A branch-and-cut method for the obnoxious p -median problem

Article *in* 4OR · December 2007

DOI: 10.1007/s10288-006-0023-3 · Source: DBLP

4OR manuscript No.
(will be inserted by the editor)

A branch-and-cut method for the Obnoxious p-Median problem

Pietro Belotti[1], **Martine Labbé**[2], **Francesco Maffioli**[1], **Malick Ndiaye**[3]

[1] Electronics and Computer Science Department, Politecnico di Milano, Italy.
e-mail: {belotti,maffioli}@elet.polimi.it
[2] Computer Science Department, Université Libre de Bruxelles, Belgium.
e-mail: mlabbe@ulb.ac.be
[3] KFUPM, System Engineering Dept., Dharan 31261, Saudi Arabia.
e-mail: mndiaye@ccse.kfupm.edu.sa

The date of receipt and acceptance will be inserted by the editor

Abstract The Obnoxious p-Median (OpM) problem is the repulsive counterpart of the more known attractive p-median problem. Given a set I of cities and a set J of possible locations for obnoxious plants, a p-cardinality subset Q of J is sought, such that the sum of the distances between each city of I and the nearest obnoxious site in Q is maximised. We formulate OpM as a $\{0,1\}$ Linear Programming problem and propose three families of valid inequalities whose separation problem is polynomial. We describe a branch-and-cut approach based on these inequalities and apply it to a set of instances found in the location literature. The computational results presented show the effectiveness of these inequalities for OpM.

Keywords: Obnoxious Facility Location – Branch-and-cut – p-Median.

Mathematics Subject Classification (2000): 90C57–90B80

1 Introduction

A facility is called *obnoxious* when it is desired to locate it as far as possible from an inhabited centre. Obnoxious location problems have received significant attention in the last decades, due to the increasing environmental and social impact of facilities such as power plants and dump sites. Several models have been presented in the Operations Research literature for placing such facilities either on graphs or

The work of the first author has been partially supported by the Coordinated Project C.A.M.P.O. and that of the third author by a short mobility grant, both of the Italian National Research Council.

in a Euclidean space. In Erkut and Neuman (1989) many such problems are discussed and classified. More recent surveys are, for instance, Cappanera (1999), Drezner and Hamacher (2002), and Eiselt and Laporte (1995).

Generally, given a set J of possible locations for facilities, one seeks a subset Q of J with given properties. The distance between points i and j in J is denoted as c_{ij}. Some problems admit the a priori installation of a subset of sites, to be considered in the objective or in the problem constraints.

Many real–world problems deal with facilities for which transportation costs (Carrizosa and Conde, 2002) or distance constraints (Moon and Chaudhry, 1983) must be taken into account. Such facilities (e.g. airports, recycle plants etc.) are called *semi-obnoxious*: although their immediate closeness is disturbing or dangerous, they cannot be located too far. This situation can be tackled by minimising total nuisance while bounding distances from clients, or minimising transportation costs while guaranteeing a given distance from clients. This work deals with fully obnoxious sites, hence we do not consider transportation costs or distance limits.

If p undesirable facilities have to be located, one possibility is to maximise the sum of all inter-facility distances, thus adopting a maxi-sum objective. These are called p*-maxian* problems (Church and Garfinkel, 1978; Kincaid, 1992) and their NP-hardness is proven in Hansen and Moon (1988). In Kuby (1987) a $\{0, 1\}$ Linear Programming model is presented for p-maxian problems, and in Chandrasekaran and Daughety (1981) a polynomial algorithm is presented for tree networks. In Erkut et al. (1990) the p-maxian problem is solved through an implicit enumeration method.

In *dispersion* problems (Shier, 1977) the minimum distance between all pairs of facilities is maximised, thus adopting a maxi-min model. Erkut (1989) presents a model and a heuristic for p-dispersion instances with $|J| = 40$ and $p = 16$, while a greedy algorithm for dispersion problems is presented in Erkut and Neuman (1989) Another possibility is to adopt a maxi-sum-min scheme: in *defense* problems (Chaudhry et al., 1986) the sum of minimum inter-facility distance is maximised. A dynamic programming scheme is presented in Chhajed and Lowe (1994) for solving diverse maxi-sum and dispersion problems. In Tamir (1991) the complexity of several obnoxious problems is addressed. Both p-maxi-sum and p-maxi-min problems are shown to be NP-hard for general graphs and in the discrete case. A polynomial heuristic is then presented for p-maxi-sum.

In some problems, a set I of *clients* is given, possibly disjoint from J. When maximising the minimum distance between client sites contained in I and their respective nearest facility, we speak of *anti-center* problems (see e.g. Klein and Kincaid (1994)). In *anti-median* problems, the sum of distances between every site in I and its nearest open facility is maximised. One such example of maxi-sum-min scheme with facility dispersion is introduced in Ting (1988). Welch and Salhi (1997) discuss a combination of p-maxi-sum and p-maxi-min problem. Minieka (1983) studies the single facility anti-median and anti-center problems. In Burkard

et al. (2000) an $O(n^3)$ algorithm is described for solving the mini-sum-min weighted problem, where negative weights are used to model obnoxious sites.

In the class of maxi-sum-min problems, which is of interest to us, the objective function is a combination of

$$\sum_{i\in I} \min_{j\in Q} c_{ij}, \qquad \sum_{j\in Q} \min_{k\in Q\setminus\{j\}} c_{jk}.$$

The problem dealt with in this work, which we call Obnoxious p-Median (OpM), can be described as follows:

Input: a set I of clients and a set J of potential facility locations; a $n \times m$ matrix of distances $c_{ij} \in \mathbb{R}_+$, where $n = |I|$ and $m = |J|$; a positive integer $p < m$.

Output: a subset Q of J such that $|Q| = p$ and $\sum_{i\in I} \min\{c_{ij} : j \in Q\}$ is maximum.

OpM is a discrete maxi-sum-min problem where only the distance between clients and facilities is considered. A proof of its NP-hardness can be found in Tamir (1991). As all facilities in this work are treated as fully obnoxious, our model can be applied when transportation costs are negligible w.r.t. the overall damage or danger caused by each site. It is worth pointing out that since only distances between clients and facilities are taken into account, this model does not achieve facility dispersion, needed when facilities have a negative influence on one another. Such dispersion could be obtained, for example, by imposing a minimum inter-facility distance, but would also lead to a different type of problem.

Section 2 presents a $\{0, 1\}$ Linear Programming formulation of OpM. In order to solve mid-size instances of OpM, we have devised three families of valid inequalities, described in Section 3. In Section 4 we present a Branch-and-Cut method that we have applied on a number of OpM instances. We report computational results in Section 5 and provide some conclusions in Section 6.

2 A mathematical model

Consider, for each pair $i \in I$, $j \in J$, the set of facilities more distant than j from client i,

$$S(i, j) = \{k \in J : (c_{ik} > c_{ij}) \vee (c_{ik} = c_{ij} \wedge k > j)\}.$$

A client is said to be *allocated* to the open location closest to it. If location j is open, then client i cannot be allocated to any location in $S(i, j)$. Notice that two facilities having the same distance from a client are compared in a lexicographic manner. In the following, a facility is said *more distant* from a client i than another facility when the rule defined above applies, bearing in mind that this retains at least one optimal solution: two facilities j and k equally distant from client i have the same coefficient $c_{ij} = c_{ik}$ in the objective function.

For each $j \in J$, we define the binary variable

$$y_j = \begin{cases} 1 & \text{if a facility is located at } j, \\ 0 & \text{otherwise,} \end{cases}$$

and for each $i \in I$, $j \in J$, the variable

$$x_{ij} = \begin{cases} 1 & \text{if client } i \text{ is allocated to } j, \\ 0 & \text{otherwise.} \end{cases}$$

A formulation of OpM is as follows

$$\begin{aligned} & \max \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j \in J} y_j \geq p & & (1) \\ & x_{ij} \leq y_j & \forall i \in I, j \in J & (2) \\ & y_j + \sum_{k \in S(i,j)} x_{ik} \leq 1 & \forall i \in I, j \in J & (3) \\ & x_{ij}, y_j \in \{0,1\} & \forall i \in I, j \in J. & (4) \end{aligned}$$

Note that (1) can be inequality instead of equality constraints because of the assumption that all c_{ij}'s are positive. The importance of client i may be taken into account, for instance to reflect its population α_i, by replacing c_{ij} in the objective function with $c'_{ij} = \alpha_i c_{ij}$. Constraint (2) imposes that a client be allocated to an open location, whereas (3) allocates each client to the nearest open location and is required in this obnoxious location problem as opposed to attractive location problems. Constraint (4) specifies the discreteness of the problem variables.

Let $F_k(i)$ be the k-th most distant location from i, i.e., the location j such that $|S(i,j)| = k-1$. The most distant location from i is then $F_1(i)$ with this notation. It is barely worth noting here that some variables of the original model can be fixed from the beginning by setting $x_{ij} = 0$ for all $(i,j) \in I \times J$ such that $j \in S(i, F_p(i))$, since none of the $p-1$ facilities farthest from client i can be allocated to it. We also notice that a client i is allocated to its nearest location $j = F_m(i)$ if and only if j is open. Therefore, for each client i we have $x_{i\,F_m(i)} = y_{F_m(i)}$. For this reason, we do not include the constraint that assigns at least one facility to every client i, i.e., $\sum_{j \in J} x_{ij} \leq 1 \ \forall i \in I$, because this is already implied by (3) – again, we use "$\leq$" instead of equality because of non-negativity of c_{ij}'s.

3 Valid inequalities

Due to the size of common instances of OpM, a branch-and-bound approach may take great advantage of ad-hoc families of valid inequalities for the model. Therefore we propose a branch-and-cut method based on three families of valid inequalities, described in detail in the following subsections.

3.1 Inequalities A

If the open location nearest to client i is in $S(i,j)$, then so are all p open locations, as they are at least as distant from i as location j. In other words, $\sum_{k\in S(i,j)} x_{ik} = 1$ implies $\sum_{k\in S(i,j)} y_k \geq p$ and hence we have

$$\sum_{k\in S(i,j)} (px_{ik} - y_k) \leq 0. \tag{5}$$

To generalise (5), consider a subset A of $S(i,j)$. If $\sum_{k\in S(i,j)} x_{ik} = 1$, there are at least $p - |S(i,j) \setminus A|$ open locations in A (there are exactly as many if all locations in $S(i,j) \setminus A$ are open – see Figure 1a). We then have valid inequalities

$$(p - |S(i,j) \setminus A|) \sum_{k\in S(i,j)} x_{ik} \leq \sum_{k\in A} y_k, \tag{6}$$

defined by indices i, j and $A \subseteq S(i,j)$.

3.2 Inequalities B

Consider a pair of clients (i_1, i_2) and a pair of locations (j_1, j_2). If $S(i_1, j_1)$ and $S(i_2, j_2)$ have at most $p-1$ locations in common, then either i_1 is allocated to a location in $S(i_1, j_1)$ or i_2 is allocated to a location in $S(i_2, j_2)$, but not both, i.e., $\sum_{k\in S(i_1,j_1)} x_{i_1k}$ and $\sum_{k\in S(i_2,j_2)} x_{i_2k}$ cannot be both one. In fact, all p open locations would otherwise be in $S(i_1, j_1) \cap S(i_2, j_2)$, contradicting the assumption. We have then

$$\sum_{k\in S(i_1,j_1)} x_{i_1k} + \sum_{k\in S(i_2,j_2)} x_{i_2k} \leq 1 \tag{7}$$

for all i_1, i_2, j_1, j_2 such that

$$|S(i_1, j_1) \cap S(i_2, j_2)| \leq p - 1. \tag{8}$$

Consider now a location l in $J \setminus (S(i_1, j_1) \cup S(i_2, j_2))$. If l is open, then neither $S(i_1, j_1)$ nor $S(i_2, j_2)$ contain the open location to which i_1 or i_2 are allocated, since location l is closer to i_1 and to i_2 than any one contained in the two sets (see Figure 1b). As a consequence, only one of the three expressions $\sum_{k\in S(i_1,j_1)} x_{i_1k}$, $\sum_{k\in S(i_2,j_2)} x_{i_2k}$ and y_l can be nonzero. This gives the family of valid inequalities

$$\sum_{k\in S(i_1,j_1)} x_{i_1k} + \sum_{k\in S(i_2,j_2)} x_{i_2k} + y_l \leq 1 \tag{9}$$

with $(i_1, i_2, j_1, j_2, l) \in I^2 \times J^3$ s.t. (8) holds and $l \in J \setminus (S(i_1, j_1) \cup S(i_2, j_2))$. Although we have used inequality (9) in our tests, it can be generalised as follows: consider a set T of pairs (i, j) such that (8) holds between any two elements of T. Analogously to the clique inequality, a general version of (7) is

then $\sum_{(i,j)\in T}\sum_{k\in S(i,j)} x_{ij} \leq 1$. If there exists a facility l such that $l \in J \setminus \bigcup_{(i,j)\in T} S(i,j)$, then we have the generalised valid inequality

$$\sum_{(i,j)\in T}\sum_{k\in S(i,j)} x_{ik} + y_l \leq 1. \tag{10}$$

3.3 Inequalities C

Let us denote $S^A(i,j) = S(i,j) \cap A$ for any $A \subseteq J$, i.e., the set of locations in A more distant than j from client i. Let also $F_k^A(i)$ denote the k-th most distant location, among those in A, from i, i.e. $F_k^A(i)$ is the location j such that $|S^A(i,j)| = k-1$. Consider two clients i_1 and i_2. If i_1 is allocated to $F_p(i_1)$, i.e., $x_{i_1 F_p(i_1)} = 1$, then because of (5), the p locations most distant from i_1 and belonging to $B = S(i_1, F_{p+1}(i_1))$ are open. Client i_2 is allocated to the nearest open location of B, which is also the p-th most distant element of B from i_2, thus yielding $x_{i_2 F_p^B(i_2)} = 1$. Hence the valid inequality $x_{i_1 F_p(i_1)} \leq x_{i_2 F_p^B(i_2)}$.

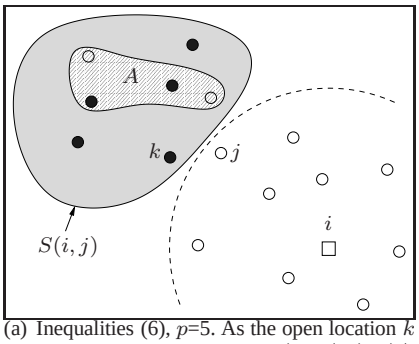
In general, if $\sum_{k\in S(i_1,j_1)} x_{i_1 k} = 1$ then client i_2 is allocated to an open location $j_2 \in S(i_1,j_1)$. However, j_2 is none of the $p-1$ locations most distant from i_2 within $S(i_1,j_1)$, i.e., $j_2 \notin \tilde{S} = S^{S(i_1,j_1)}(i_2, F_p^{S(i_1,j_1)}(i_2))$. Therefore, j_2 is to be found in $S(i_1,j_1) \setminus \tilde{S}$, (see Figure 1c). Hence the valid inequality

$$\sum_{k\in S(i_1,j_1)} x_{i_1 k} \leq \sum_{k\in S(i_1,j_1)\setminus\tilde{S}} x_{i_2 k}. \tag{11}$$

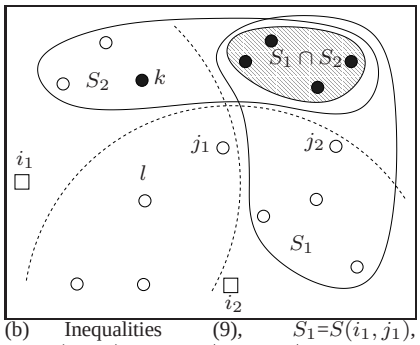
4 A Branch-and-Cut approach

In this section we present a branch-and-cut method (Naddef and Rinaldi, 2000) to tackle large instances of OpM. We outline below the procedures that improve the efficiency of our method: a Tabu Search heuristic that gives an initial lower bound, an upper bounding routine, a heuristic for generating a feasible solution, and a separation procedure for the three families of valid inequalities presented in the previous section. A best first strategy is used for selecting the next active node to analyse. We assume that a solution to the linear relaxation of OpM is given by $(y^\star, x^\star)$. We have substantially reduced the separation time through efficient data structures, such as the following matrices:

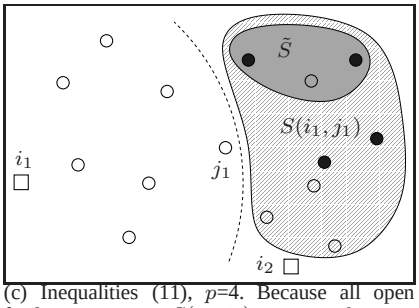
- L: element $l(i,j)$ gives the index of the j-th nearest location to client i (e.g., if 3 and 7 are the closest and the second closest location to client 8, respectively, then $l(8,1) = 3$ and $l(8,2) = 7$);
- U: $u(i,j)$ is the position of location j in row $l(i,\cdot)$ of L (it allows to detect whether $k \in S(i,j)$ for any triple (i,j,k));



(a) Inequalities (6), p=5. As the open location k is the nearest to i, x_{ik}=1. Thus $(p - |S(i,j) \setminus A|) = 2$ and $2\sum_{l \in S(i,j)} x_{il} \leq \sum_{l \in A} y_l$ is valid.



(b) Inequalities (9), S_1=$S(i_1, j_1)$, S_2=$S(i_2, j_2)$, p=5. As $|S_1 \cap S_2| \leq p - 1$, at least one open facility is not in $S_1 \cap S_2$, hence either $\sum_{j \in S_1} x_{i_1 j}$, $\sum_{j \in S_2} x_{i_2 j}$ or y_l can be one, where $l \in J \setminus (S_1 \cup S_2)$.



(c) Inequalities (11), p=4. Because all open facilities are in $S(i_1, j_1)$, so is the one closest to i_2. However, it cannot be in $\tilde{S} = S^{S(i_1,j_1)}(i_2, F_p^{S(i_1,j_1)}(i_2))$. Thus, $\sum_{k \in S(i_1,j_1)} x_{1k} \leq \sum_{k \in S(i_1,j_1) \setminus \tilde{S}} x_{2k}$.

- Open facility
- Unopened facility
- Client

Fig. 1. An explanation of the valid inequalities (Euclidean distance is assumed).

$\hat{X}, \hat{Z}$: values of $x^\star$ and $y^\star$ sorted w.r.t. L, i.e., $\hat{x}_{ij} = x^\star_{i\,l(i,j)}$ and $\hat{z}_{ij} = y^\star_{l(i,j)}$;
Θ, Λ, whose elements are the sum of $\hat{x}$ and $\hat{y}$, respectively, according to L, i.e., $\theta_{ij} = \sum_{k \in S(i,l(i,j))} x^\star_{ik}$ and $\lambda_{ij} = \sum_{k \in S(i,l(i,j))} y^\star_k$.

4.1 A Tabu Search heuristic

A lower bound is computed before applying the branch-and-cut method in order to further decrease the computational effort. We have applied a variant of the Tabu Search heuristic that has appeared in a previous work on the Cumulative Assignment problem with an application to satellite communications (Dell'Amico et al., 1999). This approach, called *eXploring Tabu Search* (X-TS), adds to the clas-

sic Tabu Search paradigm the following features, some of which are well-known techniques to diversify the search:

a) A long-term solution list: this list contains some *good* solutions that have been evaluated but not explored by the procedure. This technique allows for a better local exploration of the solution space.
b) A dynamic tabu list, whose length increases when the objective of the current solution decreases for h_{d} consecutive iterations and decreases when the current solution improves for h_{i} consecutive iterations.
c) A global restart mechanism that generates a new starting solution by applying a randomized greedy algorithm that uses the rank information described in Subsection 4.3 below.

Applying this heuristic step has boosted the performance of our algorithm, as the primal solution found by this procedure is often optimal. We shall not give details about this heuristic approach to OpM; the interested reader may refer to Belotti et al. (2000).

4.2 A bounding heuristic

As a lot of time is spent by the LP code at each node, before calling the LP solver it is effective to compute a less tight bound by a fast heuristic. If this upper bound is lower than the value of the best feasible solution found so far, the node can be fathomed. Given a node ν in the decision tree and a set of y variables that have been set in the ancestor nodes, let J_o be the set of open locations, $J_o = \{j : y_j = 1 \text{ at node } \nu\}$, and J_c the set of closed ones. Hence no feasible solution descending from ν can have objective greater than

$$\sum_{i \in I} \min \left(\min_{j \in J_o} c_{ij}, c_{iF_p^{J \setminus J_c}(i)} \right).$$

4.3 Obtaining feasible solutions

Given a fractional solution $y^\star$ to the linear relaxation of a problem in the branch-and-bound tree, a greedy solution Q' to OpM can be obtained as follows: consider a vector $\mathbf{d}$ such that $d_j = \min\{c_{ij} | i \in I\}$ for each location $j \in J$. Let us define a rank $r(j)$ for all $j \in J$ as the number of facilities k whose value of d_k is greater than d_j, i.e., $r(j) = |\{k \in J : d_k > d_j\}| \ \forall j \in J$. The lower the rank $r(j)$, the greater the distance from j to the nearest client, and therefore the more likely j is to be included in an optimal solution. The p elements with lowest rank are a solution to OpM that can be used to generate a feasible solution at low computational cost satisfying the branching constraints at each node of the decision tree.

Locations j whose variable has not been set at node ν are included in Q' depending on their fractional value in the current LP solution and on their rank. More

precisely, let $\alpha(j) = (1 + \gamma y_j^\star)\mu^{r(j)}$, where $\gamma, \mu \in [0, 1]$. Assuming k locations are already open according to the variables fixed by the branching, we include in Q' those $p - k$ location j with highest values of $\alpha(j)$. Parameters γ and μ balance the importance of $y_j^\star$ versus that of $r(j)$. In our tests we have used values $\mu = 0.85$ and $\gamma = 0.5$.

4.4 Separation routines

The separation step is performed after the linear relaxation of the subproblem associated with a node of the Branch-and-bound tree is solved. In order to give the same chances to our inequalities, we separate them in the same call to the routine and re-solve the LP solution if some new cuts have been added.

Inequalities A. These are uniquely identified by indices i, j and a set $A \subseteq S(i, j)$, whose cardinality is between $|S(i, j)| - p + 1$ and $|S(i, j)|$. In order not to insert too many inequalities at each iteration of the cutting plane routine, for each client i we look for the facility j and the subset A yielding the most violated inequality. Although the number of subsets A of $S(i, j)$ is exponential in $|S(i, j)|$, it suffices to consider the facilities k with small $y_k^\star$.

For each client i, consider all j such that $\sum_{k \in S(i,j)} x_{ik} > 0$. At most $(m - p)$ pairs (i, j) may give violated inequalities, as all locations k more distant than $F_p(i)$ from i have $x_{ik} = 0$. Consider a permutation of location indices $(k_1, k_2, \ldots, k_m)$ that sorts vector $y^\star$ in non-decreasing order, i.e., $y^\star_{k_1} \leq y^\star_{k_2} \leq \ldots \leq y^\star_{k_m}$. The set $A \subseteq S(i, j)$ of given cardinality h (where $|S(i, j)| - p + 1 \leq h \leq |S(i, j)|$) that minimizes the right-hand side of (6) is given by the first h elements of the permutation within $S(i, j)$ – set A is obtained by scanning the sorted array $S(i, j)$, a step whose complexity is $|S)i, j)| \leq m$. Hence, separating inequalities A requires $O(m \log m + n(m - p)m) = O(nm(m - p))$ steps.

Inequalities B. These are identified by a 5-ple (i_1, j_1, i_2, j_2, l) such that (8) holds and $l \in J \setminus (S(i_1, j_1) \cup S(i_2, j_2))$. Analogously to inequalities A, we limit the number of separated cuts by choosing at most one inequality for each (i_1, i_2) (we assume $i_1 < i_2$ to avoid symmetries). The triplet (j_1, j_2, l) that gives the maximally violated inequality can then be found in polynomial time. Although this procedure requires $O(n^2 m^3)$ steps, the actual separation time is negligible with respect to the LP time, as shown in the next section. It is also worth pointing out that the separation problem of the generalised inequality (10) is NP-hard as it is equivalent to the Max-Clique.

Inequalities C. Separating these inequalities takes more computational time than it seems. Although each is identified by a triplet (i_1, j_1, i_2), computing its violation requires repeated scans of vectors $x^\star$ and $y^\star$. However, the data structures described in the beginning of the section help reducing the running time. Again, we

have chosen to limit the number of inequalities by selecting, for each pair (i_1, i_2), the facility j_1 that gives the maximally violated inequality. As checking the violation for each triple requires $O(m)$ simple operations, the separation procedure has complexity $O(n^2m^2)$.

5 Experimental results

We have applied the algorithm described above to a set of instances with a wide range of values for n and m, so as to give a complete overview of the performance of our method when compared to a general MIP solver. We have obtained these instances from three sources available on the Internet:

- (http://www.bus.ualberta.ca/eerkut/testproblems) the p-median Test Problems Page, with several p-median instances used as test bed in Alp et al. (2003), Galvão and ReVelle (1996), and Koerkel (1989).
- (http://people.brunel.ac.uk/˜mastjjb/jeb/info.html) the OR-Library, with data sets for several Operations Research problems (Beasley, 1990).
- (http://www.lac.inpe.br/˜lorena/instancias.html) a web page with diverse p-median instances.

As the majority of the instances give a generic set of locations and do not discern clients from facilities, we have randomly divided them into sets I and J. None of them specifies the value of p, hence we solve each instance for p equal to $m/2$, $m/4$, and $m/8$. All tests have been performed on a Sun Fire 240 workstation, with two 1.64GHz UltraSparc64 processors, 4 GB of RAM memory, and the parallel Cplex solver. The routines explained in the previous section have been coded in C language and linked to Cplex 9.1's Callable Libraries. In order to evaluate the families of cuts introduced, we compare our branch-and-cut approach with the standard branch-and-cut procedure provided by Cplex with the default cutting planes, and the feasible solution computed by our Tabu Search procedure in the beginning is passed to both exact methods. A time limit of four hours has been given to both methods. In Tables 2, 1, and 3 we report, for each instance:

- the instance parameters (m, n and p);
- the running time of Cplex's branch-and-cut (cpu1) and of our branch-and-cut (cpu2) methods, respectively, if the instance is solved to optimality, otherwise in brackets the gap w.r.t. the best upper bound;
- the number of B&B nodes (nod1 and nod2) for the two algorithms;
- the number of inequalities separated (#A, #B, #C) and the total running time taken by the separation routines (sep).

For many instances, our branch-and-cut approach either takes shorter running time or obtains a smaller gap than the standard MILP method. As the primal bound is the same for the two methods, a smaller gap indicates that the valid inequalities separated are effective for OpM.

				cplex		b&c					
Name	n	m	p	cpu1	nod1	cpu2	nod2	#A	#B	#C	sep
orlib-cap41	50	16	4	0.3	0	0.3	0	0	0	0	0.01
orlib-cap81	50	25	6	0.5	0	0.5	0	0	0	0	0.01
orlib-cap111	50	50	12	2.2	0	2.2	0	0	0	0	0.03
Hoefer-O1	100	100	25	(43.8)	19119	1894.5	117	981	5863	0	11.67
Hoefer-O2	100	100	25	(45.4)	4671	(43.9)	1532	986	69	1	2.70
Hoefer-P1	200	200	50	(63.8)	11	(60.1)	0	3928	0	0	28.25
Hoefer-P2	200	200	50	(63.4)	230	(60.3)	0	3790	0	0	59.30
Hoefer-Q1	300	300	75	(65.7)	2	(64.3)	0	3300	0	0	153.27
Hoefer-Q2	300	300	75	(65.7)	8	(64.2)	0	3592	0	0	167.12
BK-D1.1	80	30	7	(2.7)	41979	12431.4	48885	233	0	0	0.56
BK-D2.1	80	30	7	(23.1)	7572	4795.8	22990	225	43	0	0.24
BK-D3.1	80	30	7	(21.4)	6194	4812.1	11618	228	0	0	0.27
BK-D4.1	80	30	7	(16.2)	11500	3649.3	19600	223	0	0	0.16
BK-D5.1	80	30	7	13923.0	13129	3368.1	10747	228	10	0	0.56
BK-D6.1	80	30	7	(18.4)	12580	2522.9	24065	232	84	0	0.18
BK-D7.1	80	30	7	(24.4)	11989	3879.7	39393	228	69	1	0.18
BK-D8.1	80	30	7	(10.2)	12822	2656.6	22221	232	91	0	0.18
BK-D9.1	80	30	7	(17.8)	13289	3662.1	24026	233	0	0	0.17
BK-D10.1	80	30	7	(23.0)	10621	5974.3	28001	229	87	0	0.21
CLSA1032	100	100	25	(47.7)	22639	(43.3)	16286	114	6	0	2.35
CLSB1131	100	100	25	(49.3)	12844	(48.6)	14714	92	7	0	3.19
CLSC1133	100	100	25	(53.1)	17733	(52.8)	12566	27	0	0	1.79
FPP11-1	133	133	33	(79.7)	1179	(78.4)	1405	526	2	0	8.00
FPP11-10	133	133	33	(79.8)	243	(79.7)	116	658	5	13	10.33
FPP11-30	133	133	33	(79.9)	1511	(79.4)	2534	307	1	0	7.55
GR-50.1	50	50	12	189.2	659	701.0	1328	163	47	2	0.19
GR-50.2	50	50	12	74.6	263	61.4	233	26	23	79	0.25
GR-100.1	100	100	25	(8.8)	1114	(12.3)	957	447	125	1	3.78
GR-100.2	100	100	25	(13.2)	682	(16.3)	268	519	86	32	3.92

Table 1. Computational results over a set of real-life instances of OpM ($p = \lfloor m/4 \rfloor$). Under the "cpu1" and "cpu2" columns we report the running time of the two algorithms if the instance is solved to optimality, otherwise the gap w.r.t. the best upper bound is given in brackets.

The best results are obtained for $p = \lfloor m/4 \rfloor$ (see Table 1). For the 80×30 BildeKrarup (BK) instances, the result is more apparent: virtually all such instances are solved within two hours while Cplex alone reaches the time limit in many of them. The difference in performance is less remarkable in the Hoefer set of instances. Notice however that, unlike in BK instances, our branch-and-cut code generates less nodes and spends much more separation time, which results in a slightly better gap even though for Hoefer-P and Hoefer-Q instances no branch is performed. Instance Hoefer-O1 has been solved in a relatively short time, apparently due to the insertion of a large number of inequalities B which took a short separation time. The instances orlib, CLS and FPP do not differ so much in terms of branch-and-bound nodes, but our method still achieves a better result. For the Galvão-Raggi (GR) instances, instead, our method does better in one only instance out of four. It is worth pointing out that in many instances a better gap or running

				cplex		b&c					
Name	n	m	p	cpu1	nod1	cpu2	nod2	#A	#B	#C	sep
orlib-cap41	50	16	8	0.1	0	0.2	0	0	5	2	0.00
orlib-cap81	50	25	12	0.2	0	0.2	0	0	0	0	0.00
orlib-cap111	50	50	25	0.6	0	0.6	0	0	0	0	0.02
Hoefer-O1	100	100	50	(29.4)	5520	(28.7)	6585	0	41	37	5.84
Hoefer-O2	100	100	50	(28.5)	5631	(29.6)	5286	0	52	111	6.25
Hoefer-P1	200	200	100	(45.7)	24	(44.3)	24	0	131	72	62.68
Hoefer-P2	200	200	100	(45.3)	27	(38.8)	64	0	783	46	26.13
Hoefer-Q1	300	300	150	(47.1)	0	(47.1)	0	0	0	0	40.94
Hoefer-Q2	300	300	150	(47.2)	0	(47.2)	0	0	129	0	35.53
BK-D1.1	80	30	15	(16.1)	12100	(16.4)	11456	0	0	9	0.20
BK-D2.1	80	30	15	3701.0	9103	3167.0	7222	0	0	5	0.72
BK-D3.1	80	30	15	3001.9	9341	(7.3)	9593	0	0	7	0.20
BK-D4.1	80	30	15	(11.6)	12633	(12.7)	11268	0	3	9	0.21
BK-D5.1	80	30	15	3105.6	11302	3264.3	11265	0	2	16	0.23
BK-D6.1	80	30	15	(9.7)	14318	(10.9)	12702	0	21	13	0.20
BK-D7.1	80	30	15	(13.3)	13756	(14.6)	13494	0	1	10	0.26
BK-D8.1	80	30	15	(3.9)	13978	(6.9)	13362	0	0	16	0.23
BK-D9.1	80	30	15	(9.1)	14152	(9.9)	13165	0	0	12	0.22
BK-D10.1	80	30	15	(9.9)	13591	(12.1)	12794	0	4	13	0.21
CLSA1032	100	100	50	(74.1)	11150	(73.8)	11561	0	0	0	1.24
CLSB1131	100	100	50	(75.4)	8432	(75.4)	8480	0	0	0	2.15
CLSC1133	100	100	50	(68.5)	13551	(68.5)	13361	0	0	0	1.58
FPP11-1	133	133	66	(90.0)	6266	(88.5)	3639	0	0	9	4.42
FPP11-10	133	133	66	(89.7)	3671	(89.7)	3361	0	4	11	5.10
FPP11-30	133	133	66	(90.0)	3174	(88.5)	2800	0	1	2	5.72
GR-50.1	50	50	25	166.8	2414	171.3	2524	0	2	14	0.24
GR-50.2	50	50	25	88.8	112	63.6	128	0	33	32	0.25
GR-100.1	100	100	50	(8.4)	5169	(6.9)	2387	0	155	81	2.90
GR-100.2	100	100	50	(11.8)	6636	(10.5)	4122	0	193	57	3.72

Table 2. Computational results over a set of real-life instances of OpM ($p = \lfloor m/2 \rfloor$). Under the "cpu1" and "cpu2" columns we report the running time of the two algorithms if the instance is solved to optimality, otherwise the gap w.r.t. the best upper bound is given in brackets.

time has been achieved with a very short separation time, thus strengthening the model at almost no computational cost.

For $p = \lfloor m/2 \rfloor$ (see Table 2), the two algorithms have somehow similar performances, probably due to the limited generation of inequalities. In particular, no inequalities A are violated in any node of the decision tree, while more inequalities B and C are separated. This has a significant impact for the Hoefer instances, but does not improve the performance for the remaining ones. As shown in Table 3 for $p = \lfloor m/8 \rfloor$, our inequalities improve the gap for the Hoefer instances in general, but do not give a better result in other instances, and for the Bilke-Krarup ones even turn out to be less useful than the standard MILP cuts provided within Cplex.

From Tables 1 and 3 we notice that inequalities A prevail over the other two, although we separate them independently. This could be explained by the structure of the test instances, nevertheless we cannot conclude from these results which inequality is more effective. To this purpose, we have solved some instances with

				cplex		b&c					
Name	n	m	p	cpu1	nod1	cpu2	nod2	#A	#B	#C	sep
orlib-cap41	50	16	2	1.0	0	0.9	0	0	2	15	0.00
orlib-cap81	50	25	3	0.3	0	0.3	0	0	0	0	0.01
orlib-cap111	50	50	6	1.8	0	1.8	0	0	0	0	0.04
Hoefer-O1	100	100	12	(60.68)	121	(50.63)	92	914	0	0	1.97
Hoefer-O2	100	100	12	(59.67)	115	(49.70)	447	897	0	0	1.59
Hoefer-P1	200	200	25	(67.54)	2	(63.65)	0	1400	0	0	6.61
Hoefer-P2	200	200	25	(67.79)	7	(64.47)	0	1200	0	0	5.65
Hoefer-Q1	300	300	37	(70.09)	0	(69.60)	0	599	0	0	10.10
Hoefer-Q2	300	300	37	(70.01)	0	(69.10)	0	900	0	0	17.88
BK-D1.1	80	30	3	485.42	3825	2721.74	4029	160	8	235	0.19
BK-D2.1	80	30	3	604.16	2067	1989.63	2418	160	0	33	0.11
BK-D3.1	80	30	3	375.34	2069	1952.97	2301	165	17	245	0.13
BK-D4.1	80	30	3	377.95	2042	1467.45	2269	175	15	277	0.21
BK-D5.1	80	30	3	368.96	2029	1043.14	1909	160	0	77	0.15
BK-D6.1	80	30	3	309.62	1770	1904.95	2508	170	18	237	0.17
BK-D7.1	80	30	3	603.80	2325	2195.20	3274	170	20	220	0.10
BK-D8.1	80	30	3	564.09	1951	1374.30	2053	166	7	217	0.22
BK-D9.1	80	30	3	402.77	2065	1524.95	2457	172	5	271	0.24
BK-D10.1	80	30	3	463.44	2413	1550.87	2778	165	6	135	0.18
CLSA1032	100	100	12	(30.91)	2333	(30.60)	1680	296	0	0	1.37
CLSB1131	100	100	12	(39.21)	2000	(40.47)	1798	308	0	0	1.41
CLSC1133	100	100	12	(40.75)	2160	(40.96)	2154	138	0	0	1.40
FPP11-1	133	133	16	(64.74)	321	(64.68)	131	804	0	0	4.38
FPP11-10	133	133	16	(64.65)	349	(64.66)	200	722	0	0	4.45
FPP11-30	133	133	16	(64.65)	282	(64.70)	221	634	4	0	4.15
GR-50.1	50	50	6	208.44	510	147.60	282	50	10	174	0.26
GR-50.2	50	50	6	0.27	0	0.26	0	0	0	0	0.01
GR-100.1	100	100	12	(2.10)	1183	(5.55)	544	0	3	1544	4.20
GR-100.2	100	100	12	(7.51)	856	(3.29)	870	98	24	2676	7.73

Table 3. Computational results over a set of real-life instances of OpM ($p = \lfloor m/8 \rfloor$). Under the "cpu1" and "cpu2" columns we report the running time of the two algorithms if the instance is solved to optimality, otherwise the gap w.r.t. the best upper bound is given in brackets.

$p = \lfloor m/4 \rfloor$, separating each inequality alone, and evaluated the corresponding running time or gap, with a time limit of one hour. The results are reported in Table 4. For instance GR-100.1 and GR-50.1 inequality B makes the difference, although inequalities C, separated in relatively short time, solve instances GR-50.1 and GR-50.2 in comparable time. Remarkably, inequalities A, which are separated extensively in the first test, show to be useful only for instance CLSA1032.

Table 4. Comparison of the valid inequalities.

From a modeling standpoint, it is worth noting that for some instances the optimal solution is a set of facilities spread unevenly on the geographic area, in a corner or at the boundaries of the region, as in the left part of Figure 2, where the

optimal solution of a randomly-generated instance with $(n, m, p) = (20, 40, 8)$ is shown. This is justified by an objective function that takes into no account other practical needs, such as transportation costs or reachability of the open locations, which are dealt with by dispersion models. Considering clients lying outside the region may change the solution but does not avoid clustering of the open facilities, as shown in the right part of Figure 2, with 53 new clients surrounding the initial area.

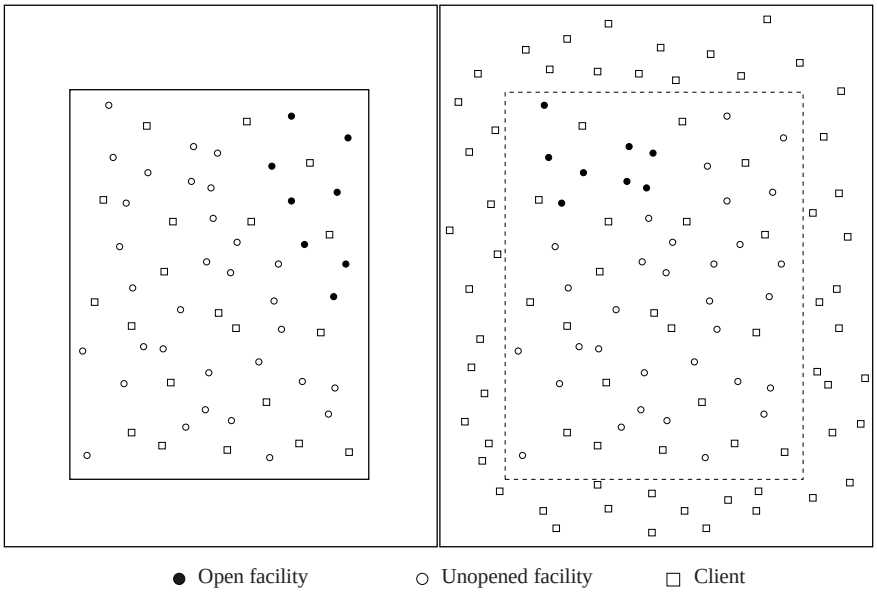


Fig. 2. An optimal solution of two instances of O*p*M. 20 clients are placed in the left part, and the obnoxious p-median objective function yields a *clustered* solution where all open facilities are in the upper-right corner. Considering 53 more clients around the instance obtains a different, but still clustered, solution.

6 Concluding remarks

In this work, we study an obnoxious location problem where all locations are fully obnoxious, and the average distance from each client to the nearest open facility is maximised. This problem is found in those contexts where facility dispersion is not necessary because of limited inter-facility nuisance and/or negligible transportation cost.

We have presented three families of valid inequalities, whose separation problem is polynomial, and developed an efficient exact procedure to tackle mid-size

instances in limited time. This exact method, coupled with a fast tabu search procedure, helps computing the optimal solution more rapidly than with a general-purpose Mixed Integer Programming software. Although we have tuned our separation and our branch-and-cut algorithms to this specific problem, the valid inequalities we have introduced are rather general and can be applied in different location problems.

Acknowledgments. The authors wish to thank two anonymous referees for useful comments and suggestions.

References

Alp O, Drezner Z, Erkut E (2003) An efficient genetic algorithm for the p-Median problem, *Annals of Operations Research* 122 (1-4): 21-42

Beasley JE (1990) OR-Library: distributing test problems by electronic mail, *Journal of the Operational Research Society* 41(11):1069-1072

Belotti P, Labbé M, Maffioli F (2000) A Tabu Search approach to the Obnoxious p-Median problem. Internal report 2000.49, DEI, Politecnico di Milano

Burkard RE, Çela E, Dollani H (2000) 2-Medians in Networks with Pos/Neg Weights. *Discrete Applied Mathematics* 105:51–71

Cappanera P, Gallo G, Maffioli F (1999) Discrete Facility Location and Routing of Obnoxious Activities. *Discrete Applied Mathematics* 133:3–28

Carrizosa E, Conde E (2002) A fractional model for locating semi-desirable facilities on networks. *European Journal of Operational Research* 136:67–80

Chandrasekaran R, Daughety A (1981) Location on tree networks: p-centre and n-dispersion problems. *Mathematics of Operations Research* 6:50–57

Chaudhry SS, McCormick ST, Moon ID (1986) Locating Independent Facilities with Maximum Weight: Greedy Heuristics. *Omega International Journal of Management Science* 14:383–389

Chhajed D, Lowe TJ (1994) Solving structured multifacility location problems efficiently. *Transportation Science* 28:104–115

Church RL, Garfinkel RS (1978), Locating an obnoxious facility on a network. *Transportation Science* 12:107–118

Dell'Amico M, Lodi A, Maffioli F (1999) Solution of the Cumulative Assignment Problem With a Well-Structured Tabu Search Method. *Journal of Heuristics* 5(2):123–143

Drezner Z, Hamacher H (eds) (2002) *Facility Location: Applications and Theory*. Springer Verlag, Berlin

Eiselt HA, Laporte G (1995) Objectives in Location Problems. In: Drezner Z (ed), *Facility Locations: a survey of Applications and Methods*. Springer Series in OR, New York, 151–180

Erkut E, Neuman S (1989) Analytical Models for Locating Undesirable facilities. *European Journal of Operational Research* 40:275–291

Erkut E (1990) The discrete p-dispersion problem. *European Journal of Operational Research* 46:48–60

Erkut E, Neumann S (1990) Comparison of four models for dispersing facilities. *INFOR* 29:68–85

Erkut E, Baptie T, Von Hohenbalken B (1990) The Discrete p-Maxian Location Problem. *Computers and Operations Research* 17(1):51–61.

Galvão RD, ReVelle C (1996) Lagrangean heuristic for the maximal covering location problem, European Journal of Operations Research 88:114-123

Hansen P, Moon ID (1988) Dispersing Facilities on a Network. *TIMS/ORSA Joint National Meeting* Washington D.C.

Kincaid R (1992) Good Solutions to Discrete Noxious Location Problems. *Annals of Operations Research* 40:265–281

Klein CM, Kincaid RK (1994) The Discrete Anti-p-Center Problem. *Transportation Science* 28:77–79.

Koerkel M (1989) On the exact solution of large-scale simple plant location problems, European Journal of Operations Research 39:157-173

Kuby MJ (1987) Programming models for facility dispersion: the p-dispersion and maxisum dispersion problems. *Geographical Analysis* 9:315–329.
Minieka E (1983) Anticenters and Antimedians of a Network. *Networks* 13:359–364.
Moon ID, Chaudhry SS (1983) An analysis of Network Location Problems with Distance Constraints. *Management Science* 30:290–307
Naddef D, Rinaldi G (2000) Branch-and-cut Algorithms. In: Toth P, Vigo D (eds), *Vehicle Routing*, SIAM
Shier DR (1977) A min-max theorem for p-center problems on a tree. *Transportation Science* 11(3):243–252.
Tamir A (1991) Obnoxious Facility Location on Graphs. *SIAM Journal on Discrete Mathematics* 4:550–567
Ting SS (1988) *Obnoxious Facility Location Problems on Networks*, PhD thesis, The Johns Hopkins University
Welch SB, Salhi S (1997) The Obnoxious p-Facility Network Location Problem with Facility Interaction. *European Journal of Operational Research* 102:302–319