



A new approach to the multiple obnoxious facility location problem based on combinatorial and continuous tools

M. Locatelli¹

Received: 20 November 2023 / Accepted: 25 January 2024 / Published online: 3 March 2024
© The Author(s) 2024

Abstract

In this paper we address the multiple obnoxious facility location problem. In this problem p facilities need to be spread within the unit square in such a way that they are far enough from each other and that their minimal distance from n communities, with known positions within the unit square, is maximized. The problem has a combinatorial component, related to the key observation made in Drezner (Omega 87:105–116, 2019) about the role played by Voronoi points. We propose a new approach, which exploits both the combinatorial component of the problem and, through continuous local optimizations, also its continuous component. We also propose techniques to limit the impact on computation times of the number n of communities. The approach turns out to be quite competitive and is able to return 24 new best known solutions with respect to the best results reported in Kalczyński (Optim Lett 16:1153–1166, 2022).

Keywords Multiple obnoxious facility location problem · Voronoi points · Clique problems · Continuous local optimization

1 Introduction

Facility location problems (FLP) have been studied for a long time. It is difficult to state exactly which is the first paper discussing FLPs, but one of the first is certainly Balinski's paper [4] on Integer Programming, where the FLP was included among the applications of Integer Programming. In FLPs the location of some 'facilities' within an area also containing different 'communities' is to be decided in such a way that some objective function, depending both on the facilities and on the communities, is minimized or maximized. Different variants of FLPs can be found in the literature. For instance, a distinction is made between uncapacitated FLPs, where facilities are assumed to have an infinite capacity (see, e.g., [14]), and capacitated FLPs, where

✉ M. Locatelli
marco.locatelli@unipr.it

¹ Dipartimento di Ingegneria e Architettura, University of Parma, Parma, Italy

facilities are assumed to have a finite capacity (see, e.g., [1]). FLP models are more often deterministic, but also stochastic models have been explored (see, e.g., [13]). Complexity results have been derived for different sub-classes (see, e.g., [19]). Some papers combine FLPs with routing problems (see, e.g., [5]). Some applications of FLPs are so relevant that they have an independent literature, see, e.g., the survey [2] about healthcare FLPs. For a more detailed discussion about these and also other issues related to FLPs, we refer to the quite detailed survey [7].

Many papers about FLP discuss discrete FLPs, where facilities can be placed within a discrete set of positions. But some also consider continuous FLPs, where facilities can be placed within a continuous region (see, e.g., [18]). The problem tackled in this paper belongs to the class of continuous FLPs. It is worthwhile to remark that FLPs in a continuous space are strictly related to circle packing problems, where one aims at placing a fixed number of circles within a given region in such a way that their common radius is maximized, or, alternatively, one aims at placing the largest possible number of circles of fixed radius within a given region (see, e.g., [6, 15–17]). In circle packing problems, the centers of the circles play the same role as the facilities. However, the presence of communities introduces an important difference of FLPs with respect to circle packing problems. While the latter are purely continuous problems, communities placed at fixed positions introduce a combinatorial element in continuous FLPs which, as we will see, can be effectively exploited to solve them.

The paper is structured as follows. In Sect. 2, we introduce the FLP discussed in this paper, namely the multiple obnoxious FLP, and we briefly revise the recent literature about it. In Sect. 3, we present a new approach to solve this problem which combines its combinatorial and its continuous aspects. In Sect. 4, we present different computational results on the test instances available in the literature, and we discuss such results. In Sect. 5, we draw some conclusions and propose some topics for future research.

2 The multiple obnoxious facility location problem

Let $N = \{1, \dots, n\}$ be the index set of a collection of communities, and let $\mathcal{A} = \{(a_i, b_i) \in [0, 1]^2 \mid i \in N\}$ be the corresponding collection of n fixed locations, within the unit square. In the multiple obnoxious facility location problem we aim at placing p facilities at positions $(x_j, y_j) \in [0, 1]^2$, $j \in P = \{1, \dots, p\}$, in such a way that their reciprocal distance is above a given threshold D , and so that their minimal distance from the n communities is maximized. The most straightforward mathematical model for this problem is the following:

$$\begin{aligned}
 & \max L \\
 & (x_k - x_j)^2 + (y_k - y_j)^2 \geq D^2 \quad k, j \in P, k \neq j \\
 & (x_j - a_i)^2 + (y_j - b_i)^2 \geq L^2 \quad i \in N, j \in P \\
 & L \geq 0 \\
 & (x_j, y_j) \in [0, 1]^2 \quad j \in P.
 \end{aligned} \tag{1}$$

In what follows, we refer to the first set of constraints as the set of facility-facility distance (FFD) constraints, while we refer to the second set of constraints as the set of facility-community distance (FCD) constraints. For a given point $(x, y) \in [0, 1]^2$ we denote by $C(x, y)$ the index set of the points in \mathcal{A} closest to (x, y) , i.e.,

$$C(x, y) = \left\{ h \in N \mid (x - a_h)^2 + (y - b_h)^2 = \min_{i \in N} (x - a_i)^2 + (y - b_i)^2 \right\}.$$

We denote by $\mathcal{H} = \left\{ (z_x^r, z_y^r) \mid r = 1, \dots, R \right\}$ the set of so called *Voronoi points* (see, e.g., [3]), i.e.:

- points (x, y) in the interior of the unit square such that $|C(x, y)| \geq 3$;
- points (x, y) belonging to edges of the unit square such that $|C(x, y)| \geq 2$;
- the four vertices of the unit square.

In case we would like to maximize the minimum distance from points in \mathcal{A} of a single facility to be placed in the unit square, we should restrict our attention to positions in the finite set \mathcal{H} . But this set might be useful also if we would like to place $p > 1$ facilities in the unit square. In Drezner et al. [10] the following approach is proposed. First, a graph $G = (V, E)$ is introduced with $V = \{1, \dots, R\}$ and

$$E = \left\{ (r, q) \mid r, q \in V \text{ and } (z_x^r - z_x^q)^2 + (z_y^r - z_y^q)^2 \geq D^2 \right\}.$$

Thus, the vertex set of the graph corresponds to the set of points \mathcal{H} , while an edge joins two vertices if the distance between the corresponding points is not lower than the threshold distance D . Next, to each vertex $r \in V$ we associate a value

$$F_r = \min_{i \in N} \sqrt{(z_x^r - a_i)^2 + (z_y^r - b_i)^2},$$

i.e., the value is the minimal distance of the corresponding point in \mathcal{H} from points in \mathcal{A} . Finally, let \mathcal{C} be the collection of cliques in G with cardinality p and let $C \in \mathcal{C}$ be a clique with largest minimal value, i.e.,

$$C \in \arg \max_{C' \in \mathcal{C}} \min_{i \in C'} F_i. \tag{2}$$

Then, the set of p points

$$H_C = \left\{ (z_x^r, z_y^r) \mid r \in C \right\}, \tag{3}$$

is a feasible solution for (1). In Kalczyński et al. [12] three distinct techniques to improve such solution are proposed:

- In the first technique the node in C with lowest value, say node q , is discarded and then a method is run which replaces point (z_x^q, z_y^q) with a point in the unit

square which maximizes the minimal distance from points in \mathcal{A} and lies outside all circles with radius D and center in (z_x^r, z_y^r) , $r \in C \setminus \{q\}$.

- In the second technique, again the node in C with lowest value is discarded and then a new set of p points $\{(z_x^r, z_y^r) \mid r \in C' \subset V, |C'| = p\}$, maximizing the minimal distance from points in \mathcal{A} , is searched for in such a way that the value $D_0 \leq D$, for which this set of points is feasible for (1) when D_0 replaces D , is maximized. Next, an attempt is made to refine the detected solution, not feasible for (1) if $D_0 < D$, into a feasible solution for (1) by the Sequential Linear Programming (SLP) approach, first introduced in [8] and shown to work well for class of problems including (1) in [9].
- The third improvement is similar to the second one, but in selecting the p points it takes into account that a violation of the distance threshold D by some pair $i, j \in V$ is more easily removed if the values F_i and F_j are large. Indeed, if value F_i is higher than the value F_q of the discarded node, small perturbations of point (z_x^i, z_y^i) do not affect the minimal distance from points in \mathcal{A} or, equivalently, the smallest value among those of the p selected points.

We remark that an equivalent way to view the second improvement proposed in [12] is the following. Let us define a graph $G_t = (V, E_t)$, where $t \leq 1$, and

$$E_t = \left\{ (r, q) \mid r, q \in V \text{ and } (z_x^r - z_x^q)^2 + (z_y^r - z_y^q)^2 \geq t^2 D^2 \right\}.$$

Note that $E_1 \equiv E$, and that $E_{t_1} \subseteq E_{t_2}$ for $t_1 \geq t_2$. In the second improvement technique proposed in [12], we search for the largest $t \leq 1$ such that a clique $C_t \subset V \setminus \{q\}$ of cardinality p can be found in G_t . Such value corresponds to $t = \frac{D_0}{D}$ in [12]. This alternative way to view the second improvement will be useful in what follows.

In the next section, we will further develop the observations made in [10, 12] and reported above, and we will propose a new approach where both the combinatorial part and the continuous part of multiple obnoxious FLPs will be exploited.

3 A new approach combining combinatorial and continuous aspects

Due to the extremely large number of local maximizers, Multistart approaches with starting points randomly generated within the unit square perform quite badly over problem (1).

However, as already observed, the problem also has a combinatorial nature since, rather than considering all points in the unit square, we can restrict our attention to points in the finite set \mathcal{H} . This is exactly what has been done in [10], where a set of

p points in \mathcal{H} fulfilling the FFD constraints is detected through the solution of the combinatorial optimization problem (2).

But the second and third improvement techniques proposed in [12] suggest that combining the combinatorial nature of the problem with its continuous aspects may be fruitful. Indeed, both these improving techniques run the local solver SLP to refine a solution obtained through a combinatorial technique, where points are constrained to belong to the finite set \mathcal{H} , while in the local solver the continuous nature of the problem is re-introduced, since points are free to move within the unit square and not constrained to belong to the set \mathcal{H} .

The main idea of this paper is to extend this combination of combinatorial and continuous approaches to tackle problem (1). The proposed approach performs a combinatorial local search through set of p points belonging to \mathcal{H} (combinatorial part of the algorithm), but associate to each such set of p points a value which is the locally optimal value detected by a local solver applied to problem (1) when the initial solution is exactly the given set of p points (continuous part of the algorithm). For $t \leq 1$, we denote by \mathcal{C}_t the collection of p -cliques in G_t , i.e., cliques in G_t with cardinality p . For a clique $C \in \mathcal{C}_t$ we consider the set of p points H_C as defined in (3) but we remark that, unless $t = 1$, such set of points is not necessarily a feasible solution for (1). For each $i \in V$, we let

$$N_i(i) = \{q \in V \mid (i, q) \in E_t\},$$

denote the set of neighbors of node i in the graph G_t .

The algorithm works as follows.

- We generate a first set C^* of p points by solving problem (2) but over the set \mathcal{C}_t of p -cliques in graph G_t (line 1), and then we compute a local maximizer of problem (1) by running a local solver from the set of points H_{C^*} (line 2).
- Next, we try to refine the current solution C^* entering the `While` loop at line 4. In the body of the `While` loop we consider p -cliques C differing from C^* for a single node, and start a local solver from the corresponding set H_C (lines 6–11).
- If at least one of these p -cliques leads to a better solution, we set $C^* = C$ and update the current best objective function value (lines 12–13), and then we repeat the whole body of the `While` loop.
- If no p -clique leads to an improvement (`NoImp` = 0 at the end of the `For` loop), we stop and return the current set C^* and the current best observed value L^* .

This procedure is run for different values $t \in (0, 1]$. As we decrease t , we enlarge the set of p -cliques, and, thus, expand the set of potential starting points for local maximization. Note that the set \mathcal{H} of Voronoi points, which is the main input of Algorithm 1, can be computed quite efficiently by existing procedures.

In particular, in our experiments we employed a Matlab procedure to generate this set.

Algorithm 1 Solution algorithm for the multiple obnoxious facility location problem (1)

Input: \mathcal{H} , $t \in (0, 1]$

- 1 Let $C^* \in \arg \max_{C' \in \mathcal{C}_t} \min_{i \in C'} F_i$;
- 2 Run a local solver for problem (1) starting from the set of points H_{C^*} and let L^* be the corresponding locally optimal value;
- 3 Set NoImp = 0;
- 4 **while** NoImp = 0 **do**
- 5 Set NoImp = 1;
- 6 **for** $j \leftarrow 1$ **to** R **do**
- 7 **if** $j \notin C^*$ **then**
- 8 **if** $|C^* \setminus N_t(j)| = 1$ **then**
- 9 Let $q \in C^* \setminus N_t(j)$;
- 10 Set $C = C^* \cup \{j\} \setminus \{q\}$;
- 11 Run a local solver for problem (1) starting from the set of points H_C and let L be the corresponding locally optimal value;
- 12 **if** $L > L^*$ **then**
- 13 Set $L^* = L$, $C^* = C$, NoImp = 0;
- 14 **return** C^*, L^* ;

Further considerations to improve the efficiency of the algorithm, by performing more efficiently both its combinatorial part and its continuous part, are discussed in what follows.

3.1 Solution of the combinatorial optimization problem

The combinatorial optimization problem (2) is solved only once. Its solution may be computationally expensive if the set \mathcal{H} , and, thus, the number of binary variables to be introduced in the mathematical programming model of the problem, is large. The cardinality of \mathcal{H} increases more or less linearly with n (in our experiments it is slightly more than 200 for $n = 100$, and slightly more than 2000 for $n = 1000$). For this reason, while for $n = 100$ we used the full set \mathcal{H} as input set for Algorithm 1, for $n = 1000$ we considered a reduced set \mathcal{H} , only including points $i \in \mathcal{H}$ with largest F_i value. In particular, in our experiments we only included the 250 points with largest value. In this case the vertex set of a graph G_t is the index set of such points in \mathcal{H} with largest value.

3.2 Solution of the continuous optimization problem

Another cost for Algorithm 1 is represented by the calls to the local solver, the single one at line 2 and, above all, those at line 11, one for each newly generated p -clique. The local solver S_{opt} , employed in our computational experiments, turns out to be quite efficient for the local maximization of problem (1). However, after testing it for the two distinct values $n = 100$ and $n = 1000$, we observed that the performance of the local solver becomes quite poor for $n = 1000$. This is due to the fact that the number of constraints of problem (1) is $O(p^2 + pn)$. However, at any local maximizer most of the FCD constraints $(x_j - a_i)^2 + (y_j - b_i)^2 \geq L^2, i \in N, j \in P$, are not active. Therefore, we propose to solve the following problem with a reduced set of FCD constraints. Let $C = \{i_1, \dots, i_p\}$ be a p -clique, and consider the associated set of points $H_C = \{(z_x^r, z_y^r) \mid r \in C\}$. We limit the variables $x_j, y_j, j = 1, \dots, p$, to the intervals $[\ell_j^x, u_j^x]$ and $[\ell_j^y, u_j^y]$, as follows:

$$\begin{aligned} \ell_j^x &= \max\{0, z_x^{i_j} - 0.1\} & u_j^x &= \min\{1, z_x^{i_j} + 0.1\} \\ \ell_j^y &= \max\{0, z_y^{i_j} - 0.1\} & u_j^y &= \min\{1, z_y^{i_j} + 0.1\}, \end{aligned}$$

i.e., we only allow to move in a neighborhood of point $(z_x^{i_j}, z_y^{i_j})$. Next, we consider the following subsets of N :

$$N_j = \left\{ i \in N \mid a_i \in [\ell_j^x - 0.05, u_j^x + 0.05], b_i \in [\ell_j^y - 0.05, u_j^y + 0.05] \right\}.$$

The cardinalities of the sets N_j are much lower than the cardinality of N . Finally, rather than solving problem (1), we solve problem

$$\begin{aligned} \max L & \\ (x_k - x_j)^2 + (y_k - y_j)^2 &\geq D^2 & k, j \in P, k \neq j \\ (x_j - a_i)^2 + (y_j - b_i)^2 &\geq L^2 & i \in N_j, j \in P \\ L &\geq 0 \\ (x_j, y_j) &\in [\ell_j^x, u_j^x] \times [\ell_j^y, u_j^y] & j \in P. \end{aligned} \tag{4}$$

By this localization of the FCD constraints, we can keep their number much lower than those in (1) and, most of all, we can keep the number independent from n . Some care is needed when evaluating a local maximizer of problem (4). We cannot simply take the optimal value of the problem, since in the constraints defining L , we are omitting all $i \notin N_j$. Therefore, once a local maximizer of problem (4), say $(x_j^*, y_j^*), j \in P$, is detected, we evaluate the original objective function value at such point, i.e.,

$$\min_{i \in N, j \in P} \sqrt{(x_j^* - a_i)^2 + (y_j^* - b_i)^2}.$$

In practice, due to the restrictions $(x_j, y_j) \in [\ell_j^x, u_j^x] \times [\ell_j^y, u_j^y], j \in P$, the original objective function value is always equivalent to the optimal value of (4). At the same

time, the computational requirements for the solution of (4) are much lower than those for the solution of (1). These computational savings make Algorithm 1 efficient also when n is large. Although we have not done this in our experiments, we could actually discard in advance also some FFD constraints. Indeed, if the minimum distance between points in the rectangle $[\ell_j^x, u_j^x] \times [\ell_j^y, u_j^y]$ and points within the rectangle $[\ell_k^x, u_k^x] \times [\ell_k^y, u_k^y]$ is not smaller than D , then we can discard the FFD constraint associated to the pair composed by $j, k \in P$.

4 Computational experiments

The computational experiments have been performed on an Intel(R) Core(TM) i7-1255U, 1.70 GHz, 32GB RAM. The algorithm has been implemented in Ampl [11]. For the solution of the combinatorial problem (2) we employed the solver Gurobi, while local maximization has been performed by the solver Snopt.

4.1 Test instances

We employed the set of test problems also employed in [10, 12]. In those papers tests have been performed with:

- $n = 100$ and $n = 1000$;
- $p \in \{2, \dots, 20\}$;
- $D = \frac{1}{\sqrt{p}}$ and $D = \frac{1}{\sqrt{2p}}$.

The set \mathcal{A} of n points has been generated through the procedure described in Algorithm 2. Note that this algorithm can be found in [10], but it is reported here to make the paper self-contained.

Algorithm 2 Procedure to generate set \mathcal{A} for the test instances

Input: n

- 1 Set $r_1^x = 67$ and $r_1^y = 367$;
- 2 Set $a_1 = \frac{r_1^x}{100000}$ and $b_1 = \frac{r_1^y}{100000}$;
- 3 **for** $k \leftarrow 2$ **to** n **do**
- 4 Set $\theta_1 = 12219r_{k-1}^x$ and $\theta_2 = 12219r_{k-1}^y$;
- 5 Set $r_k^x = \theta_1 - \lfloor \frac{\theta_1}{100000} \rfloor 100000$ and $r_k^y = \theta_2 - \lfloor \frac{\theta_2}{100000} \rfloor 100000$;
- 6 Set $a_k = \frac{r_k^x}{100000}$ and $b_k = \frac{r_k^y}{100000}$;
- 7 **return** $\mathcal{A} = \{(a_i, b_i), i \in N\}$;

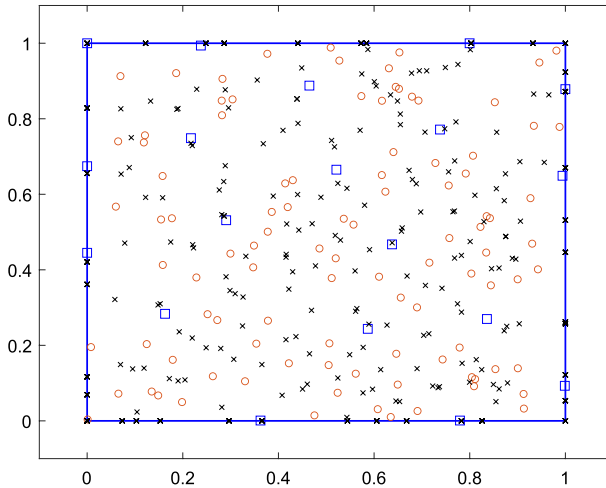


Fig. 1 New best solution for $n = 100$, $p = 19$, $D = \frac{1}{\sqrt{p}}$, improving the one reported in [12]. Facilities are denoted by blue squares, communities by small red circles, Voronoi points by black crosses

4.2 Choice of the algorithm parameter t

For tests with $n = 100$ we run Algorithm 1 with the same set of six values $t \in \{0.75, 0.8, \dots, 1\}$ both for $D = \frac{1}{\sqrt{p}}$ and for $D = \frac{1}{\sqrt{2p}}$. For tests with $n = 1000$ we run Algorithm 1 with larger values for t with respect to the case $n = 100$, and, moreover, we used different values for $D = \frac{1}{\sqrt{p}}$ and for $D = \frac{1}{\sqrt{2p}}$. More precisely, we used

Table 1 Results for $n = 100$, $D = \frac{1}{\sqrt{p}}$, $t \in \{0.75, 0.8, 0.85, 0.9, 0.95, 1\}$

p	1.00	0.95	0.90	0.85	0.80	0.75	Best (%)
10	0.0954	0.0968	0.0968	0.0968	0.0970	0.0971	0.09705 (0.24)
11	0.0954	0.0954	0.0954	0.0952	0.0952	0.0952	0.095394 (0)
12	0.0952	0.0952	0.0952	0.0925	0.0935	0.0935	0.095171 (0)
13	0.0944	0.0944	0.0945	0.0945	0.0946	0.0857	0.094559 (0.06)
14	0.0813	0.0933	0.0874	0.0896	0.0895	0.0824	0.093306 (12.75)
15	0.0814	0.0933	0.0933	0.0933	0.0933	0.0824	0.093306 (0)
16	0.0813	0.0813	0.0846	0.0705	0.0846	0.0813	0.08460 (0)
17	0.0754	0.0824	0.0824	0.0758	0.0846	0.0768	0.084607 (3.93)
18	0.0731	0.0813	0.0780	0.0822	0.0817	0.0817	0.0822201 (8.32)
19	0.0754	0.0813	0.0813	0.0783	0.0833	0.0889	0.088885 (13.56)
20	0.0768	0.0813	0.0814	0.0814	0.0849	0.0768	0.084892 (4.25)

Column Best (%) reports the best observed value and within parentheses the percentage improvement with respect to the result reported in [12]

Results in bold are those improving the results reported in [12]

Table 2 Results for $n = 100, D = \frac{1}{\sqrt{2p}}, t \in \{0.75, 0.8, 0.85, 0.9, 0.95, 1\}$

p	1.00	0.95	0.90	0.85	0.80	0.75	Best (%)
10	0.1022	0.1022	0.1022	0.1022	0.0938	0.0959	0.10219 (0)
11	0.1011	0.1011	0.1017	0.1010	0.1022	0.1011	0.10219 (1.07)
12	0.1005	0.1011	0.0993	0.0993	0.0669	0.0817	0.1011 (0)
13	0.1011	0.1011	0.1011	0.0992	0.0991	0.1005	0.1011 (0)
14	0.0965	0.1011	0.1011	0.1011	0.1011	0.1017	0.10173 (1.17)
15	0.0965	0.1010	0.0931	0.1010	0.0953	0.0931	0.10096 (0.42)
16	0.0965	0.0993	0.0993	0.0993	0.1010	0.0959	0.10096 (0.42)
17	0.0965	0.0965	0.0965	0.1005	0.0987	0.0938	0.10054 (0)
18	0.0954	0.0954	0.0954	0.0965	0.1005	0.0957	0.10054 (4.04)
19	0.0945	0.0954	0.0954	0.0965	0.0977	0.0827	0.097656 (1.85)
20	0.0943	0.0943	0.0943	0.0965	0.0992	0.0667	0.099178 (3.82)

Column Best (%) reports the best observed value and within parentheses the percentage improvement with respect to the result reported in [12]

Results in bold are those improving the results reported in [12]

Table 3 Results for $n = 1000, D = \frac{1}{\sqrt{p}}, t \in \{0.95, 0.96, 0.97, 0.98, 0.99, 1\}$

p	1.00	0.99	0.98	0.97	0.96	0.95	Best (%)
10	0.0397	0.0397	0.0397	0.0397	0.0402	0.0402	0.040239 (1.27)
11	0.0381	0.0391	0.0264	0.0391	0.0397	0.0397	0.039664 (1.87)
12	0.0391	0.0391	0.0391	0.0391	0.0391	0.0391	0.039123 (0)
13	0.0381	0.0381	0.0384	0.0384	0.0370	0.0370	0.038379 (0)
14	0.0381	0.0381	0.0381	0.0385	0.0385	0.0385	0.038496 (1.09)
15	0.0370	0.0381	0.0381	0.0344	0.0344	0.0344	0.038075 (0)
16	0.0370	0.0370	0.0370	0.0370	0.0370	0.0366	0.037049 (0)
17	0.0359	0.0294	0.0359	0.0359	0.0374	0.0374	0.037418 (3.95)
18	0.0343	0.0350	0.0345	0.0348	0.0349	0.0349	0.035011 (0)
19	0.0353	0.0353	0.0353	0.0343	0.0353	0.0332	0.035288 (0)
20	0.0333	0.0333	0.0343	0.0352	0.0343	0.0353	0.035288 (5.43)

Column Best (%) reports the best observed value and within parentheses the percentage improvement with respect to the result reported in [12]

Results in bold are those improving the results reported in [12]

the set of six values $t \in \{0.95, 0.96, \dots, 1\}$ for $D = \frac{1}{\sqrt{p}}$, while for $D = \frac{1}{\sqrt{2p}}$ we used the set of six values $t \in \{0.9, 0.92, \dots, 1\}$. The motivation for these differences is the following. The lower the value t , the further from feasibility with respect to the FFD constraints of problem (1) the sets of points associated to p -cliques in C_t are likely to be. The local solver is able to drive solutions violating the FFD constraints back to the feasible region of (1), but smaller t values require larger perturbations of the points in order to recover feasibility. Recovering feasibility gets more difficult as the threshold distance D increases and for this reason in the tests with $n = 1000$ we

Table 4 Results for $n = 1000, D = \frac{1}{\sqrt{2p}}, t \in \{0.9, 0.92, 0.94, 0.96, 0.98, 1\}$

p	1.00	0.98	0.96	0.94	0.92	0.90	Best (%)
10	0.0434	0.0434	0.0434	0.0434	0.0434	0.0434	0.043385 (0)
11	0.0416	0.0416	0.0416	0.0416	0.0416	0.0416	0.04156 (0)
12	0.0416	0.0416	0.0416	0.0416	0.0416	0.0416	<i>0.04156 (-0.04)</i>
13	0.0412	0.0410	0.0415	0.0412	0.0415	0.0415	<i>0.041552 (-0.02)</i>
14	0.0412	0.0412	0.0407	0.0412	0.0412	0.0410	0.041193 (0)
15	0.0397	0.0397	0.0397	0.0397	0.0397	0.0398	0.03978 (0)
16	0.0397	0.0397	0.0397	0.0400	0.0400	0.0405	0.040549 (2.02)
17	0.0396	0.0396	0.0397	0.0402	0.0392	0.0392	0.040239 (1.43)
18	0.0397	0.0397	0.0402	0.0402	0.0389	0.0401	0.040239 (1.27)
19	0.0396	0.0402	0.0404	0.0405	0.0405	0.0405	0.040461 (0.44)
20	0.0402	0.0407	0.0407	0.0407	0.0407	0.0402	0.040705 (1.14)

Column Best (%) reports the best observed value and within parentheses the percentage improvement (or worsening, if negative) with respect to the result reported in [12]

Results in bold are those improving the results reported in [12], while results in italics are those worsening the results reported in [12]

Table 5 Results for $n = 1000, D = \frac{1}{\sqrt{2p}}, t \in \{0.8, 0.85\}$

p	0.85	0.80	Best (%)
10	0.0434	0.0416	0.043385 (0)
11	0.0413	0.0413	<i>0.041295 (-0.64)</i>
12	0.0416	0.0434	0.043385 (4.17)
13	0.0416	0.0416	0.04156 (0)
14	0.0412	0.0396	0.041193 (0)
15	0.0393	0.0391	<i>0.039301 (-1.22)</i>
16	0.0389	0.0374	<i>0.03892 (-2.08)</i>
17	0.0392	0.0380	<i>0.039184 (-1.23)</i>
18	0.0402	0.0396	0.040239 (1.27)
19	0.0407	0.0367	0.040705 (1.04)
20	0.0407	0.0402	0.040705 (1.14)

Column Best (%) reports the best observed value and within parentheses the percentage improvement (or worsening, if negative) with respect to the result reported in [12]

Results in bold are those improving the results reported in [12], while results in italics are those worsening the results reported in [12]

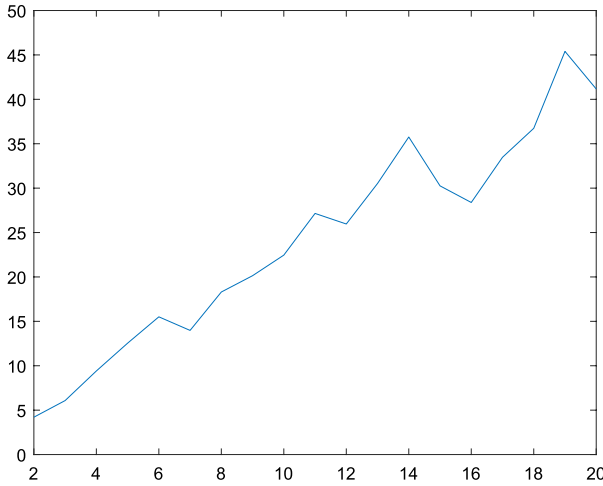


Fig. 2 Computing times (in seconds) as a function of p for $n = 1000$, $D = \frac{1}{\sqrt{2p}}$, and $t = 0.90$

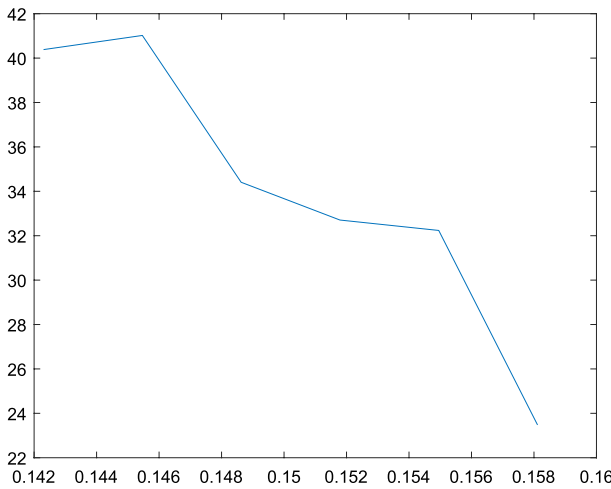


Fig. 3 Computing times (in seconds) as a function of tD for $n = 1000$, $D = \frac{1}{\sqrt{2p}}$, and $t \in \{0.9, \dots, 1.00\}$

considered larger values of t when $D = \frac{1}{\sqrt{p}}$. Moreover, as we increase n , we increase the density of the set of points \mathcal{A} and, consequently, it is more likely that a large perturbation causes a decrease of the objective function represented by the minimal distance from points in \mathcal{A} . For this reason we consider larger values of t for $n = 1000$ with respect to $n = 100$.

4.3 Results and discussion

The results are displayed in Tables 1, 2, 3 and 4. In all these tables there is one column for each tested value t . The last column reports the best obtained value (with a higher number of decimal digits) and, within parentheses, we report the percentage difference with respect to the currently best results reported in [12]. Improvements with respect to those results are highlighted in boldface in the last column, while worse results are displayed in italics. Note that we only report results for $p \geq 10$, since instances with $p < 10$ are simpler, and by the proposed approach the same results reported in [12] are obtained. It is interesting to note that the results obtained with $t = 1$ are dominated by those obtained with lower values of t , so that the best result would not change by removing tests with $t = 1$. This confirms that allowing for violation of the FFD constraints in the combinatorial part of the algorithm is effective. Improvements are detected in 24 instances and most of the percentage improvements are larger than 1%. In a couple of instances ($p = 12, 13$) with $n = 1000$ and $D = \frac{1}{\sqrt{2p}}$, we were not able to reach the best result reported in [12], but the percentage difference is very small (below 0.1%). However, these failures pushed us to perform further tests with smaller values of $t = 0.8, 0.85$. The results are reported in Table 5. While for some p values the results obtained with these t values are poor, the two failures for $p = 12, 13$ are eliminated and, moreover, a much better solution is obtained for $p = 12$ and a slightly better one is obtained for $p = 19$. This suggests that the selection of the t values might have a significant impact on the performance of the approach.

For the sake of illustration, in Fig. 1 we display the new best solution for $n = 100$, $p = 19$, $D = \frac{1}{\sqrt{p}}$ (facilities are the blue squares, communities are the small red circles, Voronoi points are the black crosses).

We stress the fact that in Algorithm 1 the combinatorial search for p -cliques in \mathcal{C}_t has been performed through a simple discrete local search procedure, corresponding to the `While` loop of the algorithm. We believe that the current results can be improved by more sophisticated searches within the collection of p -cliques, but in this paper we just wanted to put in evidence that even a simple combinatorial method, like the proposed combinatorial local search, is able to lead to results improving those available in the literature.

Concerning the computational times, we do not report all of them but we just display in Figs. 2 and 3 the computing times for some time-consuming instances, namely those with $n = 1000$, $D = \frac{1}{\sqrt{2p}}$, $t = 0.9$ and p ranging from 2 to 20 (Fig. 2), and with $n = 1000$, $D = \frac{1}{\sqrt{2p}}$, $p = 20$ and t ranging from 0.9 to 1.0 (Fig. 3, where the x -axis reports the values tD). Note that, as expected, computing times tend to increase with p (the number of variables and of constraints in (4) increases with p), while they tend to decrease with t (the number of p -cliques in \mathcal{C}_t decreases with t).

5 Conclusions and future work

In this paper we addressed the multiple obnoxious FLP where p facilities have to be placed in the unit square in such a way that their reciprocal distance never falls below a threshold D and that their minimal distance from n communities, also lying within the unit square at predefined positions, is maximized.

Exploiting the key observation made in [10] about the relevance of Voronoi points for this problem, we extended the approaches proposed in the same paper and in [12], by taking advantage both of the combinatorial aspects of the problem and of the continuous ones. Indeed, the proposed approach first solves a combinatorial problem where the positions of the facilities are discretized and restricted to Voronoi points, and then performs a discrete local search over the set of p -cliques of a suitably defined graph, where, in turn, each p -clique is evaluated through a continuous local maximization.

By the proposed approach, all the best solutions over a set of 44 test instances known in the literature have been detected, and for 24 instances we were able to report better solutions. In the test instances we employed the values $n = 100$ and $n = 1000$ for the communities.

Tests with $n = 1000$ revealed that large n values cause large computing times both for the solution of the combinatorial optimization problem, and for the local maximization. Therefore, techniques to reduce the impact of large n values have been introduced, by pre-selecting a subset of the discretized positions (Voronoi points), and by restricting the possible perturbations of positions in the local searches. While the proposed techniques work well, the search for further techniques to reduce the impact of large n values, which may further reduce the computing times, are an interesting topic for future research.

A further interesting topic for future research is the development of automatic techniques to detect the most suitable values for the parameter t employed in Algorithm 1.

Finally, exploration of the discrete space of p -cliques of graph G_t is currently performed through a simple discrete local search technique. As already remarked, the search for alternative approaches to explore this discrete space is also an interesting topic for future research.

Funding Open access funding provided by Università degli Studi di Parma within the CRUI-CARE Agreement.

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

References

1. Aardal, K.: Capacitated facility location: separation algorithms and computational experience. *Math. Program.* **81**(2), 149–175 (1998)
2. Ahmadi-Javid, A., Seyedi, P., Syam, S.S.: A survey of healthcare facility location. *Comput. Op. Res.* **79**, 223–263 (2017)
3. Aurenhammer, F., Klein, R., Lee, D.T.: *Voronoi diagrams and Delaunay triangulations*. World Scientific, Singapore (2013)
4. Balinski, M.L.: Integer programming: methods, uses, computation. *Manage. Sci.* **12**(3), 253–313 (1965)
5. Cappanera, P., Gallo, G., Maffioli, F.: Discrete facility location and routing of obnoxious activities. *Discret. Appl. Math.* **133**(1–3), 3–28 (2003)
6. Castillo, I., Kampas, F.J., Pintér, J.D.: Solving circle packing problems by global optimization: numerical results and industrial applications. *Eur. J. Op. Res.* **191**(3), 786–802 (2008)
7. Celik Turkoglu, D., Erol Genevois, M.: A comparative survey of service facility location problems. *Ann. Op. Res.* **292**(1), 399–468 (2020)
8. Courtillot, M.: New methods in mathematical programming-on varying all the parameters in a linear-programming problem and sequential solution of a linear-programming problem. *Oper. Res.* **10**(4), 471–475 (1962)
9. Drezner, Z., Kalczyński, P.: Solving nonconvex nonlinear programs with reverse convex constraints by sequential linear programming. *Int. Trans. Oper. Res.* **27**(3), 1320–1342 (2020)
10. Drezner, Z., Kalczyński, P., Salhi, S.: The planar multiple obnoxious facilities location problem: A Voronoi based heuristic. *Omega* **87**, 105–116 (2019)
11. Fourer, R., Gay, D.M., Kernighan, B.W.: *AMPL: a modeling language for mathematical programming*, 2nd edn. Brooks/Cole, Belmont, CA
12. Kalczyński, P., Drezner, Z.: Extremely non-convex optimization problems: the case of the multiple obnoxious facilities location. *Optim. Lett.* **16**(4), 1153–1166 (2022)
13. Klose, A., Drexl, A.: Facility location models for distribution system design. *Eur. J. Oper. Res.* **162**(1), 4–29 (2005)
14. Krarup, J., Pruzan, P.M.: The simple plant location problem: survey and synthesis. *Eur. J. Op. Res.* **12**(1), 36–81 (1983)
15. Locatelli, M., Raber, U.: Packing equal circles in a square: a deterministic global optimization approach. *Discret. Appl. Math.* **122**(1–3), 139–166 (2002)
16. Mihály, M.C., Csentes, T.: Csentes: a new verified optimization technique for the "packing circles in a unit square" problems. *SIAM J. Optim.* **16**(1), 193–219 (2005)
17. Nurmela, K.J., Östergård, P.R.J.: Packing up to 50 equal circles in a square. *Discret. Comput. Geom.* **18**, 111–120 (1997)
18. Plastria, F.: Continuous location problems: research, results and questions. In: *Facility location: a survey of applications and methods*, pp 85–127 (1995)
19. Tamir, A.: Obnoxious facility location on graphs. *SIAM J. Discret. Math.* **4**(4), 550–567 (1991)

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