Computers & Operations Research 39 (2012) 2625–2633

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

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

A computational study of a nonlinear minsum facility location problem

Emilio Carrizosa [a,1], Anton Ushakov [b], Igor Vasilyev [b,*]

[a] Instituto de Matemáticas de la Universidad de Sevilla, Universidad de Sevilla, Tarfia s/n, 41012 Sevilla, Spain
[b] Institute for System Dynamics and Control Theory, Siberian Branch of Russian Academy of Sciences, Lermontov Str., 134, 664033 Irkutsk, Russia

ARTICLE INFO

Available online 31 January 2012

Keywords:
p-median problem
Lagrangean relaxation
Core heuristic
Nonlinear integer programming

ABSTRACT

A discrete location problem with nonlinear objective is addressed. A set of p plants is to be open to serve a given set of clients. Together with the locations, the number p of facilities is also a decision variable. The objective is to minimize the total cost, represented as the transportation cost between clients and plants, plus an increasing nonlinear function of p.

Two Lagrangean relaxations are considered to derive lower bounds. Dual information is also used to design a core heuristic.

Computational results are given, showing that nearly optimal solutions are obtained in short running times.

1. Introduction

In this paper we address a nonlinear variant of the well-known p-median problem [1–3]. Let $I=\{1,\dots,n\}$ be a set of potential facility sites, where facilities can be opened, let $J=\{1,\dots,m\}$ be a set of clients, whose demands need to be satisfied by the facilities, and let $d_{ij}\geq 0$ be the cost of satisfying the demand of client $j\in J$ from the facility located at site $i\in I$. In the p-median problem, the location of p sites (medians) from I are sought in order to minimize the overall total cost of satisfying the demands of all clients. In other words, the p-median problem is the following combinatorial optimization problem:

$$\min_{S\subseteq I}\left\{\sum_{j\in J}\min_{i\in S} d_{ij}:\ |S|=p\right\}.$$

Different exact and heuristic algorithms have been proposed in the literature to address this NP-hard problem, see, e.g. [4–8] and the references therein, and it has been successfully applied for locating different types of facilities, both in the public and in the private sectors [9]. Moreover, the fields of applications go beyond location planning. Indeed, the p-median model has also been suggested as a clustering method in Data Analysis, e.g. [10–12]. In this context, the set I of clients is a set of entities (records in a data base), $J=I$, d_{ij} is some distance measure between entities i and j, and the goal is to find a set of p entities, so-called prototypes, so that the sum of distances between entities and their closest prototype is minimized. We also find p-median type models in the so-called Optimal Diversity Management problem [13,14], appearing in some industries: different configurations can be produced; a given configuration can be replaced by a more complete but more expensive one, and the set of p configurations to be produced minimizing the total overcost is sought.

A key ingredient of the p-median problem is that the number p of facilities is assumed to be fixed. This assumption may be rather unrealistic in several contexts. Indeed, the number of facilities to open may not be fixed when the facilities are relatively cheap objects to be bought (think, for instance, of location problems in which the facilities are garbage containers, post or phone boxes). The same may happen in Clustering problems or Optimal Diversity Management problems, in which a certain degree of arbitrariness exists with respect to the number of prototypes (respectively configurations) to be chosen, e.g. [15]. In these cases, it seems natural to address a minsum p-median type problem, in which the number p of facilities to be open is considered to be another decision variable, allowed to vary within the set $P\triangleq\{1,\dots,n\}$. The total cost to be minimized is then modeled as the sum of the transportation cost, namely, the p-median objective, plus a penalty $\phi(p)$. In other words, the problem to be addressed, and analyzed in this paper, is written as follows:

$$\min_{S\subseteq I,p\in P}\left\{\sum_{j\in J}\min_{i\in S} d_{ij}+\phi(p):\ |S|=p\right\}. \tag{1}$$

The only assumption made on ϕ is that it is increasing in p, so that the number of facilities selected (or clusters designed, or configurations produced) is as small as possible. If ϕ is linear, then (1) is the particular case of the well-known uncapacitated facility location problem [16], with common fixed cost for all facilities. Moreover, for particular choices of ϕ one obtains that Problem (1) is reduced to the classical p-median problem for fixed $p=p^*$. Indeed, let $Z(p^*)$ denote the optimal value of the p-median problem with p^* facilities,

* Corresponding author. Tel.: +7 9148752836; fax: +7 3952511616.
E-mail addresses: ecarrizosa@us.es (E. Carrizosa), aushakov@icc.ru (A. Ushakov), vil@icc.ru (I. Vasilyev).
[1] The research of E. Carrizosa is supported by projects MTM2009-14039-C06-06, FQM329.

and take any function ϕ such that $\phi(p)=0$ for all $p \le p^*$ and $\phi(p) > Z(p^*)$ for all $p > p^*$. Since the distances d_{ij} are non-negative, then it turns out that Problem (1) has an optimal solution with exactly p^* facilities, and it is also a p-median solution of $p=p^*$.

Problem (1) with a nonlinear penalty function ϕ has been introduced in [17] to conveniently model distribution problems with (dis)economies of scale. Nonlinear penalty functions are also more reasonable in Cluster problems, in which a penalty function of type $\phi(p)=Cp^2$, $(C>0)$ or $\phi(p)=p\log p$ may be more adequate to prevent the number of clusters to exploit. Similarly, in the Optimal Diversity Management context, the cost of producing, storing and handling different configurations may increase very rapidly with the number of configurations.

Despite being a very natural variant of the p-median problem, this problem has received little attention in the literature. As far as the authors know, the model has been introduced in [17], where the case of convex ϕ is addressed. A heuristic algorithm is designed which, by bisection in p, reduces the problem to a series of uncapacitated facility location or p-median problems. Limited computational experience is given, as it is also given in [18], where the convexity assumption on ϕ is relaxed. The reader is referred to, e.g. [19–25] and the references therein for other nonlinear optimization problems arising in discrete facility location.

In this paper we describe a very sharp heuristic for the integer nonlinear program (1) with two main elements: Lagrangean relaxation is used to obtain lower bounds, and in a second stage, a core selection approach [26,4,5,27], is derived to obtain very good upper bounds. The remainder of the paper is structured as follows. The problem is stated in Section 2. In Section 3 two different Lagrangean relaxations are proposed, and details on how to optimize them via a subgradient algorithm are given. The dual information provided by the Lagrange multipliers is used to design a core heuristic. Combining lagrangian relaxation with the core heuristic yields our algorithm, as described in Section 4.

Computational experience on data sets taken from the OR and from the data analysis literature is reported in Section 5. Finally, some conclusions and future perspectives are discussed in Section 6.

2. Problem statement

Problem (1) is formulated as a nonlinear problem in integer variables by modifying the usual IP formulation for the p-median problem. For each $i \in I$, let us consider a binary variable y_i, which takes the value 1 if the facility at i is open, and takes the value 0 otherwise, and, for each pair $i \in I, j \in J$, let x_{ij} be the binary variable which is equal to 1 if client j is served by facility at i. With this notation, Problem (1) is written as the following nonlinear integer problem:

$$Z^* = \min_{(x,y,p)} \sum_{i \in I}\sum_{j \in J} d_{ij}x_{ij} + \phi(p), \tag{2}$$

$$\sum_{i \in I} x_{ij} = 1, \quad j \in J, \tag{3}$$

$$x_{ij} \le y_i, \quad i \in I, \quad j \in J, \tag{4}$$

$$\sum_{i \in I} y_i = p, \tag{5}$$

$$y_i \in \{0,1\}, \quad i \in I, \tag{6}$$

$$x_{ij} \in \{0,1\}, \quad i \in I, j \in J, \tag{7}$$

$$p \in P. \tag{8}$$

Constraints (3) ensure that each client j is served by exactly one facility. Constraints (4) impose that a client can only be served by open facilities. Constraint (5) enforces the number of facilities to be $p \in P = \{p \in \mathbb{Z}: \ 1 \le p \le n\}$.

Since p takes only a finite set of values, an equivalent linear integer programming formulation for (1) is obtained by linearizing the function ϕ. Indeed, for each $k \in P$, let $\phi_k = \phi(k)$, and let z_k be the binary variable which takes the value 1 if exactly k facilities are open and 0 otherwise. With this notation, we obtain the equivalent IP problem:

$$Z^* = \min_{(x,y,z)} \sum_{i \in I}\sum_{j \in J} d_{ij}x_{ij} + \sum_{k=1}^{n} \phi_k z_k, \tag{9}$$

$$\sum_{i \in I} x_{ij} = 1, \quad j \in J, \tag{10}$$

$$x_{ij} \le y_i, \quad i \in I, \quad j \in J, \tag{11}$$

$$\sum_{i \in I} y_i = \sum_{k=1}^{n} k z_k, \tag{12}$$

$$\sum_{k=1}^{n} z_k = 1, \tag{13}$$

$$y_i \in \{0,1\}, \quad x_{ij} \in \{0,1\}, \quad i \in I, \ j \in J, \tag{14}$$

$$z_k \in \{0,1\}, \quad k = 1, \ldots, n. \tag{15}$$

Observe that constraint (12) ensures the number of open facilities is equal to k, whereas constraint (13) enforces only one value $k=1,\ldots,n$ to be chosen for the number of open facilities.

We are interested in solving problems of large size. Hence, exact methods guaranteeing that an optimal solution have been obtained are not applicable. Instead, we develop a heuristic which, as shown in the computational results presented in Section 5, performs very well in practice. First, a Lagrangean relaxation scheme is used to Problem (2)–(8). This way, a lower bound of the optimal value Z^*, as well as an upper bound (given by the objective evaluated at a heuristic solution) are obtained. The gap between the lower and upper bounds is later reduced by using a so-called core heuristic: the information provided by the Lagrange multipliers is used to select a subset of "promising" variables; then a reduced p-median problem, i.e. the p-median problem for just the subset of selected variables, is solved with commercial software using the formulation (9)–(15). We explain in Sections 3.1–3.4 how Lagrangean functions are computed and optimized. Then, we devote Section 3.5 to show how this dual information can be used in the core heuristic to obtain heuristic solutions with smaller gap.

3. Lagrangean relaxation and the core heuristic

Methods based on the Lagrangean relaxation have been widely used for solving location problems [6,28,26,11].

In this section we define two different Lagrangean relaxations of Problems (2)–(8), extending well-known relaxations of the p-median problem to the case in which p is also a decision variable. These relaxations, called hereafter $\mathcal{L}_1$ and $\mathcal{L}_2$, correspond to the case in which constraints (3) and (5) (respectively constraints (3)) are relaxed.

Relaxing constraints (3) and (5) with Lagrange multipliers $\lambda \in \mathbb{R}^m$ and $\pi \in \mathbb{R}$ respectively we obtain the Lagrangean dual function $\mathcal{L}_1(\lambda,\pi)$:

$$\mathcal{L}_1(\lambda,\pi) = \min_{(x,y,p)} \left\{ \sum_{i \in I}\sum_{j \in J} d_{ij}x_{ij} + \sum_{j \in J}\lambda_j\left(1 - \sum_{i \in I} x_{ij}\right) + \pi\left(p - \sum_{i \in I} y_i\right) + \phi(p) : x_{ij} \le y_i \right\}$$

$$= \min_{(x,y,p)} \left\{ \sum_{i \in I}\sum_{j \in J} (d_{ij} - \lambda_j)x_{ij} + \pi p - \pi \sum_{i \in I} y_i + \phi(p) : x_{ij} \le y_i \right\} + \sum_{j \in J} \lambda_j.$$

All variables x_{ij}, y_i are binary and $p \in P$. For simplicity in the notation we skip such constraints.

In the same way, if only constraints (3) are relaxed, we obtain the Lagrangean dual function $\mathcal{L}_2(\lambda)$:

$$\mathcal{L}_2(\lambda) = \min_{(x,y,p)} \left\{ \sum_{i \in I} \sum_{j \in J} d_{ij} x_{ij} + \sum_{j \in J} \lambda_j \left(1 - \sum_{i \in J} x_{ij} \right) + \phi(p) : x_{ij} \le y_i, : \sum_{i \in I} y_i = p \right\}$$

$$= \min_{(x,y,p)} \left\{ \sum_{i \in I} \sum_{j \in J} (d_{ij} - \lambda_j) x_{ij} + \phi(p) : x_{ij} \le y_i, : \sum_{i \in I} y_i = p \right\} + \sum_{j \in J} \lambda_j.$$

Implementation details on how to optimize $\mathcal{L}_1(\lambda,\pi)$ and $\mathcal{L}_2(\lambda)$ will be given in Section 3.4. Before, we will analyze in Sections 3.1 and 3.2 how to compute $\mathcal{L}_1(\lambda,\pi)$ and $\mathcal{L}_2(\lambda)$ when local convexity properties of the function $\phi(\cdot)$ can be exploited. We recall that function $\phi(\cdot)$ is said to be convex (respectively concave) on a grid $K = \{k_1, k_1+1, \ldots, k_2-1, k_2\}$ if $\phi(\cdot)$ has nondecreasing (respectively nonincreasing) increments. In other words, $\phi(\cdot)$ is said to be convex (respectively concave) on the grid K if $\Delta\phi(\cdot)$ is a nondecreasing (respectively nonincreasing) function, where $\Delta\phi(k) \triangleq \phi(k+1) - \phi(k)$. It is easily seen that $\phi(\cdot)$ is convex (respectively concave) in $\{k_1, k_1+1, \ldots, k_2-1, k_2\}$ iff the piecewise linear function with knots in K and interpolating ϕ in K is convex (respectively concave) in the interval $[k_1, k_2]$.

3.1. Evaluating the Lagrangean function $\mathcal{L}_1$

Let us compute the value of the Lagrangean dual function $\mathcal{L}_1(\lambda,\pi)$ for a given set of multipliers (λ,π). Using the notation $a^- = \min\{0,a\}$, it follows that $\mathcal{L}_1(\lambda,\pi)$ is written as the sum of one linear term in the multipliers λ, one concave piecewise linear term in the multiplier π and the coupling term $\sum_{i \in I} [\sum_{j \in J} (d_{ij} - \lambda_j)^- \pi]^-$:

Proposition 1. *One has*

$$\mathcal{L}_1(\lambda,\pi) = \sum_{i \in I} \left[\sum_{j \in J} (d_{ij} - \lambda_j)^- \pi \right]^- + \min_p \{\pi p + \phi(p)\} + \sum_{j \in J} \lambda_j.$$

Moreover, a Lagrangean solution $(x(\lambda,\pi), y(\lambda,\pi), p(\pi))$ is obtained as

$$y_i(\lambda,\pi) = \begin{cases} 1, & (d_{ij} - \lambda_j)^- \pi < 0, \\ 0, & (d_{ij} - \lambda_j)^- \pi \ge 0, \end{cases}$$

$$x_{ij}(\lambda,\pi) = \begin{cases} 1, & y_i(\lambda,\pi) = 1 \text{ and } d_{ij} - \lambda_j < 0 \\ 0 & \text{otherwise}, \end{cases}$$

$$p(\pi) \in \operatorname*{Argmin}_{q \in P} \{\pi q + \phi(q)\}.$$

Proof. One has

$$\mathcal{L}_1(\lambda,\pi) = \min_y \left\{ \min_x \left\{ \sum_{i \in I} \sum_{j \in J} (d_{ij} - \lambda_j) x_{ij} : \ x_{ij} \le y_i \right\} - \pi \sum_{i \in I} y_i \right\} + \min_p \{\pi p + \phi(p)\} + \sum_{j \in J} \lambda_j.$$

For fixed y, an optimal solution of

$$\min_x \left\{ \sum_{i \in I} \sum_{j \in J} (d_{ij} - \lambda_j) x_{ij} : \ x_{ij} \le y_i \right\}$$

is given by

$$x^*_{ij} = \begin{cases} y_i, & d_{ij} - \lambda_j < 0, \\ 0, & d_{ij} - \lambda_j \ge 0, \end{cases}$$

In this case the Lagrangean dual function becomes

$$\mathcal{L}_1(\lambda,\pi) = \min_y \left\{ \sum_{i \in I} \left[\sum_{j \in J} (d_{ij} - \lambda_j)^- \pi \right] y_i \right\} + \min_p \{\pi p + \phi(p)\} + \sum_{j \in J} \lambda_j$$

$$= \sum_{i \in I} \left[\sum_{j \in J} (d_{ij} - \lambda_j)^- \pi \right]^- + \min_p \{\pi p + \phi(p)\} + \sum_{j \in J} \lambda_j.$$

This implies that, for λ,π fixed, an optimal solution $(x(\lambda,\pi), y(\lambda,\pi), p(\pi))$ is obtained as

$$y_i(\lambda,\pi) = \begin{cases} 1, & (d_{ij} - \lambda_j)^- \pi < 0 \\ 0, & (d_{ij} - \lambda_j)^- \pi \ge 0; \end{cases}$$

$$x_{ij}(\lambda,\pi) = \begin{cases} 1, & y_i(\lambda,\pi) = 1 \text{ and } d_{ij} - \lambda_j < 0 \\ 0 & \text{otherwise}; \end{cases}$$

$$p(\pi) \in \operatorname*{Argmin}_{q \in P} \{\pi q + \phi(q)\}. \qquad \square$$

Proposition 1 shows how to obtain, for fixed multipliers λ,π, optimal $x(\lambda,\pi)$, $y(\lambda,\pi)$. Note that in order to find $p(\pi)$, one needs to solve

$$\min_{p \in P} \{\pi p + \phi(p)\}. \tag{16}$$

Problem (16) can be solved by full inspection of the set P. The following property, with straightforward proof, may be of help to avoid complete enumeration.

Proposition 2. *For a given $\pi \in \mathbb{R}$, define the function $\phi_\pi : P \mapsto \mathbb{R}$ as $\phi_\pi(p) = \phi(p) + \pi p$. One has*

1. *If $\pi \ge \max_{k<n} -\Delta\phi(k)$, then $\phi_\pi(\cdot)$ is nondecreasing in P, and thus an optimal solution of (16) is $p(\pi) = 1$.*
2. *If $\pi \le \min_{k<n} -\Delta\phi(k)$, then $\phi_\pi(\cdot)$ is nonincreasing in P, and thus an optimal solution of (16) is $p(\pi) = n$.*
3. *A necessary condition for $k_0 \in P$ to be optimal for (16) is that* $$\Delta\phi(k_0 - 1) \le -\pi \le \Delta\phi(k_0). \tag{17}$$
4. *If $\phi(\cdot)$ is concave in P, then $\phi_\pi(\cdot)$ is also concave, and thus an optimal solution of (16) is an endpoint, i.e. $p(\pi) \in \{1, n\}$.*
5. *If $\phi(\cdot)$ is convex in P, then condition (17) is also necessary for optimality, and it can be checked by doing a binary search in P.*

3.2. Evaluating the Lagrangean function $\mathcal{L}_2$

Let us now consider the second relaxation, in which only constraints 3 are relaxed to obtain the Lagrangean $\mathcal{L}_2(\lambda)$.

For each $i \in I$, define $\rho_i(\lambda)$ as the Lagrangean reduced cost of the corresponding variable y_i,

$$\rho_i(\lambda) \triangleq \sum_{j \in J} (d_{ij} - \lambda_j)^-.$$

Moreover, let $v(\lambda) = (v_1(\lambda), v_2(\lambda), \ldots, v_n(\lambda))$ be a permutation of the sites increasingly ordering their reduced costs, i.e.

$$\rho_{v_1(\lambda)}(\lambda) \le \rho_{v_2(\lambda)}(\lambda) \le \cdots \le \rho_{v_n(\lambda)}(\lambda).$$

Define $\psi(p,\lambda)$ as

$$\psi(p,\lambda) \triangleq \sum_{k=1}^{p} \rho_{v_k(\lambda)}(\lambda).$$

The following properties are useful to evaluate $\mathcal{L}_2(\lambda)$.

Proposition 3. *One has*

1. *$\mathcal{L}_2(\lambda) = \min_{p \in P} \{\psi(p,\lambda) + \phi(p)\} + \sum_{j \in J} \lambda_j$.*
2. *$\psi(\cdot,\lambda)$ is nonincreasing and convex.*

3. *$\psi(p,\lambda)+\phi(p)$ can be computed recursively as follows:*

$$\psi(1,\lambda)+\phi(1)=\rho_{\nu_1(\lambda)}(\lambda)+\phi(1),$$

$$\psi(p,\lambda)+\phi(p)=\psi(p-1,\lambda)+\rho_{\nu_p(\lambda)}(\lambda)+\phi(p)\quad \forall p=2,\dots,n.$$

Proof. Using the same reasoning than for $\mathcal{L}_1(\lambda,\pi)$, $\mathcal{L}_2(\lambda)$ can be expressed for a given λ as follows:

$$\mathcal{L}_2(\lambda)=\min_{(y,p)}\left\{\min_x\left\{\sum_{i\in I}\sum_{j\in J}(d_{ij}-\lambda_j)x_{ij}:\ x_{ij}\le y_i\right\}+\phi(p):\ \sum_{i\in I}y_i=p\right\}$$

$$+\sum_{j\in J}\lambda_j=\min_{(y,p)}\left\{\sum_{i\in I}\left[\sum_{j\in J}(d_{ij}-\lambda_j)^-\right]y_i+\phi(p):\ \sum_{i\in I}y_i=p\right\}+\sum_{j\in J}\lambda_j.$$

Hence,

$$\mathcal{L}_2(\lambda)=\min_p\left\{\sum_{k=1}^{p}\rho_{\nu_k(\lambda)}(\lambda)+\phi(p)\right\}+\sum_{j\in J}\lambda_j$$
$$=\min_{p\in P}\{\psi(p,\lambda)+\phi(p)\}+\sum_{j\in J}\lambda_j.$$

Convexity of $\psi(\cdot,\lambda)$ directly follows from the fact that $\Delta\psi(p,\lambda)\triangleq\psi(p+1,\lambda)-\psi(p,\lambda)=\rho_{\nu_{p+1}(\lambda)}(\lambda)\le 0$. This also shows that $\psi(\cdot,\lambda)$ is nonincreasing.

Part 3 is straightforward from the definition of $\psi(p,\lambda)$. $\square$

Proposition 3 shows that the value of $\mathcal{L}_2(\lambda)$ can be computed by full inspection of the values $\psi(p,\lambda)+\phi(p)$, which can be recursively computed. In order to efficiently evaluate $\mathcal{L}_2(\lambda)$, we have adapted to our problem the technique of "delayed column generation" in Avella et al. [4]. As a preprocessing step, the distance matrix is computed, and each column $j\in J$ is sorted in a nondecreasing way, i.e. one has for each $j\in J$ a permutation $u(j)$ such that

$$d(u_1(j),j)\le d(u_2(j),j)\le\cdots\le d(u_n(j),j).$$

Having done this, $\mathcal{L}_2(\lambda)$ and the associated reduced costs can be efficiently computed by the Scheme 1.

Note that a similar scheme can be designed for computing $\mathcal{L}_1(\lambda,\pi)$.

The computation of $\mathcal{L}_2^*$ in the algorithm above can be done by complete enumeration. However, as for $\mathcal{L}_1(\lambda,\pi)$, if $\phi(\cdot)$ has some structure, complete enumeration in P may be avoided. This issue is addressed in the following propositions.

Proposition 4. *If $\rho_{\nu_1(\lambda)}(\lambda)=0$, then*

$$\mathcal{L}_2(\lambda)=\rho_{\nu_1(\lambda)}(\lambda)+\phi(1)+\sum_{j\in J}\lambda_j.$$

Otherwise, the set $P_0(\lambda)\triangleq\{k\in P:\ \rho_{\nu_k(\lambda)}(\lambda)<0\}\ne\varnothing$, and then

$$\mathcal{L}_2(\lambda)=\min_{p\in P_0}\{\psi(p,\lambda)+\phi(p)\}+\sum_{j\in J}\lambda_j.$$

```
1. Initialization: $\rho(\lambda):=0$, $\mathcal{L}_2(\lambda):=0$ and $j:=1$;
2. Compute $\mathcal{L}_2(\lambda):=\mathcal{L}_2(\lambda)+\lambda_j$ and set $i:=1$;
3. If $d(u_i(j),j)-\lambda_j>=0$, then go to 6, else go to 4;
4. Compute $\rho_{u_i(j)}(\lambda):=\rho_{u_i(j)}(\lambda)+d(u_i(j),j)-\lambda_j$;
5. If $i<n$, then set $i:=i+1$ and go to 3, else go to 6;
6. If $j<m$, then set $j:=j+1$ and go to 2, else go to 7;
7. Compute $\mathcal{L}_2^*=\min_{p\in P}\{\psi(\lambda,p)+\phi(p)\}$;
8. Compute $\mathcal{L}_2(\lambda):=\mathcal{L}_2(\lambda)+\mathcal{L}_2^*$.
```

Scheme 1. Computing $\mathcal{L}_2(\lambda)$ and reduced costs.

Proof. If $\rho_{\nu_1(\lambda)}(\lambda)=0$, then $\psi(p,\lambda)=0$, $p\in P$. Since $\phi(p)$ is a nondecreasing function on P, it turns out that the function $pxmap;\psi(p,\lambda)+\phi(p)$ is nondecreasing in p, and thus the result is proved.

Suppose now $\rho_{\nu_1(\lambda)}(\lambda)<0$, and thus $P_0(\lambda)\ne\emptyset$. The result is straightforward for $P_0(\lambda)=P$, and we now show the proof for the case $P_0(\lambda)\ne P$. By definition of the permutation $\nu(\lambda)$ and the assumptions, $\exists\overline{p}\in P$ such that

$$\rho_{\nu_k(\lambda)}(\lambda)<0\quad\forall k=1,\dots,\overline{p};$$

$$\rho_{\nu_k(\lambda)}(\lambda)=0\quad\forall k=\overline{p}+1,\dots,n.$$

Let us show that

$$\psi(\overline{p},\lambda)+\phi(\overline{p})\le\psi(p,\lambda)+\phi(p),\quad\forall p\in P\backslash P_0.$$

It is clear that

$$\psi(p,\lambda)=\sum_{k=1}^{\overline{p}}\rho_{\nu_k(\lambda)}(\lambda)+\sum_{k=\overline{p}+1}^{p}\rho_{\nu_k(\lambda)}(\lambda)=\psi(\overline{p},\lambda)$$
$$+\sum_{k=\overline{p}+1}^{p}\rho_{\nu_k(\lambda)}(\lambda)=\psi(\overline{p},\lambda).$$

Since $\phi(\cdot)$ is an increasing function on P the proposition is proved. $\square$

The convexity of the function $\psi(\cdot,\lambda)$ yields the following result.

Proposition 5. *For $k_0\in\{2,3,\dots,n\}$, if $k_0\in\text{Argmin}_{p\in P}\{\psi(p,\lambda)+\phi(p)\}$, then*

$$\rho_{\nu_{k_0}(\lambda)}(\lambda)+\Delta\phi(k_0-1)\le 0\le\rho_{\nu_{k_0+1}(\lambda)}(\lambda)+\Delta\phi(k_0).$$

Moreover, if $\phi(\cdot)$ is a convex function, this condition is also sufficient to be k_0 in $\text{Argmin}_{p\in P}\{\psi(p,\lambda)+\phi(p)\}$.

Once $p(\lambda)\in\text{Argmin}_{p\in P}\{\psi(p)+\phi(p)\}$ is found, $(x(\lambda),y(\lambda))$ is obtained as

$$y_i(\lambda)=\begin{cases}1, & i\in\{\nu_1(\lambda),\dots,\nu_{p(\lambda)}(\lambda)\}\\ 0 & \text{otherwise;}\end{cases}$$

$$x_{ij}(\lambda)=\begin{cases}1, & y_i(\lambda)=1\text{ and }d_{ij}-\lambda_j<0\\ 0 & \text{otherwise.}\end{cases}$$

3.3. Comparing $\mathcal{L}_1$ and $\mathcal{L}_2$

Both $\mathcal{L}_1(\lambda,\pi)$ and $\mathcal{L}_2(\lambda)$ give lower bounds on the optimal value Z^* of Problem (2)–(8). Such bounds are related by the following property.

Proposition 6. *For $\lambda\in\mathbb{R}^m$ and $\pi\in\mathbb{R}$*

$$\mathcal{L}_1(\lambda,\pi)\le\mathcal{L}_2(\lambda)\le Z^*. \tag{18}$$

Proof. For λ,π fixed, let (LD1) and (LD2) denote the optimization problems associated with $\mathcal{L}_1(\lambda,\pi)$ and $\mathcal{L}_2(\lambda)$,

$$\mathcal{L}_1(\lambda,\pi)=\min_{(x,y,p)}\left\{\sum_{i\in I}\sum_{j\in J}d_{ij}x_{ij}+\sum_{j\in J}\lambda_j\left(1-\sum_{i\in I}x_{ij}\right)\right.$$
$$\left.+\pi\left(p-\sum_{i\in I}y_i\right)+\phi(p):\ x_{ij}\le y_i\right\} \tag{LD1}$$

$$\mathcal{L}_2(\lambda)=\min_{(x,y,p)}\left\{\sum_{i\in I}\sum_{j\in J}d_{ij}x_{ij}+\sum_{j\in J}\lambda_j\left(1-\sum_{i\in J}x_{ij}\right)\right.$$
$$\left.+\phi(p):\ x_{ij}\le y_i,:\ \sum_{i\in I}y_i=p\right\} \tag{LD2}$$

The right-hand side inequality is true by definition of relaxation. Let us show that $\mathcal{L}_1(\lambda,\pi)$ is a relaxation of $\mathcal{L}_2(\lambda)$, whence validity of left-hand side inequality follows. For this we need to show that

1. The feasible region of (LD2) is at most as large as the feasible region of (LD1). This holds because
$$\left\{(x,y,p): \ x_{ij} \le y_i, \quad \sum_{i\in I} y_i = p\right\} \subseteq \{(x,y,p): \ x_{ij} \le y_i\}.$$
2. For any feasible solution of (LD2) the objective function of (LD1) is smaller than or equal to the objective function of (LD2). Indeed, let (x',y',p') be a feasible solution in (LD2), then $(p'-\sum_{i\in I} y'_i)=0$. Since (x',y',p') is also feasible in (LD1), we have
$$\sum_{i\in I}\sum_{j\in J} d_{ij}x'_{ij} + \sum_{j\in J}\lambda_j\left(1-\sum_{i\in I}x'_{ij}\right)+\phi(p')$$
$$= \sum_{i\in I}\sum_{j\in J} d_{ij}x'_{ij} + \sum_{j\in J}\lambda_j\left(1-\sum_{i\in I}x'_{ij}\right)$$
$$+\pi\left(p'-\sum_{i\in I}y'_i\right)+\phi(p') \ge \mathcal{L}_1(\lambda,\pi).$$
Since it holds for all (x,y,p) feasible for (LD2),
$\mathcal{L}_1(\lambda,\pi) \le \mathcal{L}_2(\lambda) \quad \forall \lambda\in\mathbb{R}^m, \quad \forall \pi\in\mathbb{R}.$

3.4. Subgradient optimization

The Lagrangean relaxation is known to provide a lower bound for the optimal value of relaxed problem. For simplicity we only analyze here the Lagrangean dual function $\mathcal{L}_2(\lambda)$, obtained by relaxing constraints (3). To get the best lower bound, the Lagrangean dual function is maximized with respect to λ, i.e. the optimization problem $\max_{\lambda\in\mathbb{R}^m}\{\mathcal{L}_2(\lambda)\}$ is to be solved. A possible way to optimize the nonsmooth function $\mathcal{L}_2(\lambda)$ is via the subgradient method, which is based on the iterative formula:

$$\lambda^{k+1} = \lambda^k + \alpha_k g(\lambda^k). \tag{19}$$

In (19), $g(\lambda^k)$ is the subgradient at the iteration k, computed as

$$g_j(\lambda^k) = 1-\sum_{i\in I} x_{ij}(\lambda^k) \quad \forall j\in J,$$

and α_k is the stepsize, computed as

$$\alpha_k = \frac{\beta_k(1.05\cdot BUB-\mathcal{L}_2(\lambda^k))}{\|g(\lambda^k)\|_2^2},$$

where BUB is a current upper bound, provided for example by some upper bound heuristics; $\|\cdot\|_2$ is the euclidean metric; $\mathcal{L}_2(\lambda^k)$ is a value of the Lagrangean dual function; β_k is a parameter.

Let us present a scheme for computing a subgradient vector for a given Lagrange multipliers λ. Before the beginning of the main cycle one needs to compute vector $y(\lambda)$ and variable $p(\lambda)$, on which the minimum in step 7 of (1) is reached. Then a subgradient vector can be computed according Scheme 2.

The delayed column generation and the scheme above can be easily implemented. As pointed out by Avella et al. [4], their main disadvantage is the high memory consumption, since we need to store into the memory the whole distance matrix. However, for computing the Lagrangean dual function $\mathcal{L}_2$ and a subgradient, for each column $j\in J$ we need to consider only the first elements which are less than λ_j. Therefore for each $j\in J$ we keep into the memory only the first $n_0(j)<n$ elements. If, on any iteration of subgradient algorithm, the Lagrange multiplier λ_j^k exceeds $d(n_0(j),j)$, then the j-th column is expanded in the RAM, adding n_1 elements, i.e. $n_0(j) := n_0(j)+n_1$, and the subgradient algorithm continues.

```
1. Initialization: find a vector $y(\lambda)$:  $y_i := 1$,
   $\forall i \in \{v_1(\lambda),\dots,v_{p(\lambda)}(\lambda)\}$ and $y_i := 0$ otherwise, set $j := 1$;
2. Compute $g_j(\lambda) := 1$ and set $i := 1$;
3. If $d(u_i(j),j) - \lambda_j >= 0$, then go to 6, else go to 4;
4. If $y_{u_i(j)}(\lambda) = 1$, then set $g_j(\lambda) := g_j(\lambda) - 1$;
5. If $i < n$, then set $i := i+1$ and go to 3, else go to 6;
6. If $j < m$, then set $j := j+1$ and go to 2, else stop;
```

Scheme 2. Computing a subgradient vector for $\mathcal{L}_2$.

To prevent the oscillation of Lagrange multipliers, we have used the stabilization method suggested by Hansen et al. [11]. Initially multipliers are set at the smallest distance in the corresponding column, i.e. $\lambda_j^0 = d(v_1(j),j)$. Further, on each iteration we set upper bounds on the multipliers $\lambda_j^k \le ub_j^k$, where

$$ub_j^k = \min_{k\in I}\{d(u_k(j),j): \ d(u_k(j),j) > \lambda_j^{k-1}\}.$$

In other words, the multiplier cannot "jump" more than one value in the corresponding column of the distance matrix. In this way we prevent the explosion of the number of matrix elements (unnecessarily) stored in memory.

3.5. The core heuristic

To obtain an upper bound for the optimal value, we use the well-known core selection approach, based on choosing a subset of "promising' variables and then solving a reduced IP problem over them. Core selection is proved to be effective in solving Set Covering [27], Capacitated Facility Location [26] and p-median [5,4] problems. Let λ be the vector of Lagrange multipliers returned by the subgradient optimization and let $\rho_i(\lambda)$ and $q_{ij}(\lambda) = d_{ij}-\lambda_j$ be the reduced costs associated with variables y_i and x_{ij} respectively. The core problem is defined by the number $p(\lambda)$ of open facilities returned by the subgradient algorithm and by the subset of variables whose reduced cost is less than the given thresholds (ν for variables y and μ for variables x).

Let us remind that we use the formulation (9)–(15) for constructing a reduced IP problem, since there no further properties of $\phi(\cdot)$ are used. In other words, we solve the IP problem (9)–(15) containing only those variables y_i and x_{ij} satisfying

$$i\in I(\lambda) \triangleq \{i\in I: \ \rho_i(\lambda)\le \nu\},$$

$$ij\in W(\lambda) \triangleq \{ij: \ i\in I(\lambda), j\in J, q_{ij}\le \mu\}.$$

Needless to say, the higher the values of the threshold parameters ν,μ, the larger the size of the sets $I(\lambda),W(\lambda)$, and thus the larger the size of the IPs to be solved exactly in the core problem with a commercial solver. Finding values for such parameters yielding a good compromise between accuracy of the solution obtained and overall running times is pursued in our iterative algorithm, as discussed in the following.

4. The algorithm

As was shown (e.g. [4]) there are two main difficulties in the approach:

- Knowing a good upper bound is crucial to get a good convergence of the subgradient procedure,
- The core heuristic provides a good upper bound on a base of a good solution of subgradient optimization.

Avella et al. [4] suggested to sequentially repeat the subgradient optimization and solving the core problem. This approach is followed for our problem: we repeat the calls to the subgradient optimization routine and to the core procedure. The procedure stops after sufficiently many calls are made or when the error *Err*, defined as the relative difference between the best lower bounds and upper bounds, i.e. $Err=(BUB-BLB)/BUB\cdot 100\%$, becomes sufficiently small.

Let $Z(x,y,p)$ be a value of objective function (2) on a solution (x,y,p).

The overall procedure is summarized as follows:

1. Initialization: generate a random feasible solution $(\overline{x},\overline{y},\overline{p})$, set the best upper bound $BUB:=Z(\overline{x},\overline{y},\overline{p})$, the best lower bound $BLB:=-\infty$ and $h:=1$.
2. Find λ by the subgradient algorithm, set $BLB:=\mathcal{L}_2(\lambda)$.
3. Construct $I(\lambda)$, $W(\lambda)$ and find a solution $(\widehat{x},\widehat{y},\widehat{p})$ of the core problem. If $Z(\widehat{x},\widehat{y},\widehat{p})<BUB$, then set $BUB:=Z(\widehat{x},\widehat{y},\widehat{p})$ and $(\overline{x},\overline{y},\overline{p}):=(\widehat{x},\widehat{y},\widehat{p})$.
4. Continue the subgradient algorithm (i.e. starting from the multipliers obtained after previous run) finding updated λ, and set $BLB:=\mathcal{L}_2(\lambda)$.
5. If $h<2$, then set $h:=h+1$ and go to step 3, else go to step 6.
6. If $Err>\varepsilon$, then enlarge the core (i.e. increase the thresholds v and μ), else go to step 12.
7. Construct $I(\lambda)$, $W(\lambda)$ and find a solution $(\widehat{x},\widehat{y},\widehat{p})$ of the core problem. If $Z(\widehat{x},\widehat{y},\widehat{p})<BUB$, then set $BUB:=Z(\widehat{x},\widehat{y},\widehat{p})$ and $(\overline{x},\overline{y},\overline{p}):=(\widehat{x},\widehat{y},\widehat{p})$.
8. Continue the subgradient algorithm (i.e. starting from the multipliers obtained after previous run), finding updated λ and set $BLB:=\mathcal{L}_2(\lambda)$.
9. If $Err>\varepsilon$, then go to step 10, else go to step 12.
10. Construct $I(\lambda)$, $W(\lambda)$ and find a solution $(\widehat{x},\widehat{y},\widehat{p})$ of the core problem. If $Z(\widehat{x},\widehat{y},\widehat{p})<BUB$, then set $BUB:=Z(\widehat{x},\widehat{y},\widehat{p})$ and $(\overline{x},\overline{y},\overline{p}):=(\widehat{x},\widehat{y},\widehat{p})$.
11. Continue the subgradient algorithm (i.e. starting from the multipliers from Step 8), finding updated λ and set $BLB:=\mathcal{L}_2(\lambda)$.
12. Return *BLB*, *BUB* and $(\overline{x},\overline{y},\overline{p})$.

The main feature of our core heuristic implementation is a dynamic selection of the size of the core. If after three runs of the subgradient algorithm and two runs of the core problem solving procedure a small relative difference between computed best upper (*BUB*) and best lower (*BLB*) bounds is not obtained, we increase the thresholds v, μ, and we find new values of *BLB*, *BUB* and $(\overline{x},\overline{y},\overline{p})$. Then if a small relative difference between *BUB* and *BLB* is still high, we again run the subgradient optimization and the core problem solving procedure with the same enlarged thresholds (the steps 10 and 11).

5. Computational results

The algorithm has been written in C++, and experiments were carried out on a PC with Intel Core 2 Duo CPU 2.20 GHz and 3 Gb of RAM, o.s. Windows Vista 32-bit. The commercial IP solver used to solve the core problem was CPLEX Optimizer 12.1.0. Our algorithm is not parallel, while CPLEX Optimizer to solve the core problem uses up all two threads. Our test bed consists of two types of instances taken from the literature:

- *Synthetic dataset generator*: The first type of data sets represented here is generated as suggested in Zhang et al. [29,30] and was used by Hansen et al. [11] for testing their approach to solving large p-median clustering problems. Each data instance consists of p clusters of 2-d data points. A cluster is characterized by the number of data points in it, its radius and its center. There are three pattern parameters determining the location of the center of each cluster, namely grid, sine and random. When the grid pattern is used, the cluster centers are located on $\sqrt{p}\times\sqrt{p}$ grid. The random pattern places the cluster centers randomly. Here we report results for instances generated with grid (type I) and randomly (type III) patterns. All generated problem instances contain from 1000 to 10 000 points in the plane of each type. In total we have generated 8 instances of each dimension (i.e. each problem instance was generated with 9, 16, 25, 36, 49, 64, 81, 100 clusters). Note that data set of type I is easier to solve, while data set III is harder.
- *TSP library instances*: These instances are taken from the TSP library (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/). The numbers attached to each problem title indicate the number of points in that problem. Note, that we consider instances with a number of points which is not more than 10000, except *usa*13509 and *rl*11849.

Note that for the considered instances, we have $I=J$ and $d_{ii}=0$, so we pose $y_i=x_{ii}$ and we take into account in our code.

These data sets are of much bigger size than those reported so far in the literature on the problem [18,17]. In fact, our sets are so big that the distance matrix cannot be properly handled. Hence, as discussed in Section 3.4 we limit the number of active elements of the ordered distance matrix, namely, all active elements take 800 MB of RAM.

The core heuristic parameters on the initial steps are the same as in Avella et al. [4]:

- On step 3 and 4, parameters v and μ are chosen in order to limit the size of the core problem with $I(\lambda)=3p(\lambda)$, $W(\lambda)=5n$. The running time limit for the core problem is set to 300 s.
- On steps 7, 8 and 10, 11 the sets $I(\lambda)$ and $W(\lambda)$ are enlarged so that $I(\lambda)=6p(\lambda)$, $W(\lambda)=10n$. The running time limit for the core problem is set to 400 s on steps 7, 8 and to 600 seconds on steps 10 and 11.
- Parameter ε is set to 1.
- On step 2, parameter β is set to 1.5 and divided by 1.01 on each iteration where the lower bound is not improved.
- On step 4 ($h=1$), β is set to 0.1 and updated as in the previous way.
- Finally on step 4 ($h=2$), β is set to 0.005 and divided by 1.01 if the lower bound has not improved after 2 iterations.

5.1. Convex penalty

Computational results are reported in Tables 1–3, where columns *Err* (%) contain the relative difference between the best lower bounds and upper bounds ($Err=(BUB-BLB)/BUB\cdot 100\%$), and columns *Time* contain the total computation time in seconds. This also includes the running time of the procedure for solving the core problem.

Note that for synthetic generated instances, columns *Err* (%) and *Time* contain average values for the eight problem instances of each dimension with different number of open facilities.

The first type of penalty we use for testing the algorithm is a convex function. We consider convex functions of the form $\phi(p)=cp^2$ with different values of c, namely $\phi(p)=p^2$, $\phi(p)=5p^2$ and $\phi(p)=10p^2$.

Tables 1–3 provide results for both types of considered instances using such type of penalty function. Results for synthetic generated instances are presented in Table 1, while Tables 2 and 3 contain results on TSP problems, obtained with relaxations $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively.

Table 1
Results on synthetic generated instances with convex penalty function.

	$\mathcal{L}_2^*$						$\mathcal{L}_1^*$					
	Err (%)			Time			Err (%)			Time (s)		
$\phi(p)$	p^2	$5p^2$	$10p^2$	p^2	$5p^2$	$10p^2$	p^2	$5p^2$	$10p^2$	p^2	$5p^2$	$10p^2$
n	Type I											
1000	0.007	0.056	0.005	4.02	7.13	5.66	0.009	0.382	1.132	5.00	7.98	11.18
2000	0.004	0.063	0.111	12.30	27.27	29.20	0.049	1.191	3.145	13.64	36.54	54.01
3000	0.128	0.291	0.069	26.56	40.64	58.41	0.139	1.767	4.162	29.95	69.41	91.76
4000	0.109	0.127	0.162	60.41	135.81	107.32	0.448	2.723	6.723	63.51	114.74	159.31
5000	0.024	0.078	0.047	82.55	118.86	159.58	0.853	5.242	21.254	100.70	193.21	228.50
6000	0.017	0.033	0.056	175.09	210.50	216.95	1.070	7.284	17.268	157.75	248.04	371.95
7000	0.103	0.012	0.015	167.65	243.61	239.18	1.802	16.545	17.186	362.65	350.74	497.09
8000	0.068	0.034	0.182	259.44	225.98	327.11	2.313	16.300	25.310	485.61	353.16	558.47
9000	0.107	0.057	0.067	391.52	351.75	368.07	4.222	17.382	24.753	432.46	466.22	633.04
10 000	0.112	0.003	0.281	445.77	350.74	442.09	3.202	18.214	22.574	563.49	544.92	717.20
	Type III											
1000	0.001	0.000	0.000	3.74	4.72	5.16	0.005	0.240	10.799	4.31	5.99	9.26
2000	0.004	0.001	0.002	13.60	15.51	18.37	0.155	0.844	12.429	17.16	22.76	28.85
3000	0.040	0.001	0.001	31.11	31.79	37.07	0.342	1.586	4.178	38.83	44.68	61.70
4000	0.106	0.001	0.001	51.16	59.32	64.91	0.547	4.460	5.347	69.98	80.72	115.69
5000	0.029	0.017	0.014	109.67	109.55	101.90	1.428	6.462	7.859	114.12	138.59	165.97
6000	0.099	0.001	0.001	138.95	135.85	157.23	1.554	8.854	10.973	227.88	188.76	232.34
7000	0.154	0.002	0.100	252.22	195.20	196.17	1.739	7.722	11.765	303.14	286.07	301.95
8000	0.240	0.009	0.002	221.32	248.42	275.46	3.107	9.168	17.278	396.71	370.84	351.51
9000	0.113	0.017	0.001	410.50	329.41	329.48	8.039	12.040	18.453	552.69	375.27	449.92
10 000	0.144	0.271	0.029	470.84	342.38	414.66	5.594	18.094	25.842	736.62	486.75	499.22

Table 2
Results on TSP instances using $\mathcal{L}_1$ and convex penalty function.

Instance	Err (%)			Time (s)		
$\phi(p)$	p^2	$5p^2$	$10p^2$	p^2	$5p^2$	$10p^2$
d1291	9.640	10.124	0.035	86.14	8.74	6.21
d1655	0.249	24.150	13.112	6.56	18.06	19.25
d2103	4.316	28.576	3.530	54.29	827.87	1064.16
dsj1000	11.300	0.013	1.769	5.62	2.27	5.17
fl417	0.029	0.001	0.002	0.85	1.05	1.05
fl1400	18.647	17.472	8.995	22.86	18.31	14.24
fl1577	28.744	24.552	2.665	13.57	11.89	14.12
fl3795	15.182	72.579	88.516	53.01	51.27	50.34
fnl4461	2.303	8.314	9.043	1050.77	791.87	1068.66
nrw1379	9.311	2.796	2.277	31.72	49.84	190.16
pcb1173	0.356	0.040	0.193	3.32	11.09	6.29
pr1002	4.628	0.131	0.746	7.16	6.83	2.58
pr2392	0.737	7.190	23.298	10.39	55.26	21.17
rl1889	0.800	2.243	10.060	9.17	36.47	14.88
rl5915	8.696	8.763	59.895	135.77	299.38	193.91
rl5934	0.905	7.777	5.657	141.05	1082.89	349.98
rl11849	3.557	54.951	18.055	1215.72	1321.63	1270.33
u1060	0.783	0.001	7.177	6.69	2.40	13.66
u1432	4.143	5.622	6.510	28.69	15.33	32.15
u1817	4.536	6.125	7.144	114.42	17.24	15.08
u2152	0.125	1.776	3.422	29.16	30.31	35.10
u2319	22.225	18.681	9.387	1118.55	1027.03	1019.65
vm1084	2.167	6.959	1.204	7.75	7.11	8.17
vm1748	1.426	3.580	2.077	13.76	14.87	13.63
pcb3038	2.672	10.226	7.756	91.96	86.55	51.19
usa13509	4.520	2.654	18.881	515.35	1247.71	1567.86

Table 3
Results on TSP instances using $\mathcal{L}_2$ and convex penalty function.

Instance	Err (%)			Time (s)		
$\phi(p)$	p^2	$5p^2$	$10p^2$	p^2	$5p^2$	$10p^2$
d1291	0.030	0.020	0.034	5.09	6.04	5.76
d1655	0.062	0.015	0.004	28.30	8.13	7.64
d2103	0.015	0.077	0.355	379.23	413.29	413.88
dsj1000	0.001	0.001	0.009	2.79	2.65	3.01
fl417	0.001	0.002	0.001	1.04	1.00	1.09
fl1400	0.010	0.001	0.001	6.06	5.97	7.05
fl1577	0.012	0.002	0.004	6.44	5.79	8.28
fl3795	0.755	0.005	0.005	30.58	40.31	45.54
fnl4461	0.050	0.760	0.070	362.37	58.98	447.52
nrw1379	0.017	0.020	0.087	5.59	6.43	11.98
pcb1173	0.008	0.013	0.064	4.24	4.34	8.33
pr1002	0.010	0.000	0.000	3.77	3.13	2.61
pr2392	0.001	0.020	0.000	13.77	11.81	11.59
rl1889	0.006	0.018	0.008	9.48	8.06	9.27
rl5915	0.007	0.019	0.018	70.60	70.12	73.87
rl5934	0.012	0.011	0.030	159.62	61.34	405.07
rl11849	0.020	0.043	0.971	506.90	523.82	394.69
u1060	0.013	0.001	0.003	4.58	3.40	2.96
u1432	0.039	0.023	0.041	9.32	6.32	7.74
u1817	0.049	0.005	0.001	22.95	8.69	10.83
u2152	0.033	0.019	0.012	12.09	12.98	15.26
u2319	0.219	0.891	0.811	50.61	415.52	817.46
vm1084	0.001	0.001	0.001	3.42	2.81	3.04
vm1748	0.001	0.002	0.001	6.44	6.19	7.52
pcb3038	0.021	0.018	0.042	31.22	19.91	28.31
usa13509	0.015	0.013	0.024	260.05	292.97	560.59

As one can see for the synthetic generated instances we get good quality solutions in all cases, using $\mathcal{L}_2$ as Lagrangean relaxation. With $\mathcal{L}_1$, acceptable results have been obtained only on instances with small number of points and with the penalty functions $\phi(p)=p^2$ and $\phi(p)=5p^2$. With regard to the running time, it is slightly more in the case when we use the first type of Lagrangean relaxation $\mathcal{L}_1$. For TSP problem instances we observe the same situation in the case of using $\mathcal{L}_2$ relaxation. With $\mathcal{L}_1$ good results have been obtained only for some problems with relatively small number of points (see Table 2).

After this analysis for the simplest case of convex penalty functions, we conclude that $\mathcal{L}_2$ systematically outperforms $\mathcal{L}_1$. For harder problems, as those analyzed in Section 5.2, we only present the discussion when using the most competitive Lagrangean relaxation, namely $\mathcal{L}_2$.

5.2. Piecewise linear concave penalty

As shown in Section 5.1, our procedure yields very tight relaxations if the penalty function ϕ is convex. On the other hand, our preliminary computational experiments with arbitrary concave penalty functions did not give the same good results, since the duality gap was much bigger. Hence, if this big gap was to be closed, we should have increased the sizes of the IPs to be solved exactly within the core procedure.

However, as detailed below, our approach can be successfully applied for a particular yet very important class of concave penalty functions, namely the class of concave piecewise linear functions. Indeed, in this case the problem can be split into a series of problems with a smaller range of possible values for p, within which ϕ is linear, thus convex, and hence the good results for the sharp bounds are also obtained within each subproblem.

Let as consider a concave piecewise linear function $\phi(\cdot)$, $\phi(p) = \min_{1 \le i \le s} k_i p + b_i$. Such a function can be rewritten in the form

$$\phi(p) = \begin{cases} k_1 p + b_1, & p \in [l_0, l_1], \\ k_2 p + b_2, & p \in [l_1, l_2], \\ \cdots \\ k_s p + b_s, & p \in [l_{s-1}, l_s], \end{cases} \tag{20}$$

with $k_1 > k_2 > \cdots > k_s$, and $l_0 = 1$, $l_s = n$.

To solve Problem (1) with $\phi(\cdot)$ given by (20), we sequentially run the algorithm restricting p to be in the corresponding interval $[l_{j-1}, l_j]$, and considering as lower and upper bounds for (1) the lowest ones of those returned by the algorithm in the different pieces.

In our experiments we consider a piecewise linear function with $s=4$ and $s=6$ pieces. The coefficients b_i were chosen so that $b_0 = 0$ and $\phi(\cdot)$ is continuous, and the knots l_i in (20) satisfy the relation $2(l_i - l_{i-1}) = l_{i+1} - l_i$. As a set of slopes $K_s = \{k_i \in \mathbb{R}, i = 1, \ldots, s\}$ we take in the synthetic instances $K_4 = \{25, 15, 8, 1\}$ and $K_6 = \{30, 25, 18, 11, 6, 1\}$. Since the costs d_{ij} in the TSP instances are much bigger, we enlarge the slopes, taking $K_4 = \{300, 250, 225, 200\}$ and $K_6 = \{170, 140, 120, 100, 75, 50\}$. Note that for this type of penalty function $\phi(\cdot)$ we only use the second Lagrangean relaxation $\mathcal{L}_2$, which in our preliminary numerical experience reported the best results.

Table 4
Results on synthetic generated instances using $\mathcal{L}_2$ and concave piecewise linear penalty function.

k	Err (%)		Time (s)	
	$s=4$	$s=6$	$s=4$	$s=6$
n	Type I			
1000	0.001	0.000	3.20	10.03
2000	0.008	0.004	8.30	17.33
3000	0.082	0.102	17.32	29.06
4000	0.099	0.161	31.96	36.50
5000	0.093	0.057	64.67	121.33
6000	0.490	0.705	93.29	133.88
7000	0.461	0.454	293.65	194.24
8000	0.921	0.052	230.44	198.59
9000	0.871	0.859	426.12	448.25
10 000	2.035	1.406	439.14	559.90
	Type III			
1000	0.114	0.085	3.29	5.69
2000	0.004	0.161	10.30	15.98
3000	0.014	0.018	44.62	36.33
4000	0.070	0.202	69.92	77.52
5000	0.122	0.211	91.89	142.08
6000	0.571	0.587	205.16	124.38
7000	0.893	0.323	338.12	185.75
8000	0.412	1.366	276.46	416.15
9000	2.396	0.651	421.77	570.32
10 000	1.583	0.497	630.57	558.83

Table 5
Results on TSP instances using $\mathcal{L}_2$ and concave piecewise linear penalty function.

Instance	Err (%)		Time (s)	
k	$s=4$	$s=6$	$s=4$	$s=6$
d1291	0.380	1.265	6.69	143.40
d1655	0.074	0.008	40.06	134.55
d2103	0.548	0.620	519.52	1221.68
dsj1000	0.000	0.000	6.18	26.71
fl417	0.001	0.002	1.50	7.86
fl1400	0.001	0.002	7.48	38.44
fl1577	0.521	0.050	12.86	344.45
fl3795	0.529	0.085	79.95	717.35
fnl4461	0.086	0.524	432.93	2417.17
nrw1379	0.019	0.560	10.27	44.91
pcb1173	0.020	0.012	7.72	50.45
pr1002	0.082	0.000	14.74	36.49
pr2392	0.008	0.001	20.54	110.11
rl1889	0.004	0.001	18.33	76.11
rl5915	0.002	0.073	122.23	2127.14
rl5934	0.004	0.014	192.76	1739.04
rl11849	0.611	0.023	622.83	5418.43
u1060	0.068	0.000	6.98	32.31
u1432	0.609	0.000	12.24	368.13
u1817	0.170	0.818	33.27	73.04
u2152	0.014	0.034	26.89	76.27
u2319	1.097	0.000	4700.02	4786.08
vm1084	0.683	0.058	8.42	36.41
vm1748	0.176	0.001	16.20	69.09
pcb3038	0.008	0.031	71.11	714.17
usa13509	0.001	0.007	769.51	4140.02

Table 4 presents the results for synthetic generated instances. Columns 4 and 6 contain the average relative difference and the average computational time with a piecewise linear function with 4 and 6 pieces respectively. Results for TSP problem instances are given in Table 5.

As one can see for the synthetic generated instances a very small relative difference, less than one percent, between the best upper and lower bounds is obtained for all instances with small number of points (for problem instances of Type III, with up to 8000 points and for problem instances of Type I with up to 10 000 points). For the TSP instances, the relative difference is small in most cases (except $d1291$ and $u2319$), but the running time is much higher in the event that a piecewise linear function with 6 pieces is used.

6. Conclusions

In this paper, a nonlinear variant of the minsum facility location problem has been addressed, in which the number of facilities to be placed is a decision variable. Two Lagrangean relaxations have been introduced, and a core heuristic is proposed to obtain upper bounds. Computational experience is reported for several classical problems in facility location, and also a class of data sets from Data Analysis. Different penalty functions have been considered, showing that, in all instances with convex penalty functions, the combination of lagrangian relaxation and core heuristics provides in very reasonable time very good heuristic solutions. Moreover, one lagrangian relaxation systematically outperforms the other. When the penalty function is not convex, the duality gap is much bigger. Although the problem is still tractable when the penalty function is piecewise linear with a small set of pieces, it remains an open question how to obtain so competitive results in the general case.

References

[1] Daskin M. Network and discrete location: models, algorithms, and applications. New York: John Wiley & Sons, Inc.; 1995.

[2] Kariv O, Hakimi S. An algorithmic approach to network location problems; part 2. The *p*-medians. SIAM Journal on Applied Mathematics 1979;37(3): 539–60.

[3] Mirchandani P, Francis R, editors. Discrete location theory. New York: John Wiley & Sons, Inc.; 1990.

[4] Avella P, Boccia M, Salerno S, Vasilyev I. An aggregation heuristic for large scale *p*-median problem. Computers and Operations Research 2012;39(7): 1625–32.

[5] Avella P, Sassano A, Vasilev I. Computational study of large-scale *p*-median problems. Mathematical Programming 2007;109(1):89–114.

[6] Beasley J. Lagrangean heuristics for location problems. European Journal of Operational Research 1993;65(3):383–99.

[7] Mladenović N, Brimberg J, Hansen P, Moreno-Pérez J. The *p*-median problem: a survey of metaheuristic approaches. European Journal of Operational Research 2007;179(3):927–39.

[8] Reese J. Solution methods for the *p*-median problem: an annotated bibliography. Networks 2006;28(3):125–42.

[9] Marianov V, Serra D. Location problems in the public sector.Facility location: applications and theory. New York: Springer; 2002 pp. 119–44.

[10] Brusco M, Köhn H. Optimal partitioning of a data set based on the *p*-median model. Psychometrika 2008;73(1):89–105.

[11] Hansen P, Brimberg J, Urosević D, Mladenović N. Solving large *p*-median clustering problems by primal-dual variable neighborhood search. Data Mining and Knowledge Discovery 2009;19(3):351–75.

[12] Klastorin T. The *p*-median problem for cluster analysis: a comparative test using the mixture model approach. Management Science 1985;31(1):84–95.

[13] Avella P, Boccia M, Martino CD, Oliviero G, Sforza A, Vasilev I. A decomposition approach for a very large scale optimal diversity management problem. 4OR 2005;3(1):23–37.

[14] Briant O, Naddef D. The optimal diversity management problem. Operations Research 2004;52(4):515–26.

[15] Sugar C, James G. Finding the number of clusters in a dataset: an information-theoretic approach. Journal of the American Statistical Association 2003; 98(463):750–63.

[16] Krarup J, Pruzan P. The simple plant location problem: survey and synthesis. European Journal of Operational Research 1983;12(1):36–81.

[17] Mirchandani P, Jagannathan R. Discrete facility location with nonlinear diseconomies in fixed costs. Annals of Operations Research 1989;18(1): 213–24.

[18] Körkel M. Discrete facility location with nonlinear facility costs. RAIRO Recherche Opérationnelle 1991;25(1):31–43.

[19] Carrizosa E, Conde E. A fractional model for locating semi-desirable facilities on networks. European Journal of Operational Research 2002;136(1):67–80.

[20] Feldman E, Lehrer F, Ray T. Warehouse location under continuous economies of scale. Management Science 1966;12(3):670–84.

[21] Hajiaghayi M, Mahdian M, Mirrokni V. The facility location problem with general cost functions. Networks 2003;42(1):42–7.

[22] Harkness J, ReVelle C. Facility location with increasing production costs. European Journal of Operational Research 2003;145(1):1–13.

[23] Murray W, Shanbhag U. A local relaxation method for nonlinear facility location problems. In: Multiscale optimization and applications (nonconvex optimization and its applications). New York: Springer; 2006. p. 173–204.

[24] Soland R. Facility location with concave costs. Operations Research 1974; 23(2):373–82.

[25] Wu L, Zhang X, Zhang J. Capacitated facility location problem with general setup cost. Computers and Operations Research 2006;33(5):1226–41.

[26] Avella P, Boccia M, Sforza A, Vasilyev I. An effective heuristic for large-scale capacitated facility location problems. Journal of Heuristics 2008;15(6): 597–615.

[27] Caprara A, Fischetti M, Toth P. A heuristic method for the set covering problem. Operations Research 1999;47(5):730–43.

[28] Mulvey J, Crowder H. Cluster analysis: an application of Lagrangean relaxation. Management Science 1979;25(4):329–40.

[29] Zhang T, Ramakrishnan R, Livny M. Birch: an efficient data clustering method for very large databases. Journal of the American Statistical Association 1996;98(463):103–14.

[30] Zhang T, Ramakrishnan R, Livny M. Birch: a new data clustering algorithm and its applications. Data Mining and Knowledge Discovery 1997;1(2):141–82.