on discrete facility location problems: a survey. *Opsearch*, *52*, 530–561.

Bellio, R., Ceschia, S., Di Gaspero, L., & Schaerf, A. (2021). Two-stage multi-neighborhood simulated annealing for uncapacitated examination timetabling. *Computers & Operations Research*, *132*, 105300. http://dx.doi.org/10.1016/j.cor.2021.105300, https://authors.elsevier.com/c/1cyGn15N8SFYRV.

Birattari, M., Yuan, Z., Balaprakash, P., & Stützle, T. (2010). F-race and iterated F-race: An overview. In *Experimental methods for the analysis of optimization algorithms* (pp. 311–336). Berlin: Springer.

Caserta, M., & Voß, S. (2020). A general corridor method-based approach for capacitated facility location. *International Journal of Production Research*, *58*(13), 3855–3880.

Celik Turkoglu, D., & Erol Genevois, M. (2020). A comparative survey of service facility location problems. *Annals of Operations Research*, *292*, 399–468.

Ceschia, S., Di Gaspero, L., Rosati, R. M., & Schaerf, A. (2021). Multi-neighborhood simulated annealing for the minimum interference frequency assignment problem. *EURO Journal on Computational Optimization*, 1–32. http://dx.doi.org/10.1016/j.ejco.2021.100024, https://www.sciencedirect.com/science/article/pii/S2192440621001519.

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.

Chudak, F. A., & Williamson, D. P. (2005). Improved approximation algorithms for capacitated facility location problems. *Mathematical Programming*, *102*, 207–222.

Cornuéjols, G., Nemhauser, G., & Wolsey, L. (1983). *The uncapicitated facility location problem*: *Tech. rep*, Cornell University Operations Research and Industrial Engineering.

Cortinhal, M. J., & Captivo, M. E. (2003). Upper and lower bounds for the single source capacitated location problem. *European Journal of Operational Research*, *151*(2), 333–351.

Drezner, Z., & Hamacher, H. W. (2004). *Facility location: Applications and theory*. Springer Science & Business Media.

Fernández, E., & Landete, M. (2015). Fixed-charge facility location problems. In *Location science* (pp. 47–77). Springer.

Fischetti, M., Ljubić, I., & Sinnl, M. (2016). Benders decomposition without separability: A computational study for capacitated facility location problems. *European Journal of Operational Research*, *253*(3), 557–569.

Franzin, A., & Stützle, T. (2019). Revisiting simulated annealing: A component-based analysis. *Computers & Operations Research*, *104*, 191–206.

Goossens, D., & Spieksma, F. C. (2009). The transportation problem with exclusionary side constraints. *4OR*, *7*, 51–60.

Görtz, S., & Klose, A. (2012). A simple but usually fast branch-and-bound algorithm for the capacitated facility location problem. *INFORMS Journal on Computing*, *24*(4), 597–610.

Guastaroba, G., & Speranza, M. G. (2012). Kernel search for the capacitated facility location problem. *Journal of Heuristics*, *18*(6), 877–917.

Hammersley, J. M., & Handscomb, D. C. (1964). *Monte Carlo methods*. London: Chapman and Hall.

Kirkpatrick, S., Gelatt, D., & Vecchi, M. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Klose, A., & Drexl, A. (2005). Facility location models for distribution system design. *European Journal of Operational Research*, *162*(1), 4–29.

Korupolu, M. R., Plaxton, C. G., & Rajaraman, R. (2000). Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, *37*(1), 146–188.

Krarup, J., & Pruzan, P. M. (1983). The simple plant location problem: Survey and synthesis. *European Journal of Operational Research*, *12*(36–81), 41.

Lai, M. C., suk Sohn, H., Tseng, T. L. B., & Chiang, C. (2010). A hybrid algorithm for capacitated plant location problem. *Expert Systems with Applications*, *37*(12), 8599–8605.

Laporte, G., Nickel, S., & da Gama, F. S. (2015). *Location science*. Springer Nature.

Letchford, A. N., & Miller, S. J. (2014). An aggressive reduction scheme for the simple plant location problem. *European Journal of Operational Research*, *234*(3), 674–682.

Maia, M. R., Reula, M., Parreño-Torres, C., Vuppuluri, P. P., Plastino, A., Souza, U. S., Ceschia, S., Pavone, M., & Schaerf, A. (2023). Metaheuristic techniques for the capacitated facility location problem with customer incompatibilities. *Soft Computing*, *27*(8), 4685–4698.

Marín, A., & Pelegrín, M. (2019). Adding incompatibilities to the Simple Plant Location Problem: Formulation, facets and computational experience. *Computers & Operations Research*, *104*, 174–190.

Nethercote, N., Stuckey, P. J., Becket, R., Brand, S., Duck, G. J., & Tack, G. (2007). MiniZinc: Towards a standard CP modelling language. In C. Bessière (Ed.), *LNCS*: *vol. 4741*, *CP 2007* (pp. 529–543). Springer.

Pandey, A., Taneja, N., & Vuppuluri, P. P. (2023). Heuristic strategies for warehouse location with store incompatibilities in supply chains. In *Applied intelligence in human-computer interaction* (pp. 185–201). CRC Press.

Rabbani, M., Heidari, R., Farrokhi-Asl, H., & Rahimi, N. (2018). Using metaheuristic algorithms to solve a multi-objective industrial hazardous waste location-routing problem considering incompatible waste types. *Journal of Cleaner Production*, *170*, 227–241.

Revelle, C. S., & Laporte, G. (1996). The plant location problem: new models and research prospects. *Operations Research*, *44*(6), 864–874.

Swamy, C., & Shmoys, D. B. (2008). Fault-tolerant facility location. *ACM Transactions on Algorithms (TALG)*, *4*(4), 1–27.

Verter, V. (2011). *International series in operations research & management science*: *vol. 155*, *Foundations of location analysis* (pp. 25–37). Springer.

Weninger, D., & Wolsey, L. A. (2023). Benders-type branch-and-cut algorithms for capacitated facility location with single-sourcing. *European Journal of Operational Research*, *310*(1), 84–99.