An efficient Benders decomposition for the p-median problem

Cristian Durán Mateluna, Zacharie Alès, Sourour Elloumi

An efficient Benders decomposition for the p-median problem

Cristian Duran-Mateluna[a,b,c,*], Zacharie Ales[a,b], Sourour Elloumi[a,b]

[a] *UMA, ENSTA Paris, Institut Polytechnique de Paris, 91120 Palaiseau, France.*
[b] *CEDRIC, Conservatoire National des Arts et Métiers, 75003 Paris, France.*
[c] *LDSPS, Industrial Engineering Department, University of Santiago of Chile, 9160000 Santiago, Chile.*

Abstract

The p-median problem is a classic discrete location problem with numerous applications. It aims to open p sites while minimizing the sum of the distances of each client to its nearest open site. We study a Benders decomposition of the most efficient formulation in the literature. We show that the Benders cuts can be separated in linear time. The Benders reformulation leads to a compact formulation for the p-median problem. We implement a two-phase Benders decomposition algorithm that outperforms state-of-the-art methods on benchmark instances by an order of magnitude and allows to exactly solve for the first time several instances among which are large TSP instances and BIRCH instances. We also show that our implementation easily applies to the uncapacitated facility location problem.

Keywords: location; p-median problem; Benders decomposition; integer programming formulation; polynomial separation algorithm

1. Introduction

Discrete location problems aim at choosing a subset of locations from a finite set of candidates in which to establish facilities in order to allocate a finite set of clients. The most common objective for these problems consist in minimizing the sum of the fixed costs of the facilities and the allocation costs of supplying the clients. Within these problems, the *p-median problem* (pMP) is one of the fundamental problems (Laporte et al. (2019)). In the (pMP), we have to choose p locations from the set of candidate sites, no fixed costs are considered and the allocation costs are equal to the distance between clients and sites. More formally, given a set of N clients $\{C_1, ..., C_N\}$ and a set of M potential sites to open $\{F_1, ..., F_M\}$, let d_{ij} be the distance between client C_i and site F_j and $p \in \mathbb{N}$ the number of sites to open. The objective is to find a set S of p sites such that the

*Corresponding author
Email address: cristian.duran@ensta-paris.fr (Cristian Duran-Mateluna)

sum of the distances between each client and its closest site in S is minimized. The (pMP) is an NP-hard problem (Kariv & Hakimi (1979)) and leads to applications where the sites correspond to warehouses, plants, shelters, etc. This includes the contexts of emergency logistics and humanitarian relief (An et al. (2014); Mu & Tong (2020); Takedomi et al. (2022)). Another important application is a particular clustering problem, usually called *k-medoids problem* when the set of clients and sites are identical. In this problem, sub-groups of objects, variables, persons, etc. are identified according to defined criteria of proximity or similarity. (Klastorin (1985); Park & Jun (2009); Marín & Pelegrín (2019); Ushakov & Vasilyev (2021); Voevodski (2021))

A great interest in solving large location problems has led to the development of various heuristics and meta-heuristics in the literature. However, the exact solution of large instances remains a challenge. Some location problems have recently been efficiently solved using the Benders decomposition method within a *branch-and-cut* approach (see e.g., Fischetti et al. (2017); Cordeau et al. (2019); Gaar & Sinnl (2022)). Among them, the *uncapacitated facility location problem* (UFL) is probably the most studied location problem. In the (UFL) the number of sites to be opened is not fixed, but an opening cost is associated with each site.

1.1. Contribution and outline

In this paper, we explore a Benders decomposition for the (pMP). We propose a polynomial time algorithm for the separation of its Benders cuts. We implement an efficient two-phase Benders decomposition algorithm which provides better results than the best exact solution method in the literature `Zebra` (García et al. (2011)). We present our results on about 230 benchmark (pMP) instances of different sizes (up to 238025 clients and sites) satisfying or not the triangle inequality. We finally extend our implementation to solve the (UFL) and present some results.

The rest of the paper is organized as follows. Section 2 presents the literature review of the (pMP). Section 3 describes our Benders decomposition method. Section 4 presents the computational results. In Section 5 we draw some conclusions together with research perspectives.

2. Literature review

The (pMP) was introduced by Hakimi (1964) where the problem was defined on a graph such that a client can only be allocated to an open neighbor site. Since then, exact and approximation methods have been developed to solve the problem, as well as a wide variety of variants and extensions.

The following is a summary of the main formulations of this problem and its state-of-the-art exact solution methods.

2.1. MILP formulations

The classical mathematical programming formulation for the (pMP) was proposed by ReVelle & Swain (1970) who formulated the problem with a binary variable y_j for each sites F_j that takes value of 1 if the site is open and 0 otherwise; and a binary variable x_{ij} that takes value of 1 if client C_i is allocated to site F_j and 0 otherwise. In the following, we denote by $[n]$ the set $\{1, 2, \ldots, n\}$ for any $n \in \mathbb{N}^*$.

$$(F1) \begin{cases} \min \quad \displaystyle\sum_{i=1}^{N}\sum_{j=1}^{M} d_{ij}x_{ij} & & & (1) \\ \text{s.t.} \quad \displaystyle\sum_{j=1}^{M} y_j = p & & & (2) \\ \displaystyle\sum_{j=1}^{M} x_{ij} = 1 & & i \in [N] & (3) \\ x_{ij} \leq y_j & & i \in [N],\ j \in [M] & (4) \\ x_{ij} \geq 0 & & i \in [N],\ j \in [M] & (5) \\ y_j \in \{0,1\} & & j \in [M] & \end{cases}$$

Constraint (2) fixes the number of open sites to p. Constraints (3) ensure that each client is allocated to exactly one site and Constraints (4) ensure that no client is allocated to a closed site. The binary variables x_{ij} can actually be relaxed as in Constraints (5).

An alternative formulation $(F2)$ was proposed by Cornuejols et al. (1980) which orders for each client all its distinct distances to the sites. More formally, for any client $i \in [N]$, let $K_i \leq M$ be the number of different distances from i to any site. Let $D_i^1 < D_i^2 < ... < D_i^{Ki}$ be these distances sorted. Formulation $(F2)$ uses the same y variables as in formulation $(F1)$ and introduces new binary variables z. For any client $i \in [N]$ and $k \in [K_i]$, $z_i^k = 0$ if and only if there is an open site at distance at most D_i^k from client i.

$$(F2)\begin{cases} \min \quad \sum_{i=1}^{N}\Big(D_i^1 + \sum_{k=1}^{K_i-1}(D_i^{k+1}-D_i^k)z_i^k\Big) & & (6)\\ \text{s.t.} \quad \sum_{j=1}^{M} y_j = p & & (7)\\ z_i^k + \sum_{j:d_{ij}\le D_i^k} y_j \ge 1 & i\in[N],\ k\in[K_i] & (8)\\ z_i^k \ge 0 & i\in[N],\ k\in[K_i] & (9)\\ y_j \in \{0,1\} & j\in[M] & \end{cases}$$

Objective (6) minimizes the sum of the allocation distances over all clients. Constraints (8) ensure that variable z_i^k takes the value 1 if there is no site at a distance less than or equal to D_i^k of client i. In that case $(D_i^{k+1} - D_i^k)$ is added to the objective. Otherwise, given the positive coefficients in the objective function, z_i^k takes the value 0. Here again, the binary variables z_i^k can be relaxed as in Constraints (9).

Formulation $(F2)$ can be much smaller than $(F1)$ and both have the same linear relaxation value (Cornuejols et al. (1980)). Formulation $(F1)$ contains $N \times M$ variables x and $1 + N + N \times M$ constraints while $(F2)$ contains $K = \sum_{i=1}^{N} K_i$ variables z and $K + 1$ constraints. As $K \le N \times M$, it follows that $(F2)$ has at most as many variables and constraints as $(F1)$. Usually K is significantly smaller than $N \times M$.

Elloumi (2010) introduced another formulation based on $(F2)$. Given that, by definition, z_i^{k-1} equal to 0 implies that z_i^k is also equal to 0, Constraints (8) can be replaced by (12) and (13).

$$(F3)\begin{cases} \min \quad \sum_{i=1}^{N}\Big(D_i^1 + \sum_{k=1}^{K_i-1}(D_i^{k+1} - D_i^k)z_i^k\Big) & & (10)\\ \text{s.t.} \quad \sum_{j=1}^{M} y_j = p & & (11)\\ z_i^1 + \sum_{j:d_{ij}=D_i^1} y_j \ge 1 & i\in[N] & (12)\\ z_i^k + \sum_{j:d_{ij}=D_i^k} y_j \ge z_i^{k-1} & i\in[N],\ k=2,...,K_i & (13)\\ z_i^k \ge 0 & i\in[N],\ k\in[K_i] & (14)\\ y_j \in \{0,1\} & j\in[M] & (15)\end{cases}$$

Constraints (12) correspond to Constraints (8) for $k = 1$. Constraints (13) ensure that z_i^k takes the value 1 if $z_i^{k-1} = 1$ and if there is no open site at distance D_i^k exactly from i. Formulations $(F2)$ and $(F3)$ use the same set of variables y and z, have exactly the same objective function, and have the same linear relaxation bound (Elloumi (2010)). However, $(F3)$ has much more zeros in the constraint coefficient matrix, which makes it perform significantly better than $(F2)$. Therefore, we consider $(F3)$ for our Benders decomposition.

2.2. Solution methods

The literature contains many solution methods for the (pMP). The main heuristics are presented in the following surveys: Reese (2006); Mladenović et al. (2007); Basu et al. (2015); Irawan & Salhi (2015a). In the following, we only mention the most relevant methods for the exact solution. We refer to Marín & Pelegrín (2019) for details and more references.

Galvão (1980) solved the (pMP) within a *branch-and-bound* framework solving many linear relaxations of sub-problems of size $N = 30$ using formulation $(F1)$. He then devised a method to efficiently obtain good lower bounds instead of optimally solving the relaxed continuous sub-problems.

Avella et al. (2007) designed a *branch-and-cut-and-price* algorithm also based on $(F1)$ that was able to solve instances up to $N = 5535$. Cuts were added based on valid inequalities called lifted odd hole and cycle inequalities. Pricing was carried out by solving a master problem to optimality and using dual variables to price out the variables of the initial problem that were not considered in the master, adding new variables if necessary. The novelty of the approach was that Constraints (3) were also relaxed and incorporated to the master problem when the corresponding column was.

García et al. (2011) considered a cut and column generation algorithm based on formulation $(F2)$. The main idea, also presented in Elloumi (2010) on formulation $(F3)$ and implemented in Elloumi & Plateau (2010), relies on the property that the z variables satisfy $z_i^k \geq z_i^{k+1}$ in any optimal solution of $(F2)$ or its LP relaxation. Therefore, it is enough to solve these problems on a reduced subset of variables z, keep enlarging this subset, and stop as soon as one is sure that the remaining z variables can be set to zero to get an optimal solution. This idea is implemented within a *branch-and-cut-and-price* method that the authors name `Zebra`. It starts with a very small set of z variables and constraints, and adds more when necessary. `Zebra` is an exact solution method that performed well on instances up to $N = 85900$ with large values of p.

The Benders decomposition has been of great interest in the literature. A survey of this method can be found in Rahmaniani et al. (2017). This approach showed good results on discrete

location problems. It was already studied on formulation ($F1$) presented previously and to a similar formulation of the (UFL) in Cornuejols et al. (1980) and Magnanti & Wong (1981). Most recently, Fischetti et al. (2017) propose a Benders decomposition method within a *branch-and-cut* approach to solve efficiently very large size instances of the (UFL). Cordeau et al. (2019) described Benders decomposition for two problems: *the maximal covering location problem* ($MCLP$), which requires finding a subset of facilities that maximizes the amount of client demand covered while respecting a budget constraint on the cost of the facilities; and *the partial set covering location problem* ($PSCLP$), which minimizes the cost of the opened facilities while forcing a certain amount of client demand to be covered. They study a decomposition approach of the two problems based on a *branch-and-Benders-cut* reformulation. Their approach is more efficient when the number of clients is much larger than the number of potential facility locations. Gaar & Sinnl (2022) perform a Benders decomposition on the *p-center problem* (pCP). The (pCP) is closely related to the (pMP). The only difference is that instead of minimizing the sum of the allocation distances, the largest allocation distance is minimized.

3. Benders decomposition for the *(pMP)*

The Benders Decomposition was introduced by Benders (1962). The method splits the optimization problem into a *master problem* and one or several *sub-problems*. The master problem and the sub-problems are solved iteratively and at each iteration each sub-problem may add a cut to the master problem. In this section, we present a Benders decomposition for the *(pMP)* based on formulation ($F3$). We show that there is a finite number of Benders cuts and that they can be separated using a polynomial time algorithm.

3.1. Formulation

For a fixed value of the y variables, the problem decomposes into N sub-problems. Each one computes the allocation distance of a client. In the master problem, we remove all z_i^k variables and we introduce a new set of continuous variables θ_i representing the allocation distance of each client $i \in [N]$:

$$(MP)\begin{cases} \min & \sum_{i=1}^{N} \theta_i & & \\ \text{s.t.} & \sum_{j=1}^{M} y_j = p & & \\ & \theta_i \text{ satisfies } BD_i & i \in [N] & \\ & y_j \in \{0,1\} & j \in [M] & \end{cases}$$

where BD_i is the set of benders cuts associated to client i. This set is initially empty and grows through the iterations.

The sub-problem for each client $i \in [N]$ associated to a feasible solution $\bar{y}$ of (MP) is defined by:

$$(SP_i(\bar{y}))\begin{cases} \min & D_i^1 + \sum_{k=1}^{K_i-1} (D_i^{k+1} - D_i^k) z_i^k & & \\ \text{s.t.} & z_i^1 \geq 1 - \sum_{j:d_{ij}=D_i^1} \bar{y}_j & & \\ & z_i^k - z_i^{k-1} \geq - \sum_{j:d_{ij}=D_i^k} \bar{y}_j & k \in \{2,...,K_i\} & \\ & z_i^k \geq 0 & k \in [K_i] & \end{cases}$$

and its corresponding dual sub-problem is:

$$(DSP_i(\bar{y}))\begin{cases} \max & D_i^1 + v_i^1(1 - \sum_{j:d_{ij}=D_i^1} \bar{y}_j) - \sum_{k=2}^{K_i} v_i^k \sum_{j:d_{ij}=D_i^k} \bar{y}_j & & \\ \text{s.t.} & v_i^k - v_i^{k+1} \leq D_i^{k+1} - D_i^k & k \in [K_i - 1] & \\ & v_i^k \geq 0 & k \in [K_i] & \end{cases}$$

Note that $(SP_i(\bar{y}))$ and $(DSP_i(\bar{y}))$ are feasible for any $\bar{y}$. From an extreme point $\bar{v}$ of $(DSP_i(\bar{y}))$, we obtain the following optimality Benders cut:

$$\theta_i \geq D_i^1 + \bar{v}_i^1(1 - \sum_{j:d_{ij}=D_i^1} y_j) - \sum_{k=2}^{K_i} \bar{v}_i^k \sum_{j:d_{ij}=D_i^k} y_j \tag{16}$$

3.2. Separation problem

The performance of Benders decomposition lies on how we solve the master problem and the sub-problems. In our decomposition, we can have a large number of sub-problems to solve since it is equal to the number of clients at each iteration. Below, we show that the sub-problems can be solved efficiently.

Let $\bar{y}$ either be a solution of the master problem (MP) or of its LP-relaxation. Since (SP_i) minimizes an objective function with non-negative coefficients, the $\bar{z}_i^k$ variables are as small as possible in an optimal solution. Thus, an optimal solution $\bar{z}_i$ for $(SP_i(\bar{y}))$ can be obtained by setting

$$\bar{z}_i^k = \max_{k \in [K_i]} \left(0,\ 1 - \sum_{j : d_{ij} \leq D_i^k} \bar{y}_j\right) \quad i \in [N] \tag{17}$$

We observe that the optimal values of variables z_i^k in $(SP_i(\bar{y}))$ are decreasing when k increases. In order to obtain a dual solution, we identify the last strictly positive term of this sequence.

Definition 1. *Given a solution $\bar{y}$ of the master problem (MP) or of its LP-relaxation. Let $\tilde{k}_i$ be the following index:*

$$\tilde{k}_i = \begin{cases} 0 & \text{if } \sum_{j : d_{ij} = D_i^1} \bar{y}_j \geq 1 \\ \max\{k \in [K_i] : \sum_{j : d_{ij} \leq D_i^k} \bar{y}_j < 1\} & \text{otherwise} \end{cases} \qquad i \in [N]$$

Note that, if $\bar{y}$ is binary, then the allocation distance of client i in the feasible solution $\bar{y}$ is $D_i^{\tilde{k}_i+1}$. Given the indices $\tilde{k}_i$, the optimal value of $SP_i(\bar{y})$ for $i \in [N]$ is:

$$OPT(SP_i(\bar{y})) = \begin{cases} D_i^1 & \text{if } \tilde{k}_i = 0 \\ D_i^{\tilde{k}_i+1} - \sum_{j : d_{ij} \leq D_i^{\tilde{k}_i}} (D_i^{\tilde{k}_i+1} - d_{ij}) \bar{y}_j & \text{otherwise} \end{cases} \tag{18}$$

Furthermore, considering the complementary slackness conditions, an optimal solution $\bar{v}_i$ for $DSP_i(\bar{y})$ can be obtained by setting :

$$\bar{v}_i^k = \begin{cases} D_i^{\tilde{k}_i+1} - D_i^k, & \text{if } k \leq \tilde{k}_i \\ 0, & \text{otherwise} \end{cases} \qquad i \in [N]\ k \in [K_i] \tag{19}$$

Consequently, the Benders cuts (16) can be written as follows:

$$\begin{cases} \theta_i \geq D_i^1 & \text{if } \tilde{k}_i = 0 \\ \\ \theta_i \geq D_i^{\tilde{k}_i+1} - \sum\limits_{j:d_{ij} \leq D_i^{\tilde{k}_i}} (D_i^{\tilde{k}_i+1} - d_{ij}) y_j & \text{otherwise} \end{cases} \tag{20}$$

We observe that these inequalities are the same as those obtained in Cornuejols et al. (1980) and Magnanti & Wong (1981) for (pMP) on formulation $(F1)$. This was quite unexpected, since we used the formulation $(F3)$, even though the master problems are the same in the two decompositions, the sub-problems are different. These same inequalities were also presented in Fischetti et al. (2017) for the (UFL) on a similar formulation to $(F1)$.

3.3. Polynomial separation algorithm

Since a Benders cut can be obtained in polynomial time by computing $\tilde{k}_i$, we use Algorithm 1 to separate Constraints (20). For each client $i \in [N]$, we first compute $\tilde{k}_i$ and $OPT(SP_i(\bar{y}))$ from the current (MP) solution $(\bar{y},\ \bar{\theta})$ (steps 3 and 4) thus updating the upper bound UB of (MP) (step 5). Then, if the value of the allocation distance in the current (MP) solution is underestimated (step 6), we directly construct the corresponding Benders cuts (20) (step 7).

Algorithm 1: Separation algorithm

input :

- Instance data ($[N]$, $[M]$, $[K_i]$, distances $D_i^0, ..., D_i^{K_i}$ and d_{ij} for each $i \in [N]$, $j \in [M]$)
- Current (MP) solution $(\bar{y},\ \bar{\theta})$

output :

- Upper bound of (MP).

1 $UB \leftarrow 0$
2 **for** $i \in [N]$ **do**
3 Compute $\tilde{k}_i$ with Algorithm 2
4 Compute $OPT(SP_i(\bar{y}))$ through (18)
5 $UB \leftarrow UB + OPT(SP_i(\bar{y}))$
6 **if** $\bar{\theta}_i < OPT(SP_i(\bar{y}))$ **then**
7 Add the corresponding cut (20) to (MP)
8 **return** UB

The memory management of the distances between the clients and the sites can be challenging for large-scale instances. For example, in the work of Cornuejols et al. (1980); Magnanti & Wong (1981); Fischetti et al. (2017), the increasingly ordered distances $\{d_{ij}\}_{j\in[M]}$ of each client $i \in [N]$ are considered as an input.

In our approach, the complexity of Algorithm 1 is determined by the computation of index $\tilde{k}_i$. We show that it can be computed in $O(M)$ by Algorithm 2. This algorithm takes as an input a vector $S_i \in [M]^M$ such that S_{ir} is the r^{th} closest site to client $i \in [N]$. Hence, $d_{i\mathcal{S}_{ir}}$ is the distance between i and its r^{th} closest site. Afterwards, given index $\tilde{k}_i$, steps 4 and 5 of Algorithm 1 can be computed in $O(M)$ and $O(1)$, respectively. Then, considering the N clients, a complexity in $O(NM)$ is obtained for Algorithm 1. As in García et al. (2011), the distances $\{d_{ij}\}_{j\in[M]}$ of each client $i \in [N]$ are calculated as they are needed.

Consequently, the $N \times M$ matrix $\mathcal{S}$ is built only once in a preprocessing step in $O(NMlog(M))$ using the QuickSort algorithm. The computation time of this matrix may be longer than the runtime of the solution method depending on the size of the instances. For example, for instances with 5000, 13000, 27000, 85000 clients and sites, the computer described below in Section 4 builds the matrix on average in 5, 25, 90 and 1100 seconds, respectively. Furthermore, to reduce the memory requirements for storing this matrix $\mathcal{S}$, we considered the fact that a client will never be allocated to one of its furthest p sites. Therefore, the size of the matrix $\mathcal{S}$ is reduced to $N \times (M - p)$.

Algorithm 2: Computing $\tilde{k}_i$

input :

- Instance data ($[N]$, $[M]$, $[K_i]$, $\mathcal{S}$ matrix, and distances d_{ij} for $i \in [N]$, $j \in [M]$)
- Current (MP) solution $\bar{y}$
- $i \in [N]$

output :

- The index $\tilde{k}_i$ associated to $\bar{y}$

1 $\tilde{k}_i \leftarrow 0$
2 $r \leftarrow 1$
3 $val \leftarrow 1 - \bar{y}_{\mathcal{S}_{ir}}$
4 **while** $val > 0$ *and* $r < M$ **do**
5 **if** $d_{i(\mathcal{S}_{i(r+1)})} > d_{i\mathcal{S}_{ir}}$ **then**
6 $\tilde{k}_i \leftarrow \tilde{k}_i + 1$
7 $r \leftarrow (r + 1)$
8 $val \leftarrow val - \bar{y}_{\mathcal{S}_{ir}}$
9 **return** $\tilde{k}_i$

3.4. Compact Benders reformulation

The Bender cuts (20) lead to the following compact formulation for (pMP):

$$
(F4)\begin{cases}
\min \sum_{i=1}^{N} \theta_i & \\
\text{s.t.} \quad \sum_{j=1}^{M} y_j = p & \\
\theta_i \geq D_i^1 & i \in [N] \qquad (21)\\
\theta_i \geq D_i^{k+1} - \sum_{j:d_{ij} \leq D_i^k} (D_i^{k+1} - d_{ij}) y_j & i \in [N],\ k \in [K_i - 1] \qquad (22)\\
y_j \in \{0,1\} & j \in [M]
\end{cases}
$$

Constraints (22) ensure that each variable θ_i is larger than D_i^{k+1} unless a site is opened at a smaller distance than D_i^k from i. This formulation $(F4)$ has $(N + M)$ variables which is less than $(F2)$ and $(F3)$ but it has the same number of constraints. Nevertheless, the constraint matrix is roughly as dense as $(F2)$ and it has the same continuous relaxation.

Table 1 presents the results of four formulations of the (pMP) on five instances from OR-Library described in Section 4.1. A time limit of 600 seconds is considered. For each formulation the relative optimality gap and the runtime in seconds are presented.

INSTANCE				F1		F2		F3		F4	
name	N=M	p	OPT	*gap*	$t(s)$	*gap*	$t(s)$	*gap*	$t(s)$	*gap*	$t(s)$
pmed26	600	5	**9917**	0%	228	0%	40	0%	8	0%	57
pmed31	700	5	**10086**	0%	282	0%	36	0%	7	0%	58
pmed35	800	5	**10400**	0%	527	0%	104	0%	9	0%	95
pmed38	900	5	**11060**	74,1%	600	0%	75	0%	19	0%	115
pmed39	900	5	**11069**	10,7%	600	0%	66	0%	19	0%	105
pmed40	900	5	**12305**	0%	579	0%	60	0%	10	0%	104

Table 1: Comparison between different (pMP) formulations with a time limit of 600 seconds.

Results in Table 1 confirm the expected performance between formulations $(F1)$, $(F2)$ and $(F3)$ already described in Section 2.1. Moreover, we see that $(F4)$ takes more time than $(F2)$ and $(F3)$.

3.5. Decomposition algorithm implementation

To improve the performance of the Benders decomposition, we implement a two-phase algorithm. Let $(\overline{MP})$ be the master problem without the integrity constraints. We solve first the Benders decomposition for $(\overline{MP})$ (*Phase 1*). Then, we add the integrity constraints to the obtained master problem to solve it through a *branch-and-cut* algorithm (*Phase 2*).

3.5.1. Phase 1: Solving the linear relaxation of the master problem

Phase 1 is summarized in Algorithm 3. The current master problem $\overline{MP}$ is solved at step 4 through a linear programming solver and provides a candidate solution $(\bar{y}, \overline{\theta})$ while the sub-problems are solved at steps 2 and 6 using Algorithm 1. To enhance the performance this phase includes the following improvements:

- **Initial solution**: Providing a good candidate solution to the initial $(\overline{MP})$ can significantly reduce the number of iterations. Consequently, as García et al. (2011), we compute a first solution using the `PopStar` heuristic (Resende & Werneck (2004)) which, to the best of our knowledge, is the best heuristic for the (pMP). `PopStar` is a hybrid heuristic that combines elements of several metaheuristics. It uses a multi-start method in which a solution is built at each iteration as in a GRASP algorithm. It is followed by an intensification strategy, with a tabu search and a scatter search. And in a post-optimization phase, they use the concept of multiple generations, a characteristic of genetic algorithms. The solution y^h provided by this heuristic and its objective value UB^h are inputs of Algorithm 3. The latter is used to initialize UB^1 at step 1.

- **Rounding heuristic**: Since in Phase 1 most of the solutions provided by $(\overline{MP})$ are fractional, we use a primal heuristic to try to improve the upper bound of the problem. At each iteration we open the sites associated to the p largest values of $\bar{y}$ (steps 7 to 11 in Algorithm 3).

The objective value of $(\overline{MP})$ and the sub-problem optimal value $OPT(SP)$ allow us to update the optimality bounds on the value of the linear relaxation of the problem. In each iteration the rounding heuristic tries the improve UB^1. The iterative algorithm is terminated when no more violated Benders cuts are found for the current solution $\overline{y}$ and hence we have obtained the value of the linear relaxation of the problem.

Algorithm 3: Phase 1 - Solving $(\overline{MP})$

input :

- Instance data (N, M, p , Distances $D_i^0, ..., D_i^{K_i}$ and d_{ij} with $i \in [N]$, $j \in [M]$)
- Heuristic (pMP) solution y^h with value UB^h

output :

- Lower bound LB^1 and a feasible integer solution y^1 with value UB^1

$(y^1, UB^1) \leftarrow (y^h, UB^h)$

Use Algorithm 1 to generate violated Benders cuts associated with y^h to $(\overline{MP})$

while *violated cut has been found* **do**

  $(\bar{y}, \bar{\theta}) \leftarrow$ Solve $(\overline{MP})$

  $LB^1 \leftarrow \sum_{i=1}^{N} \bar{\theta}_i$

  Use Algorithm 1 to generate violated Benders cuts associated with $\bar{y}$ to $(\overline{MP})$

  if $\bar{y}$ *is fractional* **then**

    $(y^r, UB^r) \leftarrow$ Get a rounded heuristic solution from $\bar{y}$

    if $UB^r < UB^1$ **then**

      $UB^1 \leftarrow UB^r$

      $y^1 \leftarrow y^r$

return LB^1, y^1, UB^1

3.5.2. Phase 2: Solving the master problem with branch-and-Benders-cut approach

Once the continuous relaxation of (MP) is solved by Phase 1, we add the integrity constraints on variables y and we use a *branch-and-cut* algorithm to solve the problem. We solve the sub-problems at each node which provides an integer solution in order to generate Benders cuts. The solution of the sub-problems is performed through callbacks which is a feature provided by mixed integer programming solvers. In order to enhance the performance of Phase 2, we implement the following improvements:

- **Constraint reduction**: At the end of Phase 1, most of the generated Benders cuts are not saturated by the current fractional solution. We remove most of them to reduce the problem size. The cuts of a client i are related to indexes $\tilde{k}_i$ obtained at different iterations. Let $\hat{k}$ be the highest of these indexes associated with a saturated constraint. We remove all constraints of client i which associated index is higher than $\hat{k}$. This reduction performs better than removing all unsaturated constraints.

- **Reduced cost fixing**: At the end of Phase 1, given the bounds LB^1 and UB^1, we can perform an analysis of the reduced costs $\overline{rc}$ of the last fractional solution $\overline{y}$ provided by Algorithm 3. For any site j such that $LB^1 + \overline{rc}_j > UB^1$, y_j can be set to 0. Similarly, for any site j such that $LB^1 - \overline{rc}_j > UB^1$, y_j can be fixed to 1. We computationally observed that that these rules are efficient in instances where p is small (i.e., when the ratio p/M is less than 20%). At higher values of p, there may exist many equivalent solutions. Therefore, opening or closing a site often does not have a strong impact on the objective value.

4. Computational study

In this section, we compare the results of our Benders decomposition method with those of the state-of-the-art methods described in Section 2.2.

4.1. Benchmark instances

We study the same instances used in García et al. (2011) that is the p-median instances from OR-Library (Beasley (1990)) and TSP-Library (Reinelt (1991)). In all these instances, the sites are at the same location as the clients and thus $N = M$. The set of OR-Library contains instances with 100 to 900 clients, and the value of p is between 5 and 500. The set of TSP-Library selected contains between 1304 and 238025 clients. Following previous works such as García et al. (2011), all client points are given as two-dimensional coordinates, and the Euclidean distance rounded down to the nearest integer is used as distance.

Another set of symmetric instances that satisfy triangle inequality are the BIRCH instances, usually solved by heuristics algorithms (see e.g Hansen et al. (2009); Avella et al. (2012); Irawan et al. (2014)). These instances consist of p clusters of two-dimensional data points generated in a square. We considered instances with sizes from 10000 to 20000 points and from 25000 to 89600 points for two types of instances, named Type I and Type III. These instances were kindly provided by the authors of Avella et al. (2012). For the comparison we have considered the results presented in Avella et al. (2012), in which it is proposed an aggregation heuristic. We denote this heuristic as `AvellaHeu`. We considered for large BIRCH instances the results presented in Irawan & Salhi (2015b), in which is proposed a hybrid heuristic combining aggregation and variable neighborhood search. We denote this heuristic as `IrawanHeu`.

We also consider the RW instances originally proposed by Resende & Werneck (2004) with the `PopStar` heuristic. They correspond to completely random distance matrices. The distance between

each site and each client is an integer value taken uniformly in the interval $[1, n]$. Moreover, the distance between client i and site j is not necessarily equal to the distance between site j and client i. Four different values of $N = M$ are considered: 100, 250, 500, and 1000.

Finally, we include in our experimentation the ODM instances which were introduced by Briant & Naddef (2004) and are used in Avella et al. (2007) with a *branch-and-cut-and-price* algorithm. We name this algorithm `AvellaB&C`. These instances correspond to the *optimal diversity management problem* which can be treated as a p-median problem in which certain allocations between clients and sites are not allowed. For this problem there exist instances with N equal to 1284, 3773, and 5335. It was already observed by Avella et al. (2007) that instances with N equal to 1284 and 5335 are easy to solve. Therefore, we have only considered the instances with the value of $N = 3773$.

4.2. Technical specifications

Our study was carried out on an Intel XEON W-2145 processor 3,7 GHz, with 16 threads, but only 1 was used, and 256 GB of RAM. IBM ILOG CPLEX 20.1 was used as *branch-and-cut* framework. We apply the described separation algorithm in the GenericCallback of CPLEX, which gets called whenever a feasible integer solution is found. We set the absolute tolerance to the best integer objective (EpGap) to 10^{-10}, and the tolerance to the best remaining node, also called absolute MIP gap (EpAGap), to 0.9999. Considering that our Benders decomposition can easily find feasible solutions, we have set the MIP emphasis switch to BestBound in order to prove the optimality as fast as possible. We set the branch-up-first parameter BRDIR to 1, since this tends to produce branching trees with fewer nodes. We use a time limit of 10 hours for Phase 2, indicated by TL in the tables when this is reached.

We were able to run the `Zebra` and `PopStar` methods on our computer. `Zebra` code was provided by the authors of García et al. (2011) and `PopStar` code is available online[1]. On the other hand, we do not report an updated time for the heuristic algorithms: `AvellaB&C` was originally carried out on a Compaq EVO W4000 Personal Computer with Pentium IV-1.8 GHz processor with 1 GB of RAM using the LP solver IBM ILOG CPLEX 8.0 with a time limit of 100 hours per instance (indicated by TL2 in the corresponding table), `AvellaHeu` was carried out on an Intel Core 2 Quad CPU 2.6 GHz workstation with 4 GB of RAM with a single core, and `IrawanHeu` was carried out on a PC Intel Core i5 CPU 650@ 3.20 GHz of processor with 4 GB of RAM. In order to compare more objectively the computation times of our method with the ones of these approaches, we consider

[1] http://mauricio.resende.info/popstar/

the benchmark score of each computer from the geekbench website[2]. To obtain a solution time for these three methods which is closer to the one that would have been obtained if we had executed them, one can multiply the time reported in their articles by the ratio between the score of our computer and the score of the computer on which they were obtained (i.e., 3 for `AvellaHeu`, 2.5 for `IrawanHeu`, and 6 for `AvellaB&C`).

4.3. Performance analysis

The results for the different instances are presented below. The information in the tables is organized as follows:

- Instance data
  - name: name of the instance.
  - $N = M$: size of the instance (number of clients equal to the number of sites).
  - p: number of sites to open.
  - OPT/BKN: optimal value of the instance (in **bold**) if it is known or the best-known solution value obtained given the time limit, otherwise. If the value is underlined, it means that it is the first time the instance is solved to optimality or that we improve the best-known value.
- Our Phase 1 results:
  - LB^1: lower bound of the (pMP) obtained at the end of Phase 1.
  - UB^1: upper bound of the (pMP) obtained at the end of Phase 1.
  - T^1: CPU time in seconds required to complete Phase 1.
- Our Phase 1 + Phase 2 results:
  - *gap*: relative optimality gap between the lower and upper bound obtained at the end of Phase 2.
  - *iter*: number of total iterations required for the Benders decomposition, i.e., the number of times a fractional solution or an integer solution was separated in Phase 1 and Phase 2, respectively.
  - *nodes*: number of the explored nodes of the *branch-and-cut*.

[2]https://browser.geekbench.com

- T^{tot}: the total CPU time in seconds required to exactly solve the instance.

- `Zebra`, `AvellaHeu`, `IrawanHeu`, `PopStar`, and `AvellaB&C` results:
  - gap / UB^h: relative optimality gap when available or the solution value obtained at the end of the corresponding method.
  - T: total CPU time time in seconds required to complete the algorithms of García et al. (2011), Avella et al. (2012), Irawan & Salhi (2015b), Resende & Werneck (2004) or Avella et al. (2007) respectively. A diamond (⧫) means that the computer ran out of memory while solving the problem.

- Average total time
  - the average total time by our method and `Zebra` is presented in the corresponding tables. This average is calculated considering only the instances where both methods solve the instances to optimality.

OR-Library and TSP-Library instances

Similarly to `Zebra`, we reach the optimal value of all the OR-Library instances in few seconds. Consequently, we only present the results on TSP-Library instances in Tables 2, 3, 4 and 5 for small, medium, large, and huge instances, respectively. Our method reaches the optimal solution in most instances. Very good LB^1 and UB^1 bounds are quickly found at the end of Phase 1.

In Tables 2 and 3 we can observe that 10 small and medium instances are not solved optimally by `Zebra` due to a lack of memory or for reaching the time limit. However, our method does not face any memory problem and only 2 small and 2 medium instances reach the time limit of 10 hours with an optimality gap lower than $0,1\%$.

Regarding the large and huge instances in Tables 4 and 5, we solve 57 out of the 68 instances whereas `Zebra` only solves 16 instances due to a lack of memory. For the huge instances, the rounding heuristic step of Phase 1, the reduced cost fixing step, and the constraint reduction step of Phase 2 are not used as they take too much time. Nevertheless, we use the rounding heuristic once at the end of Phase 1 to update UB^1. Moreover, for the huge instances, we use a randomly generated solution instead of `PopStar` which takes a long time. We can provide for the first time the optimal values for 7 instances with $N = M = 115455$ and 10 instances with $N = M = 238025$.

INSTANCE				PHASE 1			PHASE 1 + 2				Zebra	
name	$N = M$	p	$OPT/$ BKN	LB^1	UB^1	T^1	*gap*	*iter*	*nodes*	T^{tot}	*gap*	T
rl1304	1304	5	**3099073**	3099073	3099073	2,70	0%	9	0	2,8	0%	1233
rl1304	1304	10	**2134295**	2131788	2134295	2,90	0%	12	160	15,4	0%	1060
rl1304	1304	20	**1412108**	1412108	1412108	2,25	0%	8	0	2,3	0%	61
rl1304	1304	50	**795012**	795012	795012	1,46	0%	9	0	1,5	0%	8,3
rl1304	1304	100	**491639**	491507	491788	0,90	0%	19	37	2,4	0%	3,6
rl1304	1304	200	**268573**	268573	268573	0,35	0%	11	0	0,5	0%	0,9
rl1304	1304	300	**177326**	177318	177339	0,31	0%	12	0	0,5	0%	0,5
rl1304	1304	400	**128332**	128332	128332	0,23	0%	10	0	0,2	0%	0,1
rl1304	1304	500	**97024**	97018	97034	0,27	0%	14	0	0,4	0%	0,2
fl1400	1400	5	**174877**	174877	174877	0,82	0%	7	0	0,9	0%	245
fl1400	1400	10	**100601**	100601	100601	0,40	0%	6	0	0,4	0%	72
fl1400	1400	20	**57191**	57191	57191	0,38	0%	8	0	0,4	0%	10
fl1400	1400	50	**28486**	28486	28486	0,36	0%	8	0	0,4	0%	2,5
fl1400	1400	100	**15962**	15961	15962	0,82	0%	12	5	2,1	0%	5,0
fl1400	1400	200	**8806**	8793	8815	0,66	0%	20	570	26,9	0%	305
fl1400	1400	300	**6109**	6092	6157	0,76	0%	28	12599	385	9%	TL
fl1400	1400	400	**4648**	4636	4659	0,56	0%	51	6716041	32655	8%	TL
fl1400	1400	500	3764	3756	3773	0,53	0,09%	44	4987858	TL	8%	TL
u1432	1432	5	**1210126**	1210126	1210126	1,65	0%	7	0	1,8	0%	324
u1432	1432	10	**849759**	849759	849759	3,47	0%	7	0	3,5	0%	71
u1432	1432	20	**588766**	588720	588767	3,86	0%	12	3	5,8	0%	18
u1432	1432	50	**362072**	361724	362072	4,28	0%	25	1493	161,0	0%	128
u1432	1432	100	**243793**	243758	243850	1,86	0%	12	0	3,0	0%	7,3
u1432	1432	200	**159887**	159867	160084	0,72	0%	13	8	2,0	0%	1,9
u1432	1432	300	**123689**	123674	123876	0,63	0%	15	0	1,0	0%	2,2
u1432	1432	400	103979	103411	104102	0,91	0,11%	29	5806518	TL	0%	TL
u1432	1432	500	**93200**	93200	93200	0,16	0%	8	0	0,3	0%	0,1
vm1748	1748	5	**4479421**	4479421	4479421	2,36	0%	7	0	2,5	0%	4955
vm1748	1748	10	**2983645**	2983048	2983645	4,67	0%	14	11	10,9	0%	1364
vm1748	1748	20	**1899680**	1899588	1899681	4,76	0%	15	5	9,5	0%	309
vm1748	1748	50	**1004331**	1004325	1004339	2,23	0%	12	0	2,8	0%	21
vm1748	1748	100	**636515**	636418	636541	1,57	0%	15	3	3,2	0%	13
vm1748	1748	200	**390350**	390350	390350	0,79	0%	11	0	0,8	0%	1,6
vm1748	1748	300	**286039**	286037	286080	0,63	0%	13	0	1,0	0%	1,0
vm1748	1748	400	**221526**	221523	221545	0,53	0%	13	0	0,8	0%	1,0
vm1748	1748	500	**176986**	176977	177103	0,54	0%	14	0	0,8	0%	0,5
					Average total time					9		320

Table 2: Results on small TSP instances for our method and `Zebra` on our computer. TL=36000 seconds. The average total time is calculated with the instances in which both methods solve the instances to optimality.

INSTANCE				PHASE 1			PHASE 1 + 2				Zebra	
name	$N = M$	p	$OPT/$ BKN	LB^1	UB^1	T^1	*gap*	*iter*	*nodes*	T^{tot}	*gap*	T
d2103	2103	5	**1005136**	1005136	1005136	4	0%	8	0	4	0%	4268
d2103	2103	10	**687321**	687264	687321	8	0%	11	7	12	0%	1085
d2103	2103	20	**482926**	482798	482926	9	0%	12	71	18	0%	448
d2103	2103	50	302219	301592	302219	18	0,04%	34	149711	TL	0%	7203
d2103	2103	100	**194664**	194408	194994	17	0%	39	95993	10363	0%	16022
d2103	2103	200	**117753**	117736	117778	2	0%	16	3	5	0%	5
d2103	2103	300	**90471**	90424	90510	2	0%	21	0	3	0%	29
d2103	2103	400	**75324**	75291	75425	1	0%	19	2	4	0%	6209
d2103	2103	500	**64006**	63952	64315	1	0%	27	477	8	0%	2568
pcb3038	3038	5	**1777835**	1777665	1777835	29	0%	12	13	55	0%	TL
pcb3038	3038	10	**1211704**	1211704	1211704	18	0%	8	0	18	0%	19526
pcb3038	3038	20	**839494**	839233	839499	50	0%	18	188	146	0%	7329
pcb3038	3038	50	**506339**	506205	506339	24	0%	12	194	85	0%	1134
pcb3038	3038	100	**351500**	351404	351648	28	0%	22	390	96	0%	346
pcb3038	3038	150	**280128**	280058	280423	16	0%	24	640	269	0%	148
pcb3038	3038	200	**237399**	237328	237578	11	0%	13	734	84	0%	130
pcb3038	3038	300	**186833**	186793	186906	6	0%	19	73	17	0%	60
pcb3038	3038	400	**156276**	156268	156307	3	0%	10	0	6	0%	22
pcb3038	3038	500	**134798**	134774	134866	2	0%	17	0	4	0%	17
fl3795	3795	5	**1052627**	1052627	1052627	14	0%	7	0	14	∞	⧫
fl3795	3795	10	**520940**	520940	520940	8	0%	8	0	8	0%	4410
fl3795	3795	20	**319722**	319722	319722	6	0%	11	0	6	0%	2671
fl3795	3795	50	**150940**	150940	150940	5	0%	13	0	5	0%	193
fl3795	3795	100	**88299**	88299	88299	6	0%	12	0	6	0%	45
fl3795	3795	150	**65868**	65840	65904	14	0%	52	1833	221	0%	1825
fl3795	3795	200	**53928**	53913	54013	11	0%	68	46730	2633	0%	2501
fl3795	3795	300	**39586**	39578	39661	6	0%	47	127800	2548	0%	3061
fl3795	3795	400	**31354**	31348	31472	5	0%	57	14566	559	0%	527
fl3795	3795	500	**25976**	25976	25976	6	0%	15	0	6	0%	2
rl5934	5934	10	**9792218**	9786688	9792218	405	0%	17	329	1864	∞	⧫
rl5934	5934	20	**6716215**	6713214	6716228	437	0%	27	1381	14725	∞	⧫
rl5934	5934	50	4029999	4026936	4029999	362	0,03%	32	8068	TL	0%	TL
rl5934	5934	200	**1805530**	1805030	1807763	67	0%	27	1353	967	0%	3816
rl5934	5934	300	**1392419**	1392304	1392709	38	0%	15	25	58	0%	235
rl5934	5934	400	**1143940**	1143649	1145342	20	0%	33	1809	303	0%	1110
rl5934	5934	500	**972799**	972741	973712	17	0%	20	117	44	0%	70
rl5934	5934	600	**847301**	847233	847769	12	0%	16	0	18	0%	77
rl5934	5934	700	**751131**	751054	751569	7	0%	15	0	14	0%	88
rl5934	5934	800	**675958**	675884	676248	7	0%	20	175	21	0%	58
rl5934	5934	900	**612629**	612574	612879	7	0%	17	35	14	0%	33
rl5934	5934	1000	**558167**	558088	558311	7	0%	28	603	45	0%	731
rl5934	5934	1100	**511192**	511138	511453	7	0%	22	43	15	0%	20
rl5934	5934	1200	**469747**	469712	469943	8	0%	18	0	11	0%	11
rl5934	5934	1300	**433060**	433015	433300	7	0%	19	5	12	0%	27
rl5934	5934	1400	**401370**	401356	401597	7	0%	15	0	9	0%	6
rl5934	5934	1500	**373566**	373566	373566	7	0%	17	0	7	0%	2
					Average total time					467		2022

Table 3: Results on medium TSP instances for our method and `Zebra` on our computer. TL=36000 seconds. ⧫ means that the computer ran out of memory. The average total time is calculated with the instances in which both methods solve the instances to optimality.

INSTANCE				PHASE 1			PHASE 1 + 2				Zebra	
name	$N = M$	p	$OPT/$ BKN	LB^1	UB^1	T^1	*gap*	*iter*	*nodes*	T^{tot}	*gap*	T
usa13509	13509	10	**398561730**	398561600	398561730	288	0%	10	0	755	∞	⧫
usa13509	13509	25	**234600221**	234600221	234600221	455	0%	10	0	455	∞	⧫
usa13509	13509	50	**157819849**	157815657	157819849	431	0%	11	45	646	∞	⧫
usa13509	13509	100	**108002205**	107983102	108002411	523	0%	23	605	4043	∞	⧫
usa13509	13509	200	**74220726**	74213328	74229411	426	0%	25	969	2269	∞	⧫
usa13509	13509	300	**59340915**	59334913	59346783	473	0%	23	984	1744	0%	18760
usa13509	13509	400	**50538905**	50533013	50575463	319	0%	24	874	2556	0%	23677
usa13509	13509	500	**44469860**	44463038	44499566	278	0%	36	2883	3945	0%	25105
usa13509	13509	600	**39952138**	39944049	39991088	295	0%	34	49175	23712	0%	TL
usa13509	13509	700	**36469603**	36463603	36512930	202	0%	24	6060	2551	0%	TL
usa13509	13509	800	**33635127**	33631192	33672848	210	0%	21	796	1215	0%	6007
usa13509	13509	900	**31275114**	31269089	31299760	182	0%	31	27272	11851	0%	TL
usa13509	13509	1000	**29268216**	29262339	29309009	154	0%	31	942	1382	0%	TL
usa13509	13509	2000	**18230856**	18229432	18238229	47	0%	21	202	125	0%	584
usa13509	13509	3000	**13098935**	13097929	13101469	49	0%	28	9	72	0%	1674
usa13509	13509	4000	**9905715**	9905071	9910848	37	0%	17	0	50	0%	166
usa13509	13509	5000	**7608605**	7608242	7611958	45	0%	22	0	61	0%	86
sw24978	24978	10	**22670073**	22670073	22670073	1037	0%	12	0	1037	∞	⧫
sw24978	24978	25	**14085626**	14085352	14085626	6651	0%	11	15	9447	∞	⧫
sw24978	24978	50	**9652817**	9652817	9652817	4136	0%	10	0	4136	∞	⧫
sw24978	24978	75	7766486	7765106	7766486	6310	0,010%	12	457	TL	∞	⧫
sw24978	24978	100	6660424	6657806	6660424	9634	0,031%	14	228	TL	∞	⧫
sw24978	24978	250	4034554	4034055	4036558	2413	0,003%	14	699	TL	∞	⧫
sw24978	24978	500	2747215	2746498	2751695	1866	0,015%	20	2621	TL	∞	⧫
sw24978	24978	1000	**1841723**	1841613	1844801	621	0%	23	1061	3814	0%	30796
sw24978	24978	2000	**1197278**	1197231	1198464	208	0%	17	132	476	0%	TL
sw24978	24978	3000	**911361**	911308	911988	145	0%	17	16	344	0%	3614
sw24978	24978	4000	**737645**	737602	738045	92	0%	15	0	190	0%	TL
sw24978	24978	5000	**617637**	617593	618096	76	0%	18	0	127	0%	TL
sw24978	24978	6000	**527336**	527307	527716	72	0%	17	0	112	0%	5188
sw24978	24978	7000	**455716**	455696	456074	63	0%	16	0	99	0%	3973
sw24978	24978	8000	**397217**	397153	397540	44	0%	20	0	97	0%	TL
sw24978	24978	9000	**347376**	347322	347621	44	0%	20	13	92	0%	TL
sw24978	24978	10000	**305998**	305932	306094	33	0%	17	0	60	0%	TL
					Average total time					1178		9969

Table 4: Results on large TSP instances for our method and `Zebra` on our computer. TL=36000 seconds. ⧫ means that the computer ran out of memory. The average total time is calculated with the instances in which both methods solve the instances to optimality

INSTANCE				PHASE 1			PHASE 1 + 2				Zebra	
name	$N = M$	*p*	*OPT/ BKN*	LB^1	UB^1	T^1	*gap*	*iter*	*nodes*	T^{tot}	*gap*	*T*
ch71009	71009	10000	4274662	4273680	4424131	6326	0,006%	36	18585	TL	∞	⧫
ch71009	71009	20000	**2377760**	2377409	2419539	681	0%	40	474	3581	∞	⧫
ch71009	71009	30000	**1464151**	1464015	1473517	431	0%	27	0	819	∞	⧫
ch71009	71009	40000	**879336**	879272	881997	220	0%	17	0	465	∞	⧫
ch71009	71009	50000	**463553**	463544	463904	133	0%	24	0	258	0%	653
ch71009	71009	60000	**167565**	167558	167789	49	0%	31	0	135	0%	331
pla85900	85900	10000	166853134	166627292	182428500	2841	0,12%	30	2113	TL	∞	⧫
pla85900	85900	20000	109007210	107246411	120645337	3975	1,58%	27	618	TL	∞	⧫
pla85900	85900	30000	86944862	86944715	87547287	1411	0,0002%	84	28033	TL	∞	⧫
pla85900	85900	40000	**69944715**	69944715	69965668	1006	0%	12	0	1006	∞	⧫
pla85900	85900	50000	**52944715**	52944715	52945623	921	0%	12	0	921	∞	⧫
pla85900	85900	60000	**35944715**	35944715	35945105	858	0%	11	0	858	∞	⧫
pla85900	85900	70000	**18977475**	18977475	18977475	73	0%	13	0	73	0%	122
pla85900	85900	80000	**4512752**	4512752	4512752	12	0%	20	0	13	0%	97
usa115475	115474	20000	5287343	5286659	5383798	3366	0,001%	36	11102	TL	∞	⧫
usa115475	115474	30000	**3815620**	3815143	3861590	1494	0%	41	589	11581	∞	⧫
usa115475	115474	40000	**2876909**	2876603	2904492	1353	0%	32	459	4431	∞	⧫
usa115475	115474	50000	**2189144**	2188903	2200969	1122	0%	28	480	3189	∞	⧫
usa115475	115474	60000	**1651400**	1651234	1657118	795	0%	25	0	1588	∞	⧫
usa115475	115474	70000	**1214299**	1214177	1217251	612	0%	17	0	1045	∞	⧫
usa115475	115474	80000	**851481**	851422	852851	435	0%	24	0	788	∞	⧫
usa115475	115474	90000	**548097**	548076	548560	270	0%	18	0	544	∞	⧫
ara238025	238025	10000	1354335	1345698	1446100	5197	0,64%	19	0	TL	∞	⧫
ara238025	238025	20000	857553	857453	878372	5582	0,004%	42	696	TL	∞	⧫
ara238025	238025	30000	**630969**	630872	643171	5123	0%	33	663	33687	∞	⧫
ara238025	238025	40000	**494842**	494804	498378	4135	0%	18	0	10028	∞	⧫
ara238025	238025	50000	**401835**	401795	404218	2675	0%	19	0	8327	∞	⧫
ara238025	238025	60000	**334279**	334236	335807	2969	0%	17	0	7240	∞	⧫
ara238025	238025	70000	**283627**	283592	286065	3058	0%	28	509	19298	∞	⧫
ara238025	238025	80000	**244233**	243936	248742	2578	0%	36	378	14615	∞	⧫
ara238025	238025	90000	**214233**	213936	219673	1548	0%	27	507	28391	∞	⧫
ara238025	238025	100000	**184233**	184069	188556	1973	0%	44	613	18473	∞	⧫
ara238025	238025	150000	**88025**	88025	88334	1532	0%	27	0	6057	∞	⧫
ara238025	238025	200000	**38025**	38025	38025	319	0%	11	0	319	∞	⧫
						Average total time				120		300

Table 5: Results on huge TSP instances for our method and `Zebra` on our computer. TL=36000 seconds. ⧫ means that the computer ran out of memory. The average total time is calculated with the instances in which both methods solve the instances to optimality

BIRCH instances

The results on BIRCH instances are summarized in Tables 6 and 7. Almost all of these instances were solved in our first phase either by finding an integer solution directly or by our rounding heuristic. Consequently, we obtain the optimal values of all of these instances quickly. Even when multiplying the solution time of `AvellaHeu` and `IrawanHeu` by their benchmark ratio (2.5 and 3, respectively), our approach remains the best for most instances.

INSTANCE				PHASE 1			PHASE 1 + 2				AvellaHeu	
name	$N = M$	p	OPT/BKN	LB^1	UB^1	T^1	*gap*	*iter*	*nodes*	T^{tot}	UB^h	T
ds1x1	10000	100	**12428,5**	12428,5	12428,5	9	0%	6	0	9	**12428,5**	47
ds1x2	15000	100	**18639,3**	18639,3	18639,3	29	0%	7	0	29	**18639,3**	101
ds1x3	20000	100	**24840,3**	24840,3	24840,3	38	0%	7	0	38	**24840,3**	210
ds1x4	9600	64	**11934,8**	11934,8	11934,8	13	0%	6	0	13	**11934,8**	56
ds1x5	12800	64	**15863,8**	15863,8	15863,8	25	0%	7	0	25	**15863,8**	84
ds1x6	16000	64	**20004,5**	20004,5	20004,5	31	0%	6	0	31	**20004,6**	129
ds1x7	19200	64	**24018,3**	24018,3	24018,3	55	0%	6	0	55	**24018,3**	219
ds1x8	10000	25	**12455,7**	12455,7	12455,7	28	0%	6	0	28	**12455,7**	82
ds1x9	12500	25	**15597,1**	15597,1	15597,1	43	0%	6	0	43	**15597,1**	115
ds1x0	15000	25	**18949,3**	18949,3	18949,3	67	0%	7	0	67	**18949,3**	175
ds1xA	17500	25	**21937,4**	21937,4	21937,4	116	0%	6	0	116	**21937,4**	241
ds1xB	20000	25	**25096,8**	25096,8	25096,8	108	0%	6	0	108	**25096,8**	365
ds3x1	10000	100	**9624,8**	9624,8	9624,8	16	0%	9	0	16	**9624,8**	60
ds3x2	15000	100	**15898,2**	15895,9	15899,1	39	0%	10	88	142	15904,1	121
ds3x3	20000	100	**19976,2**	19974,6	19977,6	97	0%	10	13	260	19989,0	222
ds3x4	9600	64	**8225,6**	8224,1	8225,7	24	0%	12	5	65	**8225,6**	57
ds3x5	12800	64	**10210,4**	10210,4	10210,4	36	0%	10	0	36	**10210,4**	98
ds3x6	16000	64	**13335,4**	13335,4	13335,4	62	0%	10	0	62	13340,5	170
ds3x7	19200	64	**15207,6**	15207,1	15207,6	176	0%	18	0	400	**15207,6**	229
ds3x8	10000	25	**7203,4**	7203,4	7203,4	61	0%	11	0	61	**7203,4**	94
ds3x9	12500	25	**8576,1**	8576,1	8576,1	68	0%	7	0	68	**8576,1**	144
ds3x0	15000	25	**9513,6**	9513,6	9513,6	135	0%	13	0	136	**9513,6**	192
ds3xA	17500	25	**12535,7**	12535,7	12535,7	224	0%	16	0	224	**12535,7**	250
ds3xB	20000	25	**13022,2**	13022,2	13022,2	244	0%	11	0	244	13052,8	364

Table 6: Results on BIRCH instances for our method and the results of `AvellaHeu` reported in Avella et al. (2012).

INSTANCE				PHASE 1			PHASE 1 + 2				IrawanHeu	
name	$N = M$	p	OPT/BKN	LB^1	UB^1	T^1	*gap*	*iter*	*nodes*	T^{tot}	UB^h	T
ds1n01	25000	25	**31229,4**	31229,4	31229,4	153	0%	7	0	153	31282,6	447
ds1n02	36000	36	**45115,6**	45115,6	45115,6	311	0%	7	0	311	45115,6	780
ds1n03	49000	49	**61384,1**	61384,1	61384,1	388	0%	7	0	388	61569,7	1216
ds1n04	64000	64	**80053,9**	80053,9	80053,9	675	0%	7	0	675	80377,4	2258
ds1n05	30000	25	**37563,6**	37563,6	37563,6	305	0%	7	0	305	37617,1	559
ds1n06	43200	36	**54191,4**	54191,4	54191,4	320	0%	7	0	320	54305,8	1003
ds1n07	58800	49	**73626,8**	73626,8	73626,8	683	0%	7	0	683	73854,7	1691
ds1n08	76800	64	**96039,4**	96039,4	96039,4	949	0%	7	0	949	96393,4	2834
ds1n09	35000	25	**43902,1**	43902,1	43902,1	385	0%	7	0	386	42972,1	758
ds1n10	50400	36	**63169,2**	63169,2	63169,2	533	0%	7	0	533	63329,2	1472
ds1n11	68600	49	**85833,5**	85833,5	85833,5	910	0%	8	0	910	86082,0	2441
ds1n12	89600	64	**112059,2**	112059,2	112059,2	1332	0%	7	0	1332	112485,2	4501
ds3n01	25000	25	**17696,2**	17696,2	17696,2	112	0%	6	0	112	17718,6	527
ds3n02	36000	36	**27423,0**	27423,0	27423,0	237	0%	7	0	237	27476,1	913
ds3n03	49000	49	**44149,0**	44149,0	44149,0	294	0%	10	0	295	44282,5	1760
ds3n04	64000	64	**58832,6**	58832,6	58832,6	807	0%	11	0	807	58991,5	2624
ds3n05	30000	25	**21829,9**	21829,9	21829,9	258	0%	7	0	258	21865,1	832
ds3n06	43200	36	**32339,4**	32339,4	32339,4	337	0%	9	0	337	32391,6	1873
ds3n07	58800	49	**50857,9**	50857,9	50857,9	831	0%	12	0	831	50857,9	2692
ds3n08	76800	64	**66561,4**	66561,0	66655,7	1490	0%	17	9	4587	66944,7	4393
ds3n09	35000	25	**24810,9**	24810,9	24810,9	869	0%	10	0	869	24833,7	972
ds3n10	50400	36	**38102,6**	38102,6	38102,6	504	0%	8	0	504	38162,3	2297
ds3n11	68600	49	**61850,6**	61850,6	61850,6	1065	0%	14	0	1065	62007,4	2556
ds3n12	89600	64	**78777,0**	78777,0	78777,0	1548	0%	19	0	1548	79245,3	5779

Table 7: Results on large BIRCH instances for our method and the results of `IrawanHeu` reported in Irawan & Salhi (2015b)

RW instances

The results on RW instances are summarized in Table 8. Even small RW instances can be very difficult to solve as previously observed by Elloumi & Plateau (2010). We think that it is mainly due to the fact that the instances are non-Euclidean. Furthermore, the total number of distances K is closer to $N \times M$, leading to more variables and constraints in Formulations $(F2)$ and $(F3)$. For large values of p, our decomposition can quickly solve the instances to optimality. These instances were not considered by either Avella et al. (2007) or García et al. (2011). Moreover, the code of `Zebra` cannot handle non-symmetric instances. Consequently, we only report the computation time and value UB^h of the solution computed by the heuristic `PopStar`.

INSTANCE				PHASE 1			PHASE 1 + 2				PopStar	
name	$N = M$	p	OPT	LB^1	UB^1	T^1	*gap*	*iter*	*nodes*	T^{tot}	UB^h	T
rw100_10	100	10	**530**	475	530	0,02	0%%	46	4817	6,82	**530**	0,05
rw100_20	100	20	**277**	274	277	0,01	0%	9	0	0,20	**277**	0,03
rw100_30	100	30	**213**	213	213	0,01	0%	9	0	0,01	**213**	0,02
rw100_40	100	40	**187**	187	187	0,01	0%	7	0	0,01	**187**	0,03
rw100_50	100	50	**172**	172	172	0,01	0%	7	0	0,01	**172**	0,02
rw250_10	250	10	**3691**	2811	3698	0,26	0%	71	4072666	31099	3698	0,31
rw250_25	250	25	**1360**	1216	1364	0,14	0,4%	64	5003894	TL	1364	0,34
rw250_50	250	50	**713**	699	713	0,07	0%	23	2050	7,31	**713**	0,15
rw250_75	250	75	**523**	523	523	0,02	0%	11	0	0,02	524	0,09
rw250_100	250	100	**444**	444	444	0,02	0%	9	0	0,02	**444**	0,08
rw250_125	250	125	**411**	411	411	0,02	0%	8	0	0,02	**411**	0,07
rw500_10	500	10	16108	11012	16144	1,35	21,5%	81	211046	TL	16144	2,53
rw500_25	500	25	5683	4403	5716	0,85	16,3%	73	506777	TL	5716	1,80
rw500_50	500	50	2627	2321	2627	0,52	6,5%	44	1290135	TL	2627	1,03
rw500_75	500	75	1757	1672	1757	0,36	1,1%	31	2436992	TL	1757	0,84
rw500_100	500	100	**1379**	1353	1382	0,25	0%	45	209745	1482	1382	0,47
rw500_150	500	150	**1024**	1024	1024	0,05	0%	8	0	0,05	**1024**	0,30
rw500_250	500	250	**833**	833	833	0,03	0%	9	0	0,03	**833**	0,25
rw1000_10	1000	10	68136	44697	68136	10,35	36,2%	61	19577	TL	68136	8,65
rw1000_25	1000	25	24964	17387	25042	6,22	34,1%	77	32697	TL	25042	7,60
rw1000_50	1000	50	11328	8760	11328	5,18	23,2%	65	94037	TL	11328	7,09
rw1000_75	1000	75	7207	5998	7223	4,20	16,2%	70	151511	TL	7223	3,15
rw1000_100	1000	100	5233	4631	5233	3,22	9,7%	42	285579	TL	5233	4,45
rw1000_200	1000	200	2710	2664	2710	0,94	0,5%	31	1310025	TL	2710	3,27
rw1000_300	1000	300	**2018**	2017	2018	0,20	0%	12	0	0,28	**2018**	1,99
rw1000_400	1000	400	**1734**	1734	1734	0,08	0%	9	0	0,09	**1734**	1,68
rw1000_500	1000	500	**1614**	1614	1614	0,07	0%	8	0	0,08	**1614**	1,35

Table 8: Results on RW instances for our exact method and `PopStar` heuristic in our computer. TL=36000 seconds.

ODM instances

The results on ODM instances are summarized in Table 9. To solve these instances, we need to add N constraints to ensure that each client is allocated to one of its non-forbidden neighbors. This generates a more complex master problem to solve. The rounding heuristic could not be used directly with these instances, so in order to save computation time, it was not used.

Obtaining a solution for these instances is hard since `PopStar` does not support their format and since random solutions may not be feasible due to the sparsity of the graphs. Consequently, to obtain an initial solution, we solve its corresponding formulation ($F3$) by CPLEX and stop it once it has found 3 feasible solutions. This approach empirically proved to be a good compromise between computation time and optimality gap. We were also unable to use `Zebra` for these instances. We solve to optimality all these instances. Our results remains the best on all instances even after multiplying `AvellaB&C`solution times by its benchmark ratio of value 6.

INSTANCE				PHASE 1			PHASE 1 + 2				AvellaB&C	
name	$N = M$	p	OPT	LB^1	UB^1	T^1	*gap*	*iter*	*nodes*	T^{tot}	*gap*	T
BD3773	3773	5	**726954998,4**	715785543,5	748669824,0	6	0%	123	59	58	0%	1540
BD3773	3773	6	**685812258,0**	673317265,9	720632505,6	10	0%	318	221	158	0%	41551
BD3773	3773	7	**651930471,0**	636565701,5	727428978,0	11	0%	1165	627	589	0%	216851
BD3773	3773	8	**620886605,4**	606599816,1	1157543276,4	20	0%	2520	1153	1434	1,7%	329053
BD3773	3773	9	**595955799,0**	581022150,3	649313415,0	18	0%	4028	1981	2385	2,6%	TL2
BD3773	3773	10	**574634206,8**	559096162,3	633383307,0	24	0%	8229	4848	4372	2,8%	TL2
BD3773	3773	11	**554972029,2**	539810124,3	603741870,0	29	0%	10139	5940	5832	2,9%	TL2
BD3773	3773	12	**536700087,0**	522614063,7	605415767,4	30	0%	13951	5898	8083	3,1%	TL2
BD3773	3773	13	**521375065,2**	507136693,1	581851884,6	31	0%	22500	9471	12581	3,2%	TL2
BD3773	3773	14	**507756740,4**	493051932,7	550267457,4	35	0%	52705	30275	31651	3,1%	TL2

Table 9: Results on ODM instances for our method and the results of `AvellaB&C` reported in Avella et al. (2007). TL2=360000 seconds.

4.4. Adaptation for the Uncapacitaded Facility Location problem

Given the closeness of the (pMP) and the (UFL) we want to compare our two-phase decomposition algorithm with the approach proposed in Fischetti et al. (2017) for the (UFL). They solve the problem in a *branch-and-cut* approach in which they search for violated Benders cuts for both the integer solutions and the fractional solutions of the linear relaxations with a polynomial time algorithm. This approach is also known as *branch-and-Benders-cut*. They also consider some stabilization techniques and heuristics for the cut loop at the root and at the branching nodes. We denote their method as `BBC`.

We have considered the same set of KG instances from UFLLIB[3] to compare the performance on the linear formulation. These instances can be divided into three groups, with $N = M \in \{250, 500, 750\}$. Within each KG group, there are two classes of instances, symmetric and asymmetric ones, denoted by "gs" and "ga", respectively. Additionally, each class contains three sub-classes, "a", "b," and "c," representing different cost settings: in subclass a, allocation costs are an order of magnitude higher than the facility opening costs; in subclass b, these costs are of the same order; and in subclass c, facility opening costs are an order of magnitude higher than the allocation costs.

The computational study in Fischetti et al. (2017) was conducted on a cluster of identical machines each consisting of an Intel Xeon E3-1220V2 CPU running at 3.10 GHz, with 16 GB of RAM each. They reported the wall-clock times and referred to four-thread runs with a time limit of 2 hours per instance (indicated by TL3 in the corresponding table).

KG Instances

The results on KG instances are summarized in Appendix A. To solve these instances we need to modify our master problem to consider the open cost of the sites and remove the constraint of limiting the number of open sites to p. This generates a harder master problem to solve. Let c_j be the cost of to open the site j.

$$
(MP_{UFL}) \begin{cases} \min & \displaystyle\sum_{i=1}^{N} \theta_i + \sum_{j=1}^{M} c_j y_j & \\ \text{s.t.} & \displaystyle\sum_{j=1}^{M} y_j = p & \\ & \theta_i \text{ satisfies } BD_i & i \in [N] \\ & y_j \in \{0,1\} & j \in [M] \end{cases}
$$

We must also modify the heuristics used in Phase 1 since PopStar does not take into account the opening costs of the sites and we do not have the parameter p. We consider a greedy heuristic to get the initial solution. As the rounding heuristic, we set to 1 all the sites which $\overline{y}_j > 0.4$ for $j \in [M]$.

Our results are consistent with those presented in Fischetti et al. (2017) since within the 2-hour limit we also are not able to optimally solve the considered instances. Fischetti et al. (2017) do not report lower bounds on the optimal value. We observe here that the final gap is relatively small

[3]https://resources.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/index.html

since its maximal value is 1.7%. However, Fischetti et al. (2017) have a better upper bound for all the instances which is probably due to the stabilization technique and their primal heuristics used at each node.

5. Conclusions

The p-median problem is a well-studied discrete location problem in which we have to choose p sites among M to allocate N clients in order to minimize the sum of their allocation distances. This problem has various applications and several heuristic methods have been proposed to solve it. However, its exact solution remains a challenge for large-scale instances. The previously most effective approach in the literature was able to solve instances up to 85900 clients and sites.

We performed a Benders decomposition of the most efficient formulation of the p-median problem. The efficiency of our decomposition comes from a fast algorithm for the solution of the sub-problems in conjunction with improvements in a two-phase solution implementation. In the first phase, the integrity constraints are relaxed and in the second phase, the problem is solved in an efficient *branch-and-cut* approach.

Our approach outperforms other state-of-the-art methods. We solve for the first time to optimality instances having up to 89600 and 238025 clients and sites from the BIRCH and TSP libraries, respectively. We tested our decomposition algorithm on other p-median instances: RW instances which do not satisfy triangle inequality and ODM instances in which there are allocations that are not allowed between certain clients and sites. For the RW instances. we were able to solve instances of up to 1000 clients with a large value of p. For ODM instances with 3773 clients, we solve previously unsolved instances within 10 hours. We also adapt our approach to test it on the difficult KG instances of the (UFL) problem obtaining relatively small optimality gaps.

One of the perspectives of this research is to exploit these results on other families of location problems. It is also expected to use other branching strategies that allow a greater efficiency during the development of the *branch-and-cut* algorithm.

Acknowledgments

The authors would like to thank Sergio Garcia for providing the code used in García et al. (2011). This work was funded by the National Agency for Research and Development of Chile - ANID (Scholarship Phd. Program 2019-72200492).

Appendix A. Detailed results for the UFL instances

INSTANCE			PHASE 1				PHASE 1 + 2				BBC	
name	$N = M$	*BKN*	LB^1	UB^1	T^1	UB^2	*gap*	*iter*	*nodes*	T^{tot}	UB^h	T
ga500a-1	500	511383	510589,0	514040	2,0	511474	0,11%	65	227655	TL3	511383	TL3
ga500a-2	500	511255	510472,2	514397	1,8	511367	0,11%	69	217453	TL3	511255	TL3
ga500a-3	500	510810	510139,0	513356	1,9	510965	0,09%	78	264644	TL3	510810	TL3
ga500a-4	500	511008	510382,6	512694	1,7	511082	0,08%	35	369649	TL3	511008	TL3
ga500a-5	500	511239	510487,7	513475	2,0	511425	0,11%	74	237491	TL3	511239	TL3
ga500b-1	500	538060	533338,6	545628	1,6	538452	0,49%	66	119621	TL3	538060	TL3
ga500b-2	500	537850	533087,6	619389	1,6	538457	0,60%	84	87311	TL3	537850	TL3
ga500b-3	500	537924	532735,7	626127	1,7	538264	0,63%	77	76708	TL3	537924	TL3
ga500b-4	500	537925	532841,9	670237	1,5	538385	0,58%	90	86797	TL3	537925	TL3
ga500b-5	500	537482	532968,8	597360	1,5	537662	0,44%	71	124594	TL3	537482	TL3
gs500a-1	500	511188	510409,9	513472	3,4	511314	0,10%	95	233059	TL3	511188	TL3
gs500a-2	500	511179	510448,7	514006	1,9	511354	0,11%	82	165089	TL3	511179	TL3
gs500a-3	500	511112	510321,4	513779	2,3	511287	0,12%	85	141758	TL3	511112	TL3
gs500a-4	500	511137	510369,6	513787	2,2	511278	0,11%	66	159787	TL3	511137	TL3
gs500a-5	500	511293	510494,5	513990	2,0	511532	0,13%	83	145791	TL3	511293	TL3
gs500b-1	500	537931	533026,7	659721	1,4	538418	0,57%	76	84363	TL3	537931	TL3
gs500b-2	500	537763	533096,1	662841	1,4	538160	0,51%	67	113638	TL3	537763	TL3
gs500b-3	500	537854	532832,2	678990	1,7	538457	0,63%	91	76097	TL3	537854	TL3
gs500b-4	500	537742	532717,2	664753	1,5	538422	0,66%	97	74407	TL3	537742	TL3
gs500b-5	500	538270	533098,2	669016	1,6	538618	0,59%	80	83721	TL3	538270	TL3
ga750a-1	750	763528	762464,5	766540	5,8	763869	0,15%	93	50507	TL3	763528	TL3
ga750a-2	750	762653	762520,2	767067	9,2	763973	0,15%	89	46602	TL3	762653	TL3
ga750a-3	750	763697	762568,5	766608	6,2	763930	0,14%	79	43302	TL3	763697	TL3
ga750a-4	750	763945	762738,5	767839	6,6	764240	0,16%	77	36036	TL3	763945	TL3
ga750a-5	750	763786	762637,0	767096	7,3	764159	0,16%	75	40176	TL3	763786	TL3
ga750b-1	750	796454	790121,9	897053	4,6	797090	0,62%	66	26275	TL3	796454	TL3
ga750b-2	750	795963	789512,4	938670	4,5	796498	0,62%	82	25485	TL3	795963	TL3
ga750b-3	750	796130	789618,5	929140	4,6	796640	0,63%	79	23499	TL3	796359	TL3
ga750b-4	750	797013	790345,1	926574	4,7	797935	0,69%	90	22494	TL3	797013	TL3
ga750b-5	750	796387	789647,3	930851	4,0	796934	0,66%	89	23036	TL3	796549	TL3
ga750c-1	750	902026	875624,7	1018182	2,2	903292	1,59%	87	23967	TL3	902026	TL3
ga750c-2	750	899651	873946,7	1017899	2,8	902368	1,70%	84	22481	TL3	899651	TL3
ga750c-3	750	900010	874108,9	1012861	3,1	902099	1,66%	82	24265	TL3	900019	TL3
ga750c-4	750	900044	875565,9	1028964	2,4	901809	1,42%	76	23248	TL3	900044	TL3
ga750c-5	750	899235	873191,5	1028543	2,6	900541	1,55%	87	24812	TL3	899235	TL3
gs750a-1	750	763671	762564,9	767232	6,5	763925	0,14%	84	43430	TL3	763671	TL3
gs750a-2	750	763548	762529,4	766414	6,1	763666	0,11%	90	57546	TL3	763548	TL3
gs750a-3	750	763727	762568,1	765552	6,6	764031	0,16%	82	41309	TL3	763727	TL3
gs750a-4	750	763887	762788,3	766768	7,5	764208	0,15%	81	36651	TL3	763922	TL3
gs750a-5	750	763614	762528,9	766741	6,4	763947	0,15%	92	44013	TL3	763614	TL3
gs750b-1	750	797026	790349,4	994689	4,7	797713	0,68%	82	20525	TL3	797329	TL3
gs750b-2	750	796170	789669,9	895766	4,1	796843	0,66%	80	26652	TL3	796170	TL3
gs750b-3	750	796589	789935,6	996019	4,5	797357	0,69%	84	22153	TL3	796589	TL3
gs750b-4	750	796734	790080,8	997279	4,2	797176	0,65%	69	26574	TL3	797020	TL3
gs750b-5	750	796365	789902,8	930041	4,6	797026	0,63%	63	23510	TL3	796365	TL3
gs750c-1	750	900363	875363,9	1020684	2,9	903801	1,64%	80	22196	TL3	900363	TL3
gs750c-2	750	897886	874186,6	1004896	2,5	900668	1,49%	70	27230	TL3	897886	TL3
gs750c-3	750	901656	874762,5	1019498	2,5	902767	1,61%	89	21335	TL3	901656	TL3
gs750c-4	750	901239	875410,6	1007402	2,6	902591	1,61%	81	24195	TL3	901239	TL3
gs750c-5	750	900216	875956,4	1017315	3,4	903719	1,65%	78	20022	TL3	900216	TL3

Table A.10: Results on KG instances for our method and the results of `BBC` reported in Fischetti et al. (2017) TL3=72000 seconds.

References

An, Y., Zeng, B., Zhang, Y., & Zhao, L. (2014). Reliable p-median facility location problem: two-stage robust models and algorithms. *Transportation Research Part B: Methodological*, *64*, 54–72. doi:https://doi.org/10.1016/j.trb.2014.02.005.

Avella, P., Boccia, M., Salerno, S., & Vasilyev, I. (2012). An aggregation heuristic for large scale p-median problem. *Computers & Operations Research*, *39*, 1625–1632. doi:https://doi.org/10.1016/j.cor.2011.09.016.

Avella, P., Sassano, A., & Vasil'ev, I. (2007). Computational study of large-scale p-median problems. *Mathematical Programming*, *109*, 89–114. doi:10.1007/s10107-005-0700-6.

Basu, S., Sharma, M., & Ghosh, P. S. (2015). Metaheuristic applications on discrete facility location problems: a survey. *OPSEARCH*, *52*, 530–561. doi:10.1007/s12597-014-0190-5.

Beasley, J. E. (1990). Or-library: Distributing test problems by electronic mail. *The Journal of the Operational Research Society*, *41*, 1069–1072. URL: http://www.jstor.org/stable/2582903.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*, 238–252. doi:10.1007/BF01386316.

Briant, O., & Naddef, D. (2004). The optimal diversity management problem. *Operations research*, *52*, 515–526. doi:10.1287/opre.1040.0108.

Cordeau, J.-F., Furini, F., & Ljubić, I. (2019). Benders decomposition for very large scale partial set covering and maximal covering location problems. *European Journal of Operational Research*, *275*, 882 – 896. doi:10.1016/j.ejor.2018.12.021.

Cornuejols, G., Nemhauser, G. L., & Wolsey, L. A. (1980). A canonical representation of simple plant location problems and its applications. *SIAM Journal on Algebraic Discrete Methods*, *1*, 261–272. doi:10.1137/0601030.

Elloumi, S. (2010). A tighter formulation of the p-median problem. *Journal of Combinatorial Optimization*, *19*, 69–83. doi:10.1007/s10878-008-9162-0.

Elloumi, S., & Plateau, A. (2010). A computational study for the p-median problem. *Electronic Notes in Discrete Mathematics*, *36*, 455–462. doi:10.1016/j.endm.2010.05.058.

Fischetti, M., Ljubic, I., & Sinnl, M. (2017). Redesigning benders decomposition for large-scale facility location. *Management Science*, *63*, 2146–2162. doi:10.1287/mnsc.2016.2461.

Gaar, E., & Sinnl, M. (2022). A scaleable projection-based branch-and-cut algorithm for the p-center problem. *European Journal of Operational Research*, . doi:https://doi.org/10.1016/j.ejor.2022.02.016.

Galvão, R. D. (1980). A dual-bounded algorithm for the p-median problem. *Operations Research*, *28*, 1112–1121. doi:10.1287/opre.28.5.1112.

García, S., Labbé, M., & Marín, A. (2011). Solving large p-median problems with a radius formulation. *INFORMS Journal on Computing*, *23*, 546–556. doi:10.1287/ijoc.1100.0418.

Hakimi, S. (1964). Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, *12*, 450–459. doi:10.1287/opre.12.3.450.

Hansen, P., Brimberg, J., Urošević, D., & Mladenović, N. (2009). Solving large p-median clustering problems by primal–dual variable neighborhood search. *Data Mining and Knowledge Discovery*, *19*, 351–375. doi:10.1007/s10618-009-0135-4.

Irawan, C., & Salhi, S. (2015a). Aggregation and non aggregation techniques for large facility location problems: A survey. *Yugoslav Journal of Operations Research*, *25*, 1–1. doi:10.2298/YJOR140909001I.

Irawan, C. A., & Salhi, S. (2015b). Solving large p-median problems by a multistage hybrid approach using demand points aggregation and variable neighbourhood search. *Journal of Global Optimization*, *63*, 537–554. doi:10.1007/s10898-013-0080-z.

Irawan, C. A., Salhi, S., & Scaparra, M. P. (2014). An adaptive multiphase approach for large unconditional and conditional p-median problems. *European Journal of Operational Research*, *237*, 590–605. doi:https://doi.org/10.1016/j.ejor.2014.01.050.

Kariv, O., & Hakimi, S. L. (1979). An algorithmic approach to network location problems. ii: The p-medians. *SIAM Journal on Applied Mathematics*, *37*, 539–560. URL: http://www.jstor.org/stable/2100911.

Klastorin, T. D. (1985). The p-Median Problem for Cluster Analysis: A Comparative Test Using the Mixture Model Approach. *Management Science*, *31*, 84–95. URL: https://doi.org/10.1287/mnsc.31.1.84. doi:10.1287/mnsc.31.1.84.

Laporte, G., Nickel, S., & Saldanha-da Gama, F. (2019). *Location Science (2nd ed)*. Springer International Publishing. doi:10.1007/978-3-030-32177-2.

Magnanti, T. L., & Wong, R. T. (1981). Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research*, *29*, 464–484. URL: http://www.jstor.org/stable/170108.

Marín, & Pelegrín, M. (2019). The p-median problem. In *Location Science* (pp. 25–50). Springer International Publishing. doi:10.1007/978-3-030-32177-2_2.

Mladenović, N., Brimberg, J., Hansen, P., & Moreno-Pérez, J. A. (2007). The p-median problem: A survey of metaheuristic approaches. *European Journal of Operational Research*, . doi:10.1016/j.ejor.2005.05.034.

Mu, W., & Tong, D. (2020). On solving large p-median problems. *Environment and Planning B: Urban Analytics and City Science*, *47*, 981–996. URL: 10.1177/2399808319892598. doi:10.1177/2399808319892598.

Park, H.-S., & Jun, C.-H. (2009). A simple and fast algorithm for K-medoids clustering. *Expert Systems with Applications*, *36*, 3336–3341. doi:10.1016/j.eswa.2008.01.039.

Rahmaniani, R., Crainic, T. G., Gendreau, M., & Rei, W. (2017). The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, *259*, 801 – 817. doi:10.1016/j.ejor.2016.12.005.

Reese, J. (2006). Solution methods for the p-median problem: An annotated bibliography. *Networks*, *48*, 125–142. doi:10.1002/net.20128.

Reinelt, G. (1991). TSPLIB—A Traveling Salesman Problem Library. *ORSA Journal on Computing*, *3*, 376–384. doi:10.1287/ijoc.3.4.376.

Resende, M. G. C., & Werneck, R. F. (2004). A Hybrid Heuristic for the p-Median Problem. *Journal of Heuristics*, *10*, 59–88. doi:10.1023/B:HEUR.0000019986.96257.50.

ReVelle, C. S., & Swain, R. W. (1970). Central facilities location. *Geographical Analysis*, *2*, 30–42. doi:10.1111/j.1538-4632.1970.tb00142.x.

Takedomi, S., Ishigaki, T., Hanatsuka, Y., & Mori, T. (2022). Facility location optimization with pMP modeling incorporating waiting time prediction function for emergency road services. *Computers & Industrial Engineering*, *164*, 107859. doi:10.1016/j.cie.2021.107859.

Ushakov, A. V., & Vasilyev, I. (2021). Near-optimal large-scale k-medoids clustering. *Information Sciences*, *545*, 344–362. doi:10.1016/j.ins.2020.08.121.

Voevodski, K. (2021). Large Scale K-Median Clustering for Stable Clustering Instances. In A. Banerjee, & K. Fukumizu (Eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics* (pp. 2890–2898). PMLR volume 130 of *Proceedings of Machine Learning Research*. URL: https://proceedings.mlr.press/v130/voevodski21a.html.