The p-center problem: Using equivalent instances to obtain better results

Alfredo Marín

Departamento de Estadística e Investigación Operativa, Universidad de Murcia, Spain.

email: amarin@um.es

December 20, 2024

Abstract

Given a matrix $(d_{ij})_{n\times m}$ and $p \in \{2,\ldots,m-1\}$ the p-center problem looks for the set of p columns that minimizes the maximum of the minimum value of the rows in the p given columns. In location science, columns are interpreted as potential locations for facilities, rows are interpreted as demanding points and (d_{ij}) are the distances between them.

Different MILP formulations have been proposed so far. The usual way to check the goodness of these formulations has been comparing sizes and lower bounds. We introduce the concept of equivalent instances for the problem, that produce the same optimal solutions but not the same optimal values, making the comparison between lower bounds questionable. Then we take advantage of our results to design algorithms that, applied to the instances previously considered in the literature, obtain better solutions in less time.

Keywords: p-center; discrete location; integer programming;

1 Description of the problem and previous work

We are given a matrix (d_{ij}) (called distance matrix) with set of rows $N = \{1,\ldots,n\}$ and set of columns $M = \{1,\ldots,m\}$, where $n,m \geq 3$, and a constant $p \in \{2,\ldots,m-1\}$. For the sake of readability we call N the set of sites and M the set of potential centers. The p-center problem (pCP) consists in choosing a subset of potential centers $P \subset M$ with $|P| \leq p$ in order to minimize

$$\max_{i\in N}\min_{j\in P}\{d_{ij}\}.$$

The elements of P will be called centers.

The p-center problem is a classical problem in location science. [Hakimi, 1964] introduced el problem on a network when $p = 1$, and the extension to p centers was mentioned in [Hakimi, 1965] and explicitly stablished in [Minieka, 1970]. In that field, the elements of M are interpreted as potential locations for, e.g., emergency services, the elements of N are interpreted as demanding points and (d_{ij}) are the distances or travel times from centers to demanding points, or vice versa. It is also possible to think in d_{ij} as the product of a travel time and some kind of demand associated to the site i. When the entry d_{ij} is less than or equal to the optimal value of pCP, it is considered that i can be *allocated* or *assigned* to a center located in j, *with a cost* d_{ij}. It was proven in [Kariv and Hakimi, 1979] that pCP is NP-hard.

There is an obvious relation between pCP and the set covering problem. In the latter, the minimum amount of columns of the distance matrix have to be chosen to achieve that all rows contain a number less than or equal to a given threshold value R in some of the chosen columns. If this minimum number of columns is less than or equal to p, the optimal value of pCP will be less than or equal to R, see e.g. [Minieka, 1970], [Daskin, 1995], [Ilhan and Pınar, 2001]. Looking for the minimum value of R satisfying this property is a commonly used approach to solve pCP.

In the field of location, additional constraints can be imposed on the solutions. A well studied one is a limit in the demand allocated to each of the centers, see e.g. [Albareda-Sambola et al., 2010], [Espejo et al., 2015] and [Özsoy and Pınar, 2006]. Other authors extend the study to consider, for instance, upgrading (reduction of the costs under a budget limit) [Antón-Sánchez et al., 2023] or allocation to more than one center to get robust solutions [Hinojosa et al., 2023], [Duran-Mateluna et al., 2023]. A review of pCP-related problems can be consulted in [Calık et al., 2019].

Several method to exactly solve pCP have been devised through the years. Seminal articles of special interest are [Ilhan and Pınar, 2001], [Elloumi et al., 2004] and [Calık and Tansel, 2013]. Recently, more intricate methods that produce the best computational results for large instances have been published. [Chen and Handler, 1987] and [Chen and Chen, 2009] introduced and [Contardo et al., 2019] improved a row generation algorithm that iteratively solved small subproblems by considering only a subset of sites that was updated in each iteration. The algorithm is scalable and allowed to solve big instances. However, instances with larger values of p could not be solved as efficiently as those with smaller values. [Gaar and Sinnl, 2022] considered a family of formulations for pCP based on the previous knowledge of a lower bound on the optimal value of the problem. Each formulation generated a new bound that in turn replaced that of the formulation and the iterative process produced a final lower bound and heuristic solutions that, in some cases, were proved to be optimal. We give details in Section 2.

pCP can be naturally formulated as a (mixed) integer programming problem, and most of the solution methods use this possibility. We will present and analyze the main formulations of the literature in Section 2. Then, in Section 3 a new solution algorithm based on the theoretical considerations previously presented is developed. The method is designed to produce optimal solutions but also looks for good feasible solutions on the way. Computational results, and in particular improved solutions for large instances of the commonly used benchmarks are given in Section 4. After the conclusions of Section 5, and for the sake of completeness, we have added an Appendix with detailed numerical results.

2 Formulations

In what follows, given a formulation (F) with minimization of the objective function, we denote with $v(F)$ its optimal value, with $(\bar{F})$ its linear relaxation and with $v(\bar{F})$ the lower bound on $v(F)$ obtained solving the linear relaxation.

The first MILP formulation for the p-center problem we present is the classical one introduced in [Daskin, 1995]. It uses two families of binary variables given by

$$y_j = \begin{cases} 1 & \text{if } j \text{ is chosen as a center} \\ 0 & \text{otherwise} \end{cases}$$

for all $j \in M$ and

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

for all $i \in N$, $j \in M$, plus an auxiliary continuous variable θ:

$$\begin{aligned}
\text{(C)} \quad \min \quad & \theta \\
\text{s.t.} \quad & \sum_{j \in M} y_j = p && (1) \\
& x_{ij} \leq y_j \quad \forall i \in N, j \in M && (2) \\
& \sum_{j \in M} x_{ij} = 1 \quad \forall i \in N && (3) \\
& \theta \geq \sum_{j \in M} d_{ij} x_{ij} \quad \forall i \in N && (4) \\
& y_j \in \{0, 1\} \quad \forall j \in M && (5) \\
& x_{ij} \in \{0, 1\} \quad \forall i \in N, j \in M. && (6)
\end{aligned}$$

Constraint (1) fixes the amount of centers, each constraint in (2) forces x_{ij} to take value 0 if j has not been chosen as a center, constraints (3) force allocation of all elements in N and (4) plus the minimization of the objective function make θ take the value of the minimum distance. Note that (6) can be relaxed.

In order to introduce the most interesting formulations for the p-center problem, let D_0 (resp. D_g) be the smallest (resp. largest) value in the distance matrix, and let $D_0 < D_1 < D_2 < \ldots < D_g$ be the sorted different values in that matrix. Let $G := \{1, \ldots, g\}$.

For given values of the aforementioned variables y_j, let

$$f_i(k) := \sum_{\substack{j \in M: \\ d_{ij} \leq D_k}} y_j.$$

For $i \in N$ and $k \in \{1, \dots, g-1\}$, it holds $f_i(k) \leq f_i(k+1)$. The maximum of the minimum distances corresponding with the centers given by the y-variables will be $\theta = D_k$ iff (i) $f_i(k) = 1 \ \forall i \in N$ and (ii) $\exists i \in N : \ f_i(k-1) = 0$. That is to say,

$$\theta = \sum_{k \in G} D_k (\min\{1, \min_{i \in G} f_i(k)\} - \min\{1, \min_{i \in G} f_i(k-1)\}). \tag{7}$$

Taking into account that, when $y_j \in \{0, 1\} \ \forall j \in M$, all the addends in (7) are equal to 0 except one of them that takes value 1, (7) is equivalent to

$$\theta = \max_{k \in G} D_k (\min\{1, \min_{i \in G} f_i(k)\} - \min\{1, \min_{i \in G} f_i(k-1)\}). \tag{8}$$

Note that (8) is weaker than (7) in the sense that relaxing the binarity of the y-variables to $y_j \in [0, 1]$, (8) will give a value lower than the one obtained using (7).

Let then define binary variables $w_k := \min\{1, \min_{i \in G} f_i(k)\} \ \forall k \in G$ and take $\theta = \sum_{k \in G} D_k (w_k - w_{k-1}) = \sum_{k \in G} (D_k - D_{k-1}) w_k$. Since coefficients $D_k - D_{k-1}$ are always positive, minimizing this sum (lowerly bounding w_k by $f_i(k)$ for all $i \in N$) will produce a valid formulation for pCP. Radius formulation (R), introduced in [Elloumi et al., 2004], equivalently used binary variables $z_k = 1 - w_k$ and was designed as:

$$\text{(R)} \quad \min \quad D_0 + \sum_{k \in G} (D_k - D_{k-1}) z_k \tag{9}$$

$$\text{s.t.} \quad (1), (5)$$

$$z_k \geq 1 - \sum_{\substack{j \in M: \\ d_{ij} < D_k}} y_j \qquad \forall i \in N, k \in G \tag{10}$$

$$z_k \in \{0, 1\} \qquad \forall k \in G. \tag{11}$$

A very simple yet efficient simplification of formulation (R), observed in [Ales and Elloumi, 2018], takes into account that most of the constraints (10) are dominated by others after adding the much smaller set of constraints (13). The reason is that $\{j \in M : \ d_{ij} < D_k\} = \{j \in M : \ d_{ij} < D_{k+1}\}$ when D_k is not one of the values in the i-th row of the distance matrix. Let then G^i be the subset of G containing the indices k such that D_k is one of the values of row i. Replacing (10) by (12) and adding (13), the modified radius formulation (R') is given by

$$
\begin{aligned}
\text{(R')} \quad & \min \quad (9) \\
& \text{s.t.} \quad (1), (5), (11) \\
& z_k \geq 1 - \sum_{\substack{j \in M: \\ d_{ij} \leq D_k}} y_j \qquad \forall i \in N, k \in G^i \qquad (12) \\
& z_k \geq z_{k+1} \qquad \forall k \in \{1, \ldots, g-1\}. \qquad (13)
\end{aligned}
$$

A formulation similar to (R) was introduced in [Calık and Tansel, 2013]. It gives the same lower bounds as (R) and will not be considered in this paper.

On the other hand, using (8), the minimum of θ can be obtained by lowerly bounding it as

$$\theta \geq D_k(\min\{1, \min_{i \in G} f_i(k)\} - \min\{1, \min_{i \in G} f_i(k-1)\}) \ \ \forall k \in G.$$

Rewriting

$$\theta \geq D_k \min\{1, \min_{i \in G} f_i(k)\} - \min\{D_k, \min_{i \in G} D_k f_i(k-1)\} =$$

$$D_k \min\{1, \min_{i \in G} f_i(k)\} + \max\{-D_k, \max_{i \in G}(-D_k \sum_{\substack{j \in M: \\ d_{ij} \leq D_{k-1}}} y_j)\},$$

we can equivalently write

$$\theta \geq D_k + \max\{-D_k, \max_{i \in G}(-D_k \sum_{\substack{j \in M: \\ d_{ij} \leq D_{k-1}}} y_j)\} = \max\{0, \max_{i \in G} D_k - \sum_{\substack{j \in M: \\ d_{ij} \leq D_{k-1}}} D_k y_j\},$$

i.e.,

$$\theta \geq D_k - \sum_{\substack{j \in M: \\ d_{ij} \leq D_{k-1}}} D_k y_j \ \forall i \in N, \ \ \forall k \in G.$$

It was observed in [Gaar and Sinnl, 2022] that these inequalities can be improved to

$$\theta \geq D_k - \sum_{\substack{j \in M: \\ d_{ij} \leq D_{k-1}}} (D_k - d_{ij}) y_j \ \ \forall i \in N \ \forall k \in G^i.$$

Moreover, [Gaar and Sinnl, 2022] introduced a new family of formulations, based on the previous knowledge of a lower bound B on the optimal value of the problem:

$$
\begin{aligned}
\text{(G}(B)\text{)} \quad & \min \quad \theta \\
& \text{s.t.} \quad (1), (5) \\
& \theta \geq D_k - \sum_{\substack{j \in M: \\ d_{ij} \leq D_{k-1}}} (D_k - \max\{B, d_{ij}\}) y_j \ \ \forall i \in N, \ \forall k \in G^i: \ D_k \geq B \qquad (14) \\
& \theta \geq B.
\end{aligned}
$$

Formulations (G(B)) avoid the use of variables z_k obtaining directly the value of z from the values of the y-variables. In (14), if $\{\ell \in M : \ d_{ij} < D_k\} = \emptyset$ for some $k \in G$, the maximum allocation cost z will take value at least equal to D_k. In the case $\{j \in M : \ d_{ij} < D_k\} = \{t\}$, θ will be lowerly bounded by $\max\{B, d_{it}\} < \max\{B, D_k\} = B$. Otherwise, if $|\{\ell \in M : \ d_{ij} < D_k\}| \geq 2$, this lower bound will be even smaller.

Another formulation was considered in [Ales and Elloumi, 2018]. It uses an auxiliary variable r, and is given by

$$
\begin{aligned}
\text{(A)} \quad & \min \quad r \\
& \text{s.t.} \quad (1), (5) \\
& \qquad r \geq k(1 - \sum_{\substack{j \in M: \\ d_{ij} \leq D_k}} y_j) \qquad \forall i \in N, k \in G^i \cup \{g\}. \qquad (15)
\end{aligned}
$$

Note that, for a given value of i and a given distance D_k in the i-th row of the matrix, constraint (15) makes r be at least equal to the index k of the distance D_k when there is not a center under the distance D_k from i. Therefore, the optimal value of (A) is not the optimal value of pCP, but the place of the maximum distance in the sorted vector of distances (D_k), $k \in G$. That is to say, for all $B \leq v(C)$ it holds $v(C) = v(R) = v(R') = v(G(B)) = D_{v(A)}$.

Even if the lower bound produced by the formulation is the main factor to be taken into account when using it in a branch-and-bound or branch-and-cut algorithm, huge formulations make it impossible to obtain this bound when large instances are considered. Let us then compare all these formulations from two points of view: Size and quality of the lower bound.

Formulation (C) contains $O(nm)$ variables and constraints, although only m variables are properly integer. Formulation (R) contains $O(g)$ variables and $O(ng)$ constraints. The size then depends on the number of ties between the values in the matrix. In other words, g can take any value between 1 and nm. Again most of the integrality constraints on the variables can be relaxed. The derived formulation (R') succeeds to reduce the number of constraints of (R) in one order of magnitude. Finally, formulations (G(B)) reduce the total number of variables to $m + 1$ and contain $O(nm)$ constraints, depending on the value of the lower bound B.

Regarding the lower bound produced by the formulations, it was proved in [Elloumi et al., 2004] that $v(\bar{R}) \leq v(\bar{C})$ and the inequality can be strict. It was also proved in [Ales and Elloumi, 2018] that $v(\overline{R'}) = v(\bar{R})$. In [Gaar and Sinnl, 2022] they proved that $v(\overline{G(0)}) \leq v(\bar{C})$, but the bound $v(\overline{G(B)})$ can increase when B increases. Let us consider the following example, used in [Elloumi et al., 2004] to show that $v(\bar{R})$ can be strictly better than $v(\bar{C})$.

Example 2.1. *Consider the instance $n = m = 3$, $p = 2$, $(d_{ij}) = \begin{pmatrix} 0 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 0 \end{pmatrix}$ with optimal value 1. On the left hand side of Figure 1, $N = M$ are represented with nodes of a graph and distances*

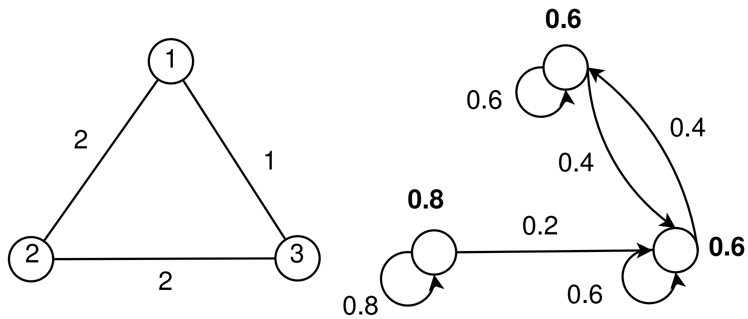

Figure 1: Example from [Elloumi et al., 2004]. Formulation (C)

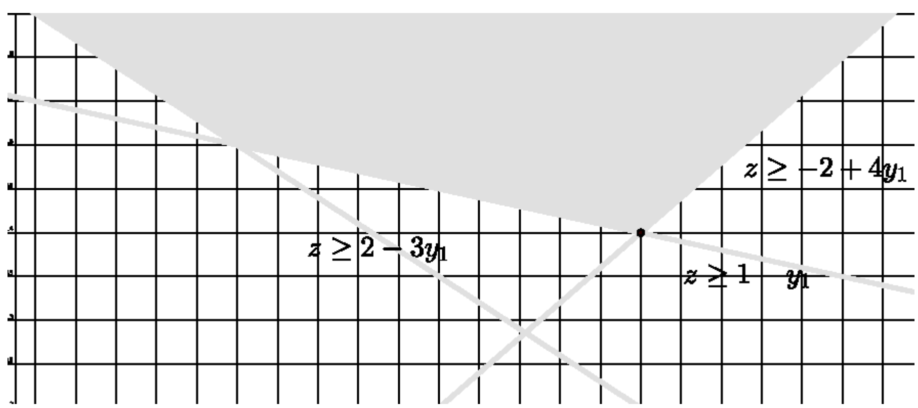

Figure 2: Example from [Elloumi et al., 2004]. Formulation (R)

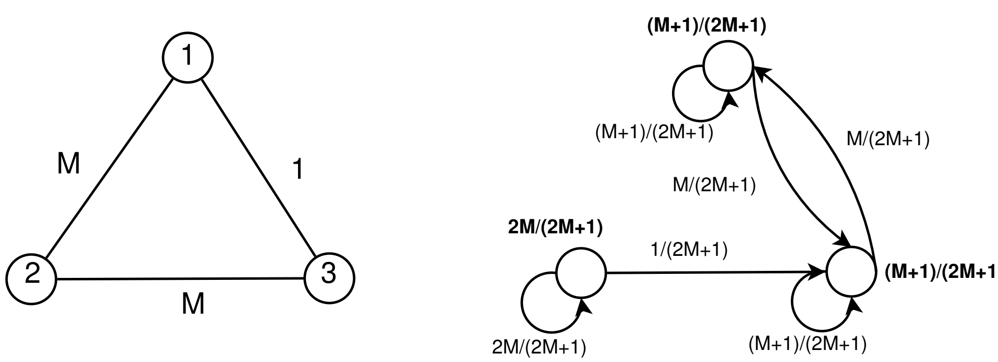

Figure 3: Equivalent instances of Example 2.2. Formulation (C)

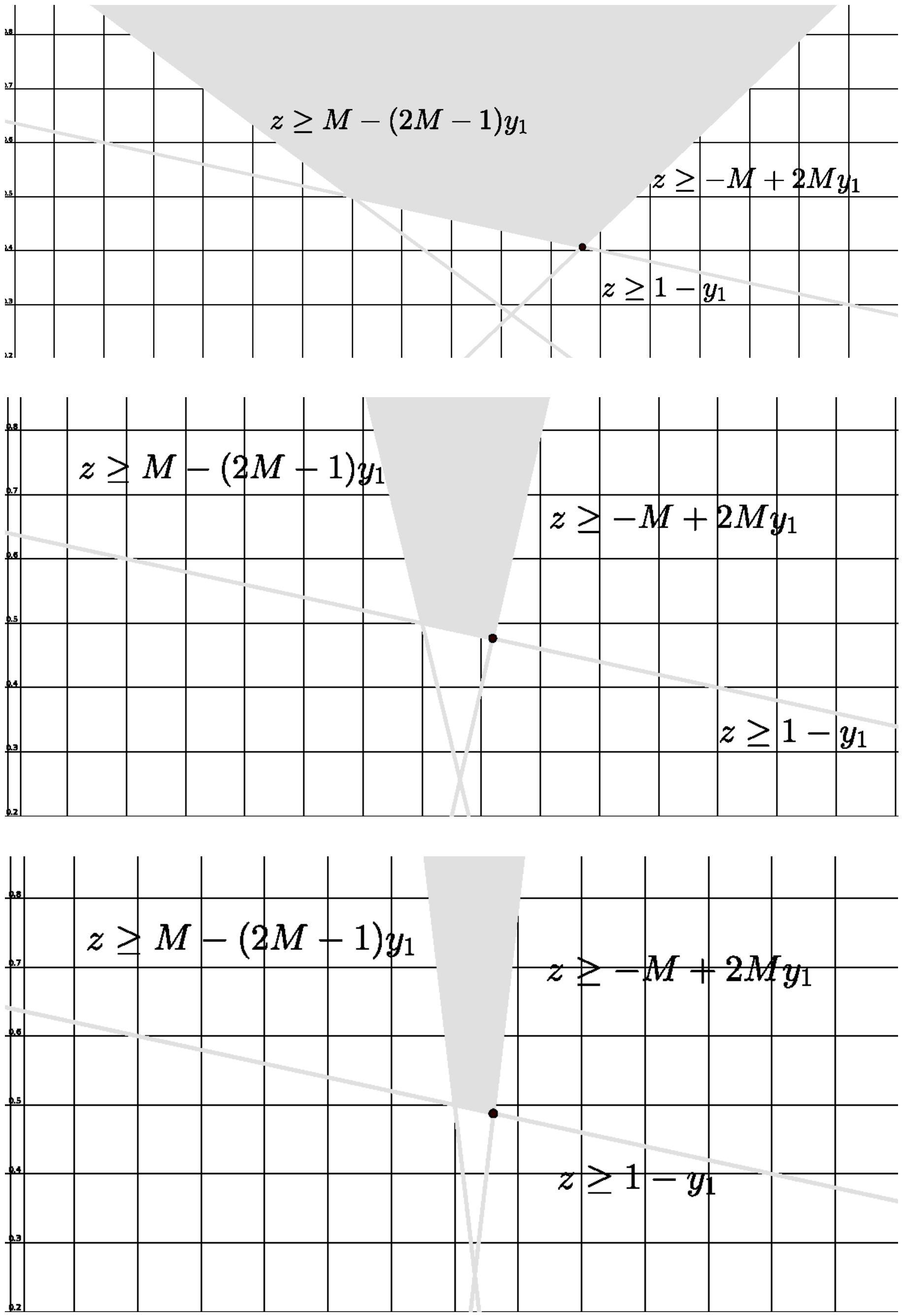

Figure 4: Equivalent instances of Example 2.2. Formulation (R)

are written next to the edges. On the right hand side, the optimal values of the y-variables (bold) and x-variables in the linear relaxation of formulation (C) are represented with numbers associated to arcs. The optimal value is $v(\bar{C}) = 0.2$. *In order to graphically represent how the optimal value* $v(\bar{R})$ *is calculated, consider all the constraints in family* (10), *apply* $y_1 + y_2 + y_3 = 2$ *to eliminate* y_2, *and use* $y_1 = y_3$ *from the symmetry of the instance to eliminate* y_3. *Then the objective function* $z := 0 + (1-0)z_1 + (2-1)z_2$ *can be bounded by*

$$z \geq 1 - y_1, \qquad z \geq 2 - 3y_1, \qquad z \geq -2 + 4y_1.$$

In Figure 2 we represent the feasible region given by these three inequalities. The optimal solution corresponds to the lowest vertex, with optimal value 0.4. *Therefore, this example proved that* $v(\bar{R})$ *can be strictly greater than* $v(\bar{C})$.

In order to deepen this analysis, we introduce some new concepts. First of all, we note that the exact value of each entry of the distance matrix is not relevant to determine the optimal solution of the problem. What actually matters is the value's position once the distances have been sorted.

Proposition 2.1. *For fixed values of* n, m *and* p, *two instances of* pCP *with distance matrices* (d^1_{ij}) *and* (d^2_{ij}) *satisfying*

- $d^1_{ij} = d^1_{i'j'} \iff d^2_{ij} = d^2_{i'j'}$
- $d^1_{ij} < d^1_{i'j'} \iff d^2_{ij} < d^2_{i'j'}$

$\forall i, i' \in N$, $\forall j, j' \in M$ *have the same optimal solutions.*

The result is evident, given the relation of pCP with the set covering problem. Given n, m and p the conditions given in Prop. 2.1 determine an equivalence relation on the set of instances of pCP. We then say that two instances in the same equivalence class are *equivalent*. Some observations follow.

Remark 2.1. *Two equivalent instances can have different optimal values.*

Remark 2.2. *The linear relaxations of two equivalent instances can give different optimal values, even if the optimal (integer) values of the instances are equal.*

We can now extract the following consequence:

Remark 2.3. *The duality gaps of two equivalent instances can be different.*

We then observe that it makes sense to compare the lower bounds on the optimal value of pCP given not only by the linear relaxation of an instance but also by the linear relaxations of equivalent instances. We try to shed light on this point revisiting the example above.

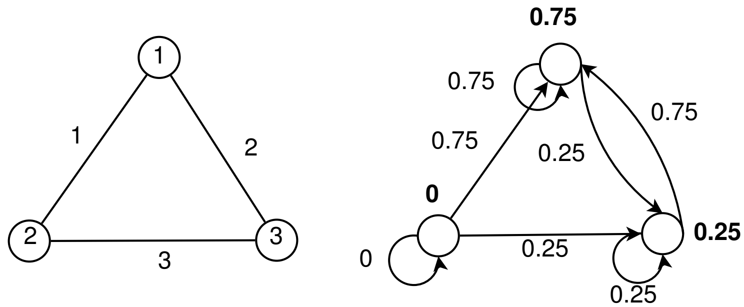

Figure 5: Instance of Example 2.3. Formulation (C)

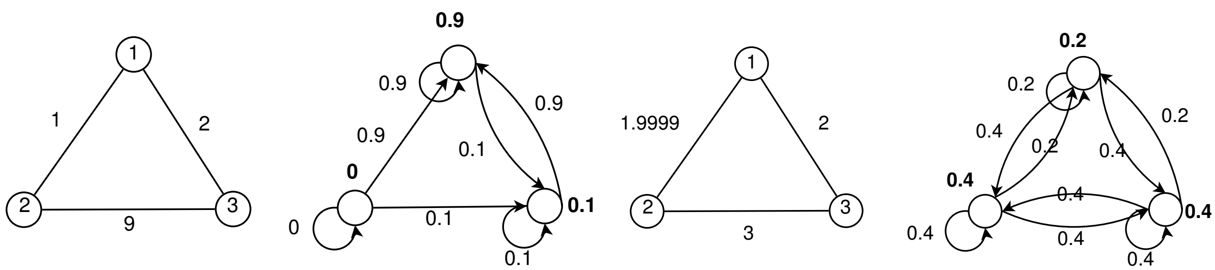

Figure 6: Equivalent instances of Example 2.3. Formulation (C)

Example 2.2. *Starting with the instance given in Example 2.1, in Figure 3 we replace* $d_{12} = d_{21} = d_{23} = d_{32}$ *by* $M > 1$*, so obtaining a family of equivalent instances. On the right hand side we present the optimal solution of formulation* $(\bar{C})$*. Note that making* M *tend to infinity, the values of the* x*-variables tend to*

$$\begin{pmatrix} 1/2 & 0 & 1/2 \\ 1 & 0 & 0 \\ 1/2 & 0 & 1/2 \end{pmatrix}$$

and the optimal value of the linear relaxation $(\bar{C})$ *tends to 0.5, greater than the value obtained in Example 2.1 for the linear relaxation of formulation (R). But of course we can also solve* $(\bar{R})$ *on the new instances. In Figure 4 we represent again the projection of the feasible region on the plane* (y_1, z) *for three different values of* M*. It can be seen that the lowest vertex of the feasible region converges to a point with* $z = 0.5$ *as well. In all these cases the optimal value of the integer formulations remains 1.*

Formulation (A) makes use of this equivalences in a certain sense, since replacing D_k by k is a way to obtain an equivalent instance. Unfortunately, the bounds produced by this “equally separated” distances produce the inverse effect, worsening the bounds one obtains. We still present another example to show the appropriate way to deal with the distances.

Example 2.3. *Again we use* $n = m = 3$ *but now* $p = 1$*. In Figure 5 we show the symmetric distances (with* $d_{ii} = 0\ \forall i$*) and the optimal solution of the linear relaxation* $(\bar{C})$*. The optimal value* *of the instance is 2, obtained choosing 1 as a center, and $v(\bar{C}) = 1.5$. On the other hand, $v(\bar{R}) = 5/3$. If $d_{13} = 2$ remains the same, better bounds can be obtained from $(\hat{C})$ increasing the other two values. By increasing d_{23} from 3 to 9, for example, the optimal solution of the left hand side of Figure 6 is obtained, with a value of 1.8 (in this case $v(\bar{R}) = 2$). But it is also possible to increase d_{12} from 1 to almost 2 (see the right hand side of the figure) to improve the bound from 1.5 to (almost) 1.6. In all these cases the optimal values of the integer formulations remain 2.*

The best choice for the values of the distances requires the knowledge of the optimal value of the problem. Making all the distances that are below $v(C)$ grow up to almost $v(C)$, and making the distances over $v(C)$ tend to infinity, the formulations of the problem will not have duality gap. This value is unknown, but the fact can be used in the resolution of the instances as we will see afterwards.

3 Resolution method based on equivalent instances

Let us start with an instance of pCP and (i) a guaranteed lower bound LB, (ii) a guaranteed upper bound UB and (iii) two intermediate values T_1 and T_2 such that LB $< T_1 < T_2 <$ UB. Using binary variables s_1 and s_2, we introduce the auxiliary formulation given by

$$\begin{aligned}
\text{(AUX)} \quad \min \quad & T_1 + (T_2 - T_1)s_1 + Cs_2 && (16)\\
\text{s.t.} \quad & s_1 \geq 1 - \sum_{\substack{j \in M: \\ d_{ij} \leq T_1}} y_j \quad \forall i \in N && (17)\\
& s_2 \geq 1 - \sum_{\substack{j \in M: \\ d_{ij} \leq T_2}} y_j \quad \forall i \in N && (18)\\
& s_1 \geq s_2 && (19)\\
& \sum_{j \in M} y_j = p && (20)\\
& y_j \in \{0,1\} \qquad \forall j \in M && (21)\\
& s_1, s_2 \in \{0,1\} && (22)
\end{aligned}$$

where C is a very large number and redundant constraints have been removed. This formulation contains $m+2$ variables and at most $2n+2$ linear constraints. Note that this formulation is (R') applied to an instance where all costs less than or equal to T_1 have been replaced by T_1, all costs in $(T_1, T_2]$ have been replaced by T_2 and costs greater than T_2 have been replaced by C. If v(pMP) is at least T_2, (AUX) is the limit of instances that are equivalent to the instance obtained from pMP replacing the largest costs by C (the limit is obtained making the lowest costs tend to either T_1 of T_2). Another equivalent instace will be obtained if v(pMP) is at most T_1. Making use of this relations, we will solve (AUX) to deduce the interval to which v(pMP) belongs.

To solve (AUX) on an instance, let us start solving its linear relaxation. Based on the observations made in the previous section we know that if, during the linear relaxation phase, the lower bound managed by the solver strictly exceeds T_1 (respectively T_2), a better lower bound equal to T_1 (resp. T_2) is available for the optimal value of pCP on the given instance. Then the resolution of the linear relaxation can be stopped and LB updated. Otherwise, the branching phase is required. Then it can be checked, at every node of the branching tree, whether the worst lower bound strictly exceeds T_1 (resp. T_2), to stop the execution and update LB to the new value T_1 or T_2.

Regarding the upper bound UB, if the solver finds a feasible solution of pCP with pCP-objective value less than UB, the execution can be stopped in order to update UB to the new value.

The third and last possibility is to finish optimally solving (AUX), so obtaining an integer optimal solution with optimal value O. In such a case there are only three possibilities:

- $O > T_2$, and then LB can be updated to T_2;
- $O = T_2$, and then UB can be updated to T_2 and LB to T_1;
- $O = T_1$, and then UB can be updated to T_1.

Solving (AUX) iteratively with new values of LB and UB and new intermediate values T_1 and T_2, the interval where the optimal value of pCP is guaranteed to be will be reduced until obtaining the optimal value of pCP and the optimal set of centers, given by the y-values of the last iteration. Although a wide computational study will be done in Section 4, let us show the potential of the method by means of an example.

Example 3.1. *Consider instance pr2392 of the TSP libray, see [Reinelt, 2013]. The size of the instance is $n = m = 2392$. For $p = 30$, the instance has been never solved. A solution of value 1765 was obtained after 1800 seconds in [Gaar and Sinnl, 2022], and a solution of value 1471 required 86400 seconds in [Contardo et al., 2019]. The best lower bound for this problem, also obtained in [Contardo et al., 2019], is 1379. These are the best results to date. Starting with LB=0 and UB=2109 (the maximum value of the distance matrix divided by 8), a new solution of value 1435 was found in 37 seconds. A better solution of value 1402 required less than 1000 seconds.*

Although this approach to solve pCP guarantees optimal solutions, we have carried out two different implementations, that we call Strategy 1 and Strategy 2, trying to accelerate the resolution and to obtain good feasible solutions in shorter times. We need to define the following modified auxiliary problem that makes use of a feasible solution given by a set of p centers $J \subset M$, $|J| = p$.

$$
\begin{aligned}
(\mathrm{AUX}_J) \quad \min \quad & T_1 + (T_2 - T_1)s_1 + Cs_2 \\
\text{s.t.} \quad & (17), (18), (19), (20), (21), (22) \\
& \sum_{j \in J} y_j \geq \lceil p/2 \rceil. \qquad (23)
\end{aligned}
$$

Algorithm 1: Refinement Algorithm R

Input: n, m, (d_{ij}), p, $J \subset M$ ($|J| = p$)

$K = \emptyset$;
repeat
  $U = \max_{i \in N} \min_{j \in J} d_{ij}$;
  Improve=FALSE;
  foreach $j \in J$, $k \notin J \cup K$ *such that* $d_{jk} \leq U$ **do** ;
  if $\{i \in N : \ d_{ij} \leq U\} \subseteq \{i \in N : \ d_{ik} \leq U\}$ **then**
    $J = J \cup \{k\} \setminus \{j\}$;
    $K = K \cup \{j\}$;
    Improve=TRUE;
    exit the loop;
until *Improve=FALSE* ;
return J, U

This problem forces the solutions to contain at least half of the centers of set J. We always take J as the set of indices of the centers in the best available solution. We observed that this constraint, in most of the cases, helps to obtain new better feasible solutions to pCP. Moreover, compared to (AUX), (AUX$_J$) takes a short time. Replacing (AUX) by (AUX$_J$) transforms the approach in heuristic, since the optimal solutions of pCP might not satisfy (23), but our solution strategies will combine the exact approach that uses (AUX) with phases of heuristic search that use (AUX$_J$) on different sets J to reduce the upper bound.

Two other procedures devised to produce better feasible solutions have been incorporated. The first one, named R, is sketched in Algorithm 1. Starting with a feasible solution given by a subset J of M of cardinality p, the objective value of pCP, called U, is calculated. Then, if a candidate that is not in J is better than a center in J, we replace the latter by the former. Here, better means that all the points in a radius U around the center are also in a radius U around the candidate. In order to avoid cycles, an element going out of J is marked (included in K) and never returned to J.

Algorithm 2: Heuristic Search Algorithm H

Input: n, m, (d_{ij}), p, UB, $\bar{y} \in [0,1]^m : \sum_{j \in M} \bar{y}_j = p$

Obtain J, the set of p indices corresponding to the p maximum values in $\bar{y}$ (ties arbitrarily broken);
Set $U = \max_{i \in N} \min_{j \in J} d_{ij}$;
return J, U

The second method to obtain solutions, very efficient in practice, is named H and presented in

Algorithm 2. It is applied in the nodes of the branching tree of the solver. When the node is solved, a linear optimal solution $\bar{y}$ satisfying $\sum_{j\in M}\bar{y}_j = p$ is available. Algorithm H selects the p maximum values in $\bar{y}$ and checks if the solutions is better than the best solution of pCP obtained so far. This checking is done only every I nodes of the search. For this reason we include the parameter I in the forthcoming core algorithms CA and CH.

Algorithm 3: Core Algorithm CA

Input: n, m, (d_{ij}), p, LB, UB, B, MTI, I

BetterBound=FALSE;

$T_1 = \frac{B-LB}{3}$, $T_2 = 2\frac{B-LB}{3}$;

Send to the solver formulation (AUX) on instance $(n, m, (d_{ij}), p)$ and time limit MTI;

Let L be the current lower bound on v(AUX) managed by the solver;

if $L > T_1$ **then**

  Set LB= T_1. BetterBound=TRUE. Stop the solver;

if $L > T_2$ **then**

  Set LB $= T_2$. BetterBound=TRUE. Stop the solver;

if *The solver finds an integer feasible solution* J *with value* $U < UB$ **then**

  Set UB $= U$. BetterBound=TRUE. Stop the solver;

  Call Algorithm R(n, m, (d_{ij}), p, J);

  if $U < UB$ **then**

    UB= U

if *The number of nodes of the branching tree explored by the solver is a multiple of* I **then**

  Obtain the optimal linear solution in the current node of the branching tree $\bar{y}$;

  Call Algorithm H(n, m, (d_{ij}), p, $\bar{y}$) to produce $\bar{J}$ and $\bar{U}$;

  if $\bar{U} < UB$ **then**

    Set UB $= \bar{U}$. Set $J = \bar{J}$. BetterBound=TRUE. Stop the solver;

return *LB, UB, J, BetterBound*

In what follows we will look for the optimal solution and value of pCP inside an interval $[A, B]$ and, depending on the strategy we consider, A (respectively B) can be or not a lower (resp. upper) bound on the optimal value of pCP. The best lower and upper bounds available will be denoted by LB and UB. We never consider values of A less than LB, but the heuristic auxiliary problem (AUX$_J$) can produce values of A greater than LB that are not guaranteed to be lower bounds on the optimal value of pCP. For this reason, sometimes LB will be replaced by the value of A but some other times it will not. Similarly, B will never be fixed to values greater than the optimal value of pCP but, in order to accelerate the search, on occasion B will not be guaranteed to be greater than or equal

Algorithm 4: Core Heuristic Algorithm CH

Input: n, m, (d_{ij}), p, UB, A, MTI, I, J

BetterBound=FALSE;
$T_1 = \frac{UB-A}{3}$, $T_2 = 2\frac{UB-A}{3}$;
Send to the solver formulation (AUX_J) on instance $(n, m, (d_{ij}), p)$ and time limit MTI;
Let L be the current lower bound on $v(\text{AUX}_J)$ managed by the solver;
if $L > T_1$ **then**
    Set $A = T_1$. BetterBound:=TRUE. Stop the solver;
if $L > T_2$ **then**
    Set $A = T_2$. BetterBound:=TRUE. Stop the solver;
if *The solver finds an integer feasible solution* J_1 *with value* $U_1 < UB$ **then**
    Set UB $= U_1$. BetterBound:=TRUE. $J = J_1$. Stop the solver;
    Call Algorithm R(n, m, (d_{ij}), p, J);
    if $U < UB$ **then**
        UB= U
if *The number of nodes of the branching tree explored by the solver is a multiple of* I **then**
    Obtain the optimal linear solution in the current node of the branching tree $\bar{y}$;
    Call Algorithm H(n, m, (d_{ij}), p, $\bar{y}$) to produce J_2 and U_2;
    if $U_2 < UB$ **then**
        Set UB $= U_2$. Set $J = J_2$. BetterBound:=TRUE. Stop the solver;
return *UB, A, J, BetterBound*

to UB. If a feasible solution of pCP is found with value less than UB (regardless of whether it is less than B or not) then UB will be updated to this value. Putting all these elements together, Algorithm 3 (from now on CA) combines the reduction of an interval (with guaranteed lower bound LB and any upper end B below UB) with the heuristic search procedures R and H. CA stops when a better bound (lower or upper) is found. A time limit of MTI seconds is passed to the solver. When the time limit is reached, the resolution of (AUX) is stopped without having reduced the interval [LB,B]. Depending on the value of the parameter BetterBound, CA will be called again (TRUE) or not (FALSE). On the other hand, Algorithm 4 (from now on CH) starts with the best feasible solution available, given by set J, and combines the heuristics with the reduction of an interval (with guaranteed upper bound UB and any lower end A not below LB) that contains the optimal value of (AUX$_J$). It also stops when a better bound is found or MTI seconds have elapsed without any reduction of the current interval [A,UB]. With *lower bound managed by the solver* we mean the worst lower bound in all nodes of the branching tree that the solver obtained, which in fact is a lower bound on v(AUX).

After some preliminary testing, the recursive resolution of CA+CH was implemented in two different ways, Strategy 1 and Strategy 2 (Algorithms 5 and 6, respectively). We distinguish different time limits. With MTI (maximum time per iteration) we denote the limit time passed to the solver: After branching MTI seconds, the solver will stop. This is the meaning of the condition $RunTime > MTI$ in the algorithms. On the other hand, with $TotalRunTime > SMTI$ we give a stop condition that is reached when the time elapsed in all the iteratins of a loop reaches SMTI.

With Strategy 1 we started with lower bound 0 and upper bound large enough, reducing the interval by the iterative application of Algorithm CA with a time limit for all the iterations of SMTI seconds (lines 2-6). Afterwards, the time limit for each iteration was fixed to MTI and the total running time was not limited. If the final interval [LB,UB] of the first phase contained a multiple of 10000, we continued with B equal to the smallest multiple of 10000 greater than LB. Otherwise, we did the same with the multiples of 1000, 100, 10 and 1. Then we iteratively solved Algorithm CA using the best lower bound available in that moment. The value of B was increased by 10000 (resp. 1000, 100, 10, 1) unless a better upper bound UB was found. Once either the application of CA did not produce better bounds the procedure called CH with increasing values of A, reseting A to LB and B to UB every time a better solution was found.

At every iteration the lower and upper bounds will be better. Then there will be y_j-variables that can be fixed to zero in both (AUX) and (AUX$_J$). This is the case of $j \in M$ when, for all values $r \in$[LB,UB], there exists j' that covers at least the same set of points as j using a radius r. Then it is always better replacing j by j' in the set of centers and y_j can be removed from the formulations (row 12 in Strategy 1). Note that the values of r to be checked are those values in the column j of the distances matrix that fall in [LB,UB].

With Strategy 2, we solved alternatingly CA and CH. When using CA we only stopped iterating

Algorithm 5: Strategy 1

```
Input: n, m, (d_ij), p, MTI, SMTI, I
A = 0, UB=max_{i∈N} max_{j∈M} d_ij/8, LB=0;
repeat
    Fix all possible variables y_j = 0;
    B = UB;
    Run CA(n, m, (d_ij), p, LB, UB, B, SMTI, I)
until TotalRunTime > SMTI ;
r = arg max_{k=1,2...} 10^k⌈LB/10^k⌉: 10^k⌈LB/10^k⌉ < UB;
B = 10^r(⌈LB/10^r⌉ − 1);
repeat
    B = B + 10^r;
    repeat
        Fix all possible variables y_j = 0;
        Run CA(n, m, (d_ij), p, LB, UB, B, SMTI, I)
    until RunTime > MTI or UB<B ;
until UB< B ;
repeat
    repeat
        B=UB;
        Fix all possible variables y_j = 0;
        Run CA(n, m, (d_ij), p, LB, UB, B, SMTI, I)
    until RunTime > MTI or UB<B ;
until BetterBound = FALSE ;
repeat
    A = LB;
    Run CH(n, m, (d_ij), p, UB, A, MTI, I, J);
until BetterBound=FALSE ;
return J, UB, LB
```

Algorithm 6: Strategy 2

```
Input: n, m, (d_ij), p, MTI, I, MaxTime, SMaxTime
A = 0, UB= max_{i∈N} max_{j∈M} d_ij/8, LB=0;
repeat
    repeat
        B = UB;
        Run CA(n, m, (d_ij), p, LB, UB, B, MTI, I)
    until LB=UB or BetterBound=FALSE or RunTime > MaxTime ;
    repeat
        A = LB;
        Run CH(n, m, (d_ij), p, UB, A, MTI, I, J)
    until LB=UB or BetterBound=FALSE or RunTime > MaxTime ;
until BetterBound = FALSE or RunTime > MaxTime ;
return J, UB, LB
```

if no better bounds were found. When using CH, we immediately stopped when a better upper bound was found, and passed it to CA. A time limit MTI was used in each iteration but the overall procedure only stopped when, after calling CA+CH or CH+CA there was no improvement.

4 Computational study

The current best computational results for large instances of pCP have been obtained in two articles: [Contardo et al., 2019] and [Gaar and Sinnl, 2022]. The processor used in [Contardo et al., 2019] was an Intel Xeon E5462 2.8GHz and 16GB of RAM. The processor used in [Gaar and Sinnl, 2022] was an Intel Xeon E5-2670v2 2.5GHz with 32 GB of RAM. Our processor was an Intel Xeon CPU E5-2623v3 3.0GHz with 15.5 GB of RAM. According to the web pages consulted, the performance of our processor is similar to the one of [Contardo et al., 2019] and slightly better than the one used in [Gaar and Sinnl, 2022]. The solver we used was FICO Xpress Mosel 64-bit v6.4.1, FICO Xpress v9.2.2 on Ubuntu linux 20.04.6 LTS.

These two other papers used the same set of instances, those from the TSP library available in [Reinelt, 2013]. Each instance is given by $n = m$ points in the plane and rounded Euclidean distances between them. Among the instances, we selected the unsolved ones with n between 3038 and 18512. In nearly all the cases the best lower and upper bounds previously known were obtained (after 86400 seconds of running time) in [Contardo et al., 2019]. Note that the time limit in [Gaar and Sinnl, 2022] was fixed in 1800 seconds. The instances are named with a prefix (e.g., “pcb”) followed by the value of n (e.g., “pcb3038”). Small values of p produced easier instances. For this reason we have checked our algorithms taking relatively large values of p (concretely $p = 20$, 25 and 30). The total number of instances was 27.

		GS 2022		CIK 2019		Strategy 1- $I = 1$			Strategy 1- $I = 100$			Strategy 2- MTI=1200					Strategy 2- MTI=3600				
p	Instance	LB	UB	LB	UB	UB	% gap	Time	UB	% gap	Time	UB	% gap	Time	t1	UB1	UB	% gap	Time	t1	UB1
20	rl11849	2080	2629	2119	2273	2264	6	22000	**2150**	80	18400	2159	74	12500	800	2201	2155	77	49600	810	2201
20	usa13509	43592	56610	44740	46719	44768	99	14600	44952	89	22000	45833	45	7900	3700	45833	**44768**	99	20100	3400	45833
20	d15112	2539	3359	2581	2717	2648	51	17000	2668	36	14800	3128	-	1600	-	-	2669	35	33200	9500	2710
20	d18512	902	1218	912	969	966	5	9600	966	5	9600	1133	-	1800	-	-	**942**	47	44600	2300	963
25	pcb3038	425	545	433	470	**438**	86	4900	**438**	86	3900	440	81	4700	400	453	**438**	86	8700	380	453
25	rl5915	1786	2201	1823	1916	**1825**	98	9900	1837	85	11600	1845	76	5900	1100	1877	1842	80	18300	1100	1877
25	tz6117	1121	1426	1152	1258	**1154**	98	5900	1155	97	10700	1158	94	6500	80	1208	1157	95	22300	80	1204
25	ei8246	421	532	429	461	**433**	88	8400	434	84	4100	441	63	7200	1700	455	**433**	88	30500	1700	455
25	fi10639	1075	1400	1103	1173	1124	70	11500	1135	54	10300	1134	56	4700	2000	1145	**1117**	80	16700	2000	1145
25	rl11849	1823	2506	1838	2099	1894	79	10800	1905	74	12500	**1877**	85	17600	2100	2068	1926	66	17900	3200	1949
25	usa13509	37471	46954	38150	40578	38324	93	12100	**38315**	93	13600	39243	55	12800	7400	39534	38347	92	35500	11100	39996
25	brd14051	693	843	703	737	**712**	74	19100	714	68	13600	714	68	16100	4300	731	716	62	27200	2600	731
25	d15112	2201	2877	2233	2447	2349	46	10400	2338	51	15600	2441	3	6800	5500	2441	**2325**	57	34100	10200	2381
25	d18512	792	1029	795	881	**853**	33	17600	859	26	12300	1133	-	1700	-	-	**853**	33	33000	6400	870
30	pr2392	1351	1765	1379	1471	**1387**	91	3700	**1387**	91	8100	1389	89	4900	37	1435	1389	89	10900	36	1435
30	pcb3038	382	508	386	412	395	65	9200	**393**	73	7900	405	27	7000	210	410	395	65	34000	230	410
30	rl5915	1618	2210	1624	1853	1695	69	8900	1676	77	11700	**1672**	79	9900	230	1778	**1672**	79	17800	230	1778
30	rl5934	1631	2116	1658	1812	**1665**	95	8100	1689	80	12700	**1665**	95	8300	260	1798	**1665**	95	17600	260	1798
30	tz6117	1001	1304	1025	1142	**1027**	98	12200	1029	97	9700	**1027**	98	7800	190	1117	1028	97	17800	190	1117
30	ei8246	384	508	386	412	402	38	9000	399	50	10800	397	58	8700	2301	409	**396**	62	24600	4700	409
30	fi10639	967	1251	974	1017	**985**	74	13800	996	49	14800	1023	-	13500	-	-	995	51	29400	12000	1011
30	rl11849	1649	2255	1641	1855	1767	41	9700	1765	42	16400	1773	38	8800	3100	1822	**1754**	47	18600	3300	1838
30	usa13509	34137	45591	34771	37036	35867	52	16900	**35655**	61	13400	36609	19	12500	7000	36796	35671	60	27800	4100	36918
30	brd14051	618	812	620	668	**644**	50	13400	646	46	12200	656	25	35600	13600	656	656	25	35600	13600	656
30	mo14185	719	936	732	767	753	40	11800	760	20	12400	753	40	14500	11600	753	745	63	16400	6900	750
30	d15112	1994	2662	2009	2254	**2126**	52	14100	2160	38	14500	2199	22	7400	1200	2199	2136	48	43000	1200	2199
30	d18512	712	963	712	786	**759**	36	17200	774	16	15000	1113	-	1400	-	-	765	28	37100	18900	777

Table 1: Computational results

The results obtained using Strategy 1 and Strategy 2 are shown in Table 1. The time limits for Strategy 1 were fixed to MTI=2400 and SMTI=1000. Columns under GS 2022 show the best lower (LB) and upper (UB) bounds obtained in [Gaar and Sinnl, 2022]. Columns under CIK 2019 show the best lower (LB) and upper (UB) bounds obtained in [Contardo et al., 2019]. Columns under $I = 1$ and $I = 100$ show our results for these two values of the parameter I in Algorithm 5. UB is the objective value of the best solution found using this algorithm, % gap shows the percentage of the gap between the previously available best bounds that our algorithm closed. The rounded time in seconds is given under Time. In all the cases, with $I = 1$ and $I = 100$, our Strategy 1 found better solutions in much less time. Upper bounds are marked in bold when the solution is the best one compared with other solutions. That is to say, bold numbers correspond to the best solutions currently known for the instances. Strategy 1 succeeded in 13 cases when $I = 1$ and 6 cases when $I = 100$, including 2 ties. The times shown in the table correspond with the total execution time of the algorithm, not with the time required to obtain the best solutions, that was typically 2400 seconds less. As said in the previous section, Strategy 1 did not make use of the bounds previously known. On the other hand, the lower bounds given in [Contardo et al., 2019] were rarely improved. Using $I = 1$, Strategy 1 run 89 hours in total, and closed in average 64% of the gap between the best lower and upper bounds previously known (calculated using the lower bound of [Contardo et al., 2019]). Using $I = 100$, that invests less time in the heuristic search of better solutions and more time in exploring the branching tree, 62% of the gap was closed in average in 92 hours.

Columns under MTI=1200 and MTI=3600 show our results for these two values of the parameter MTI in Algorithm 6 (Stragegy 2). With MTI=3600, in all the cases Strategy 2 found better solutions than the best solutions previously known in much less time. Again, upper bounds are marked in bold when the solution is the best one compared with other solutions. In 11 of the instances Strategy 2 with MTI=3600 provided the best solution. There is still room for Strategy 2 with MTI=1200. Even if the maximum time for each run of CA and CH was one third of the time when MTI=3600, there were two cases in which the only method that found the best solution was Strategy 2 with MTI=1200. In general, the computational times of MTI=1200 were very small compared to MTI=3600 (approximately one third, as expected), and slightly smaller than the runtimes of Strategy 1. The total times were 60 hours (for MTI=1200) and 201 hours (for MTI=3600). The average closed gap was 47% and 67%,respectively. Under t1 we show the time needed to find the first solution better than the best one previously known. The objective value of such a solution is given under UB1.

For the sake of checkability, we give in the Appendix the best solution obtained for each instance. During the preliminary phase of the study we found two solutions that are better than the ones presented in Table 1. The objective value of these solutions is marked in bold in the Appendix. In particular, the upper bound obtained for instance tz6117 coincides with the lower bound obtained in [Contardo et al., 2019]. This would guarantee the optimality of this solution.

The number of variables fixed to zero in Strategy 1 was typically very small. In many occasions it was simply zero, and it never exceeded some dozens.

5 Concluding remarks

Better results for all the unsolved instances of the p-center problem recently considered in the literature have been obtained by means of a method based on the concept of equivalent instances. Four different strategies were tested and all of them produced the best known solution for some of the instances. Three of the methods always overtook the best solutions previously known in much less computational times.

The method may still be improved with better implementations. Since the instances have a geometrical component, grouping rows/columns of the distance matrix can help to reduce the size of the instance in a previous phase. Also good preprocessings would help to the resolution of large and huge instances. The main idea of our methods could be generalized to be used with different decompositions of the interval that contains the optimal value, that was divided in three equal parts in our search.

Acknowledgements

The author acknowledges that this article is part of the project TED2021-130875B-I00, financied by MCIN/AEI/10.13039/501100011033 and the European Union NextGenerationEU/PRTR and has been also partially supported by research projects PID2022-137818OB-I00 (Ministerio de Ciencia e Innovación, Spain), RED2022-134149-T funded by MCIN/AEI/10.13039/501100011033 and "Data Science and Resources Optimization in Comprehensive Waste Management (DataOpt-Waste)" TED2021-130875B-I00.

Declarations

Conflict of interest. The authors have no relevant financial or non-financial interests to disclose.

References

[Albareda-Sambola et al., 2010] Albareda-Sambola, M., Díaz, J., and Fernández, E. (2010). Lagrangean duals and exact solution to the capacitated p-center problem. *European Journal of Operational Research*, 201(1):71–81.

[Ales and Elloumi, 2018] Ales, Z. and Elloumi, S. (2018). *Combinatorial Optimization: 5th International Symposium, ISCO*, chapter Compact MIP formulations for the p-center problem, pages 14–25. Springer International Publishing.

[Antón-Sánchez et al., 2023] Antón-Sánchez, L., Landete, M., and Saldanha-da-Gama, F. (2023). The discrete p-center problem with upgrading. *Omega*, 119:102894.

[Calık et al., 2019] Calık, H., Labbé, M., and Yaman, H. (2019). *Location Science*, chapter p-center problems, pages 51–65. Springer.

[Calık and Tansel, 2013] Calık, H. and Tansel, B. (2013). Double bound method for solving the p-center location problem. *Computers & Operations Research*, 40(12):2991–2999.

[Chen and Chen, 2009] Chen, D. and Chen, R. (2009). New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems. *Computers & Operations Research*, 36(5):1646–1655.

[Chen and Handler, 1987] Chen, R. and Handler, G. Y. (1987). Relaxation method for the solution of the minimax location-allocation problem in euclidean space. *Naval Research Logistics*, 34(6):775–788.

[Contardo et al., 2019] Contardo, C., Iori, M., and Kramer, R. (2019). A scalable algorithm for the vertex p-center problem. *Computers & Operations Research*, 103:211–220.

[Daskin, 1995] Daskin, M. S. (1995). *Network and Discrete Location: Models, Algorithms, and Applications*. John Wiley & Sons.

[Duran-Mateluna et al., 2023] Duran-Mateluna, C., Ales, Z., Elloumi, S., and Jorquera-Bravo, N. (2023). Robust MILP formulations for the two-stage weighted vertex p-center problem. *Computers & Operations Research*, 159:106334.

[Elloumi et al., 2004] Elloumi, S., Labbé, M., and Pochet, Y. (2004). A new formulation and resolution method for the p-center problem. *INFORMS Journal on Computing*, 16(1):84–94.

[Espejo et al., 2015] Espejo, I., Marín, A., and Rodríguez-Chía, A. (2015). Capacitated p-center problem with failure foresight. *European Journal of Operational Research*, 247(1):229–244.

[Gaar and Sinnl, 2022] Gaar, E. and Sinnl, M. (2022). A scaleable projection-based branch-and-cut algorithm for the p-center problem. *European Journal of Operational Research*, 303(1):78–98.

[Hakimi, 1964] Hakimi, S. L. (1964). Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12(3):450–459.

[Hakimi, 1965] Hakimi, S. L. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations research*, 13(3):462–475.

[Hinojosa et al., 2023] Hinojosa, Y., Marín, A., and Puerto, J. (2023). Dynamically second-preferred p-center problem. *European Journal of Operational Research*, 307(1):33–47.

[Ilhan and Pınar, 2001] Ilhan, T. and Pınar, M. (2001). An efficient exact algorithm for the vertex p-center problem. Technical report, Bilkent University, https://optimization-online.org/?p=9155.

[Kariv and Hakimi, 1979] Kariv, O. and Hakimi, S. (1979). An algorithmic approach to network location problems. *SIAM Journal on Applied Mathematics*, 37:513–538.

[Minieka, 1970] Minieka, E. (1970). The m-center problem. *Siam Review*, 12(1):138–139.

[Özsoy and Pınar, 2006] Özsoy, F. and Pınar, M. (2006). An exact algorithm for the capacitated vertex p-center problem. *Computers & Operations Research*, 33(5):1420–1436.

[Reinelt, 2013] Reinelt, G. (2013). Reinelt. `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/`.

Appendix. Best solutions found

Instance	Value	20 centers									
rl11849	2150	172	209	729	1557	2109	2469	2562	2587	2969	4109
		5806	6423	6948	6987	7352	7920	8416	8779	9553	10053
usa13509	44768	395	583	1749	1832	2546	2555	3645	4817	6045	6725
		6857	7150	7346	11310	12254	12783	12808	12859	13025	13108
d15112	2648	584	835	1226	1493	1809	1824	1986	2840	3502	5510
		5611	7198	8540	10737	12043	12295	12648	14610	14623	14737
d18512	942	1360	1379	4401	4435	5838	6257	7151	7754	8531	10959
		11749	11882	13578	13750	15061	15163	15931	16789	17720	17791

Instance	Value	25 centers									
pcb3038	438	45	131	284	338	426	490	799	941	965	1040
		1307	1435	1448	1624	1706	1719	1996	2022	2070	2296
		2516	2526	2713	2798	2812					
rl5915	1825	620	1331	1400	1638	1834	2933	2981	3190	3477	3622
		3738	4016	4063	4156	4262	4457	4859	4919	5246	5340
		5363	5670	5722	5797	5462					
tz6117	**1152**	348	433	618	646	1126	1309	1465	1502	2198	2703
		2827	3238	3796	3879	4005	4115	4630	4679	4802	4990
		5262	5347	5559	5729	6024					
ei8246	433	201	707	882	1027	1285	2101	2306	2901	2991	3091
		3485	4499	4658	5195	5283	5492	5781	6166	6244	6373
		6604	6805	7567	7869	8087					
fi10639	1117	138	340	2321	2779	3361	4796	5062	5958	5977	6778
		7274	8118	8317	9122	9460	9488	9669	10228	10315	10366
		10539	10551	10566	10619	10630					
rl11849	1877	545	553	2417	3365	3787	3811	4196	4275	5034	5202
		5523	6053	6630	8020	8280	8367	8422	8833	9077	9708
		9936	10886	11432	11607	11835					
usa13509	38315	309	546	1621	1653	1750	1809	3006	3385	3534	4947
		6090	6265	6868	6990	7974	9206	9680	12014	12024	12703
		12930	13015	13027	13030	13289					
brd14051	712	289	1325	1884	3497	3569	3611	5046	5349	5958	6349
		6808	7160	7579	9365	9872	10164	10631	11109	11296	11552
		12470	13016	13373	13880	13921					
d15112	2325	329	998	1486	1691	2381	4507	5625	5684	6355	7325
		8804	9026	9303	10258	10922	11262	11606	12001	12857	13006
		13624	13681	13903	14055	14685					
d18512	**836**	1157	1570	1939	2624	4734	5621	6324	6583	7136	7709
		10258	10515	11143	12746	13216	13940	14240	14646	15020	15853
		16017	16762	17037	17551	17930					

Instance	Value	30 centers									
pr2392	1387	4	31	61	128	145	247	308	335	482	513
		651	733	783	882	964	991	1062	1160	1401	1501
		1533	1605	1633	1742	1848	1870	1937	2085	2174	2286
pcb3038	393	156	182	212	257	305	443	778	816	852	882
		911	933	1184	1261	1416	1551	1636	1693	1790	1852
		1896	2135	2217	2229	2305	2374	2520	2853	2867	2994
rl5915	1672	108	116	175	199	258	317	660	1856	2155	2188
		2205	2244	2896	3170	3198	3307	3351	3462	3603	3947
		3967	4004	4257	4370	4385	4467	4659	5079	5156	5344
rl5934	1665	2	165	899	1094	1529	1624	1750	2241	2337	2479
		2538	2544	2609	2981	3172	3304	3329	3458	3597	3695
		3718	3731	3735	3923	4275	5087	5216	5217	5294	5519
tz6117	1027	291	399	578	831	1070	1222	1318	1558	1574	2070
		2235	2755	2981	3224	3429	3869	3905	4083	4140	4327
		4549	4623	4933	5041	5160	5256	5531	5585	5796	5867
ei8246	396	114	436	671	802	1305	2094	2155	2233	2391	2500
		2871	3091	3923	4132	4230	4554	5274	5278	5341	5810
		5816	5889	6394	6581	6852	6853	7488	7842	7847	8109
fi10639	985	245	951	2042	2147	2283	3221	4513	4918	5434	5905
		6180	6573	6690	7569	8202	8720	8726	9071	9609	9895
		10071	10173	10389	10390	10509	10525	10571	10597	10613	10619
rl11849	1754	425	442	1219	1968	2013	2367	2900	3187	4088	4283
		4342	4701	4770	5101	5632	5695	5800	6465	7119	8056
		8077	8456	8797	8943	9048	9224	9550	10533	11796	11814
usa13509	35655	216	447	914	1459	1875	2555	2670	3119	3176	3220
		3963	4532	5206	6219	7023	7812	8219	8330	8449	10662
		11389	12166	12185	12653	12797	12825	12875	12999	13005	13490
brd14051	644	274	714	1798	2397	2695	3487	4804	4821	5187	5513
		5748	6374	6406	7000	8036	8997	9136	9682	9842	10181
		10357	11062	11323	12122	12542	12601	13262	13491	13866	13921
mo14185	745	6	10	26	797	997	1710	2242	3404	4177	4244
		5024	5765	6295	7250	7685	7712	8099	8104	8668	9034
		9459	9477	9501	9519	10367	11555	11876	12378	175	13279
d15112	2126	436	1434	1523	1626	1915	2483	2653	3739	3916	4428
		4507	5437	5860	6359	6413	6923	7743	8123	8491	8592
		9110	9592	10402	10504	11316	12066	12589	12925	13810	13982
d18512	759	795	1373	1515	2787	3660	5197	5457	6263	6413	6550
		6571	7629	9758	10169	10188	10867	11580	12056	12962	13259
		13781	14478	15584	15710	16239	16517	17556	17684	17804	18294