An efficient Benders decomposition for the \( p \)-median problem

Cristian Duran-Mateluna\(^1,2,3\), Zacharie Ales\(^1,2\), Sourour Elloumi\(^1,2\)

\(^1\) UMA, ENSTA Paris, Institut Polytechnique de Paris, Palaiseau, France.
\{cristian.duran, zacharie.ales, sourour.elloumi\}@ensta-paris.fr
\(^2\) CEDRIC, Conservatoire National des Arts et Métiers, Paris, France.
\(^3\) PDSPS, Industrial Engineering Department, University of Santiago of Chile, Santiago, Chile.

Keywords: discrete location, \( p \)-median problem, Benders decomposition.

1 Introduction

The \( p \)-median problem (\( pMP \)) is one of the fundamental problems in Location Science [18]. In the \( pMP \), \( p \) sites have to be chosen from the set of candidate sites without considering set-up costs. The allocation costs are usually equal to the distance or travel time between clients and sites. The \( pMP \) is an NP-hard problem [17] and leads to applications where the sites may correspond to warehouses, plants, or shelters for example. This problem also occurs in the contexts of emergency logistics and humanitarian relief. A well-known clustering problem called \( k \)-medoids problem is also a special case of the \( pMP \) in which 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.

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 resolution of large instances remains a challenge [19]. Some location problems have recently been efficiently solved using the Benders decomposition method (see e.g., [6, 10]). In this work, we explore a Benders decomposition of a recent formulation of the \( pMP \). We prove that the Benders cuts can be separated quickly. We implement a branch-and-Benders-cut approach that outperforms by an order of magnitude the state-of-the-art method of [11] on more than 200 benchmark instances with up to 238025 clients and sites. Our work is published in [8].

2 The \( p \)-median problem (\( pMP \))

The \( p \)-median is formally defined as follows. Given a set of \( N \) clients, a set of \( M \) potential centers to open, and their corresponding index sets \( \mathcal{N} = \{1, ..., N\} \) and \( \mathcal{M} = \{1, ..., M\} \), let \( d_{ij} \) be the distance between client \( i \in \mathcal{N} \) and site \( j \in \mathcal{M} \) and let \( p \in \mathbb{N} \) be the number of sites to open. The objective is to find a set \( S \) of \( p \) sites such that the sum of the distances between each client and its closest site in \( S \) is minimized. The \( pMP \) was introduced in [12] 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 many of variants and extensions. We refer to [19] for more details and references.

There are three main MILP formulations for this problem. The classical one (\( F1 \)) introduced in [24] that considers a binary variable \( y_j \) for each site \( j \in \mathcal{M} \) that takes value 1 if the site is open, and 0 otherwise; and a binary variable \( x_{ij} \) that takes value 1 if client \( i \in \mathcal{N} \) is allocated to site \( j \in \mathcal{M} \) and 0 otherwise. An alternative formulation (\( F2 \)) was proposed by [7] which for each client \( i \in \mathcal{N} \) orders all its distinct distances to the sites. More formally, let \( K_i \) be the number of different distances from \( i \) to any site. Let \( D^1_i < D^2_i < ... < D^K_i \) be these distances sorted, and let \( K_i \) be the corresponding index set \( \{1, ..., K_i\} \). Formulation (\( F2 \)) uses the same \( y \) variables as in (\( F1 \) and introduces new binary variables \( z \). For any client \( i \in \mathcal{N} \) and \( k \in K_i \),
\( z^k_i \) is equal to 0 if there is an open site at distance at most \( D^k_i \) from client \( i \), and 1 otherwise. As \( K_i \leq M \), it follows that (F2) has at most as many variables and constraints as (F1). Finally, [9] introduced formulation (F3) based on (F2). Given that, by definition, \( z^{k-1}_i \) is equal to 0 implies that \( z^k_i \) is also equal to 0, (F3) replaces the following Constraints (1) of (F2):

\[
 z^k_i + \sum_{j: d_{ij} \leq D^k_i} y_j \geq 1 \quad i \in \mathcal{N}, \ k \in K_i
\]

by Constraints (4) and (5). Thus, the coefficients matrix of the following (F3) is more sparse:

\[
 (F3) \quad \begin{align*}
 \min & \quad \sum_{i \in \mathcal{N}} \left( D^1_i + \sum_{k=1}^{K_i-1} (D^{k+1}_i - D^k_i) z^k_i \right) \\
 \text{s.t.} & \quad \sum_{j \in \mathcal{M}} y_j = p \\
 & \quad \sum_{j: d_{ij} = D^1_i} y_j \geq 1 \quad i \in \mathcal{N} \\
 & \quad z^k_i + \sum_{j: d_{ij} = D^k_i} y_j \geq z^{k-1}_i \quad i \in \mathcal{N}, \ k = 2, ..., K_i \\
 & \quad z^k_i \geq 0 \quad i \in \mathcal{N}, \ k \in K_i \\
 & \quad y_j \in \{0, 1\} \quad j \in \mathcal{M}
\end{align*}
\]

Since (F3) is the most efficient formulation, we have studied a Benders decomposition based on this formulation. The main heuristics to solve the \((pMP)\) are presented in the following surveys: [3, 14, 20, 21]. To the best of our knowledge, the state-of-the-art exact method is the Zebra algorithm [11], which considers a branch-and-cut-and-price algorithm based on formulation (F2). It relies on the fact that the \( z \) variables satisfy \( z^k_i \geq z^{k+1}_i \) in any optimal solution of (F2) or its LP relaxation. Therefore, it is enough to solve these problems on a reduced subset of variables \( z \), which is iteratively enlarged until an optimal solution is obtained. Zebra performs well on instances with up to 85900 nodes and large values of \( p \).

3 Benders decomposition for the \((pMP)\)

The Benders decomposition 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.

Based on (F3), our master problem contains the location variables \( y \) and our sub-problem contains the allocation variables \( z \). The master problem also contains a variable \( \theta_i \) for each client \( i \in \mathcal{N} \) which represents the allocation distances of the clients. Given a feasible solution of the master problem (a set of \( p \) sites), the sub-problem can be split into \( N \) sub-problems, each computing the allocation distance of one client and potentially leading to the addition of a cut. We show that Benders cuts can be quickly separated using the following definition.

**Definition 1** Given a solution \( \bar{y} \) of the master problem from (F3) or of its LP-relaxation, we define \( k_i \) for each \( i \in \mathcal{N} \) as follows:

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

Index \( \tilde{k}_i \) is equal to 0 if client \( i \) is covered at distance \( D^1_i \). Otherwise it is the largest distance index for which customer \( i \) is not yet covered in solution \( \bar{y} \). Therefore, if \( \bar{y} \) is binary, then the allocation distance of client \( i \) for this solution is \( D^{\tilde{k}_i+1}_i \). This definition allows us to state the following theorem.
Theorem 1 Given a solution $\bar{y}$ of the master problem from (F3) of its LP-relaxation and the corresponding indices $\tilde{k}_i$, the corresponding Benders cuts for each $i \in N$ can be written as follows:

$$
\begin{align*}
\theta_i & \geq D^{k_i+1}_i - \sum_{j : d_{ij} \leq D^{k_i}_i} (D^{k_i+1}_i - d_{ij})y_j & \text{otherwise} \\
\theta_i & \geq D^{k_i+1}_i & \text{if } \tilde{k}_i = 0
\end{align*}
$$

This theorem ensures that there is a finite number of Benders cuts. This enables us to define the following new compact formulation (F4) for the p-median problem.

$$
\begin{align*}
\min_{i \in N} & \sum_{i \in N} \theta_i \\
\text{s.t.} & \sum_{j \in M} y_j = p \\
& \theta_i \geq D^{1}_i & \text{if } i \in N \\
& \theta_i \geq D^{k+1}_i - \sum_{j : d_{ij} \leq D^{k}_i} (D^{k+1}_i - d_{ij})y_j & i \in N, k \in \{1, ..., K_i - 1\} \\
& y_j \in \{0, 1\} & j \in M
\end{align*}
$$

Constraints (8) correspond to the lower bound of each allocation distance given by the smallest distance value from each client to its nearest site. Constraints (9) ensure that each variable $\theta_i$ is larger than $D^{k+1}_i$ unless a site $j$ is opened at a distance not more than $D^{k}_i$ from client $i$. In this case the allocation distance of client $i$ is at most $d_{ij}$.

The efficiency of our decomposition comes from an $O(M)$ algorithm to separate the Benders cuts along with several implementation improvements in a two-phase resolution. In Phase 1, the integrity constraints are relaxed, which allows many cuts to be generated quickly. In order to enhance the performance of this phase, we provide a good candidate solution to the initial master problem which significantly reduces the number of iterations. To that end, we use the state-of-art PopStar heuristic [23]. Additionally, we use a rounding heuristic at each iteration to try to improve the upper bound of the problem. 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 number of constraints in the master problem. Phase 1 provides a lower and an upper bound of the problem, which can be used to perform an analysis of the reduced costs of the last fractional solution to reduce the number of variables. In Phase 2, we add the integrity constraints and the obtained master problem is solved through a branch-and-Benders-cut. At each node which provides an integer solution, we solve the sub-problems in order to generate Benders cuts. The resolution of the sub-problems is performed through callbacks which is a feature provided by mixed integer programming solvers.

4 Computational study

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 the branch-and-cut framework. The absolute tolerance to the best integer objective (EpGap) has been set to $10^{-10}$, and the absolute MIP gap (EpAGap) to 0.9999.

We study the same instances used for the state-of-the-art method Zebra that is the benchmark instances from OR-Library [4] and TSP-Library [22]. 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. We select a set from the TSP-Library having between 1304 and 238025 clients. Another set of symmetric instances that satisfy triangle inequality are the BIRCH instances, usually solved by heuristics algorithms (see
We consider two types of instances with sizes from 10000 to 20000 points for
the first one and from 25000 to 89600 points for the second one. We compare performances of
our algorithm to that of an aggregation heuristic \cite{AvellaHeu} for the first type,
and an hybrid heuristic combining aggregation and variable neighborhood search \cite{IrawanHeu}
for the second type. We also consider the RW instances originally proposed
by \cite{PopStar} with the PopStar heuristic. They correspond to completely random distance matrices.
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. We
also consider the ODM instances which were introduced by \cite{AvellaB&C} and are solved in \cite{AvellaB&C}
with a branch-and-cut-and-price algorithm (denoted by AvellaB&C). These instances correspond to
the optimal diversity management problem in which certain allocations between clients and
sites are not allowed. We consider the hardest instances in which $N = 3773$.

To summarize all our results, Table 4 presents for each dataset the number of instances
considered, the average cpu time ($Time$) computed on the instances solved to optimality by
both approaches, the percentage of optimally solved instances ($Opt$), and the average final
optimality gap ($Gap$). The grey values indicate that the solution times were not obtained
on the same computer, unlike Zebra and PopStar. We can see that our solution times are
significantly faster than the two other exact approaches and are competitive with the ones of
the heuristics. Moreover, the number of instances that we solve to optimality and our average
gaps are significantly better.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># instances</th>
<th>Algorithm</th>
<th>Time</th>
<th>Opt</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSP</td>
<td>149</td>
<td>Zebra</td>
<td>2408s</td>
<td>68%</td>
<td>0.28%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Our method</td>
<td>381s</td>
<td>91%</td>
<td>0.03%</td>
</tr>
<tr>
<td>BIRCH dsx</td>
<td>24</td>
<td>AvellaHeu*</td>
<td>150s</td>
<td>79%</td>
<td>0.01%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Our method</td>
<td>95s</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>BIRCH dsn</td>
<td>24</td>
<td>IrawanHeu*</td>
<td>1965s</td>
<td>8%</td>
<td>0.32%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Our method</td>
<td>766s</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>RW</td>
<td>27</td>
<td>PopStar*</td>
<td>6s</td>
<td>48%</td>
<td>9.47%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Our method</td>
<td>15s</td>
<td>59%</td>
<td>6.90%</td>
</tr>
<tr>
<td>ODM</td>
<td>10</td>
<td>AvellaB&amp;C</td>
<td>147248s</td>
<td>30%</td>
<td>1.94%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Our method</td>
<td>560s</td>
<td>100%</td>
<td>0%</td>
</tr>
</tbody>
</table>

TAB. 1: Summary of all our results. * means heuristic method

We highlight that several instances are solved for the first time having up to 89600 and 238025
clients and sites from the BIRCH and TSP libraries, respectively. For the RW instances, it
was possible for the first time to solve instances with up to 1000 clients with a large value
of $p$. For ODM instances with 3773 clients, previously unsolved instances were solved within
10 hours. To give more detailed results, Table 2 presents the results obtained on the largest
TSP instances. In which column OPT/BKN contains the 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 the best-known value was improved. We observe that we are significantly faster than
Zebra. Moreover, when both methods reach the time limit, we are able to prove significantly
smaller gaps.

5 Conclusions

We define a Benders decomposition based on the most efficient formulation of the $p$-median
problem. The efficiency of the proposed decomposition comes from a fast algorithm for the sub-
problems in conjunction with additional improvements implemented in a two-phase algorithm.
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 on five data sets from the literature. Our approach was able to improve the best-known solution of 91% of the instances which had not previously been solved to optimality. Finally we found an optimal solution for 81% of them. 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 could allow a greater efficiency during the development of the branch-and-cut algorithm.

<table>
<thead>
<tr>
<th>INSTANCE</th>
<th>PHASE 1</th>
<th>PHASE 1 + 2</th>
<th>Zebra</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>name N = M</td>
<td>p</td>
<td>OPT/ BKN</td>
</tr>
<tr>
<td>ch71009</td>
<td>71009</td>
<td>10000</td>
<td>4274602</td>
</tr>
<tr>
<td>20000</td>
<td>2377760</td>
<td>2377409</td>
<td>2419539</td>
</tr>
<tr>
<td>30000</td>
<td>1464151</td>
<td>1464015</td>
<td>1473517</td>
</tr>
<tr>
<td>40000</td>
<td>879336</td>
<td>879272</td>
<td>881997</td>
</tr>
<tr>
<td>50000</td>
<td>463553</td>
<td>463544</td>
<td>463904</td>
</tr>
<tr>
<td>60000</td>
<td>167565</td>
<td>167558</td>
<td>167789</td>
</tr>
</tbody>
</table>

| pla85900 | 85900 | 10000 | 166853134 | 166627292 | 182428500 | 2841 | 0.12% | 30 | 2113 | TL | ∞ | ♦ |
| 20000 | 109007210 | 107246411 | 120645337 | 3975 | 1.58% | 27 | 618 | 1307 | TL | ∞ | ♦ |
| 30000 | 86944862 | 86944715 | 87547287 | 1411 | 0.0002% | 84 | 28033 | TL | ∞ | ♦ |
| 40000 | 69944715 | 69944715 | 69965628 | 431 | 0% | 27 | 0 | 421 | 0% | 938 |
| 50000 | 52944715 | 52944715 | 52945623 | 921 | 0% | 40 | 589 | 1158 | 0% | 619 |
| 60000 | 35944715 | 35944715 | 35945105 | 858 | 0% | 40 | 589 | 1158 | 0% | 619 |

| usa115475 | 115475 | 10000 | 5286599 | 538798 | 3366 | 0.003% | 36 | 11102 | TL | ∞ | ♦ |
| 20000 | 3815620 | 3815433 | 3861590 | 1494 | 0% | 41 | 589 | 1158 | ∞ | ♦ |
| 30000 | 2876909 | 2876603 | 2904492 | 1353 | 0% | 32 | 459 | 1002 | ∞ | ♦ |
| 40000 | 2189144 | 2188903 | 2200969 | 1122 | 0% | 28 | 480 | 3189 | 0% | 978 |
| 50000 | 1651400 | 1651234 | 1657118 | 795 | 0% | 25 | 0 | 1588 | 0% | 605 |

| ara238025 | 238025 | 10000 | 1354335 | 1345698 | 1446100 | 5197 | 0.64% | 19 | 0 | TL | ∞ | ♦ |
| 20000 | 851275 | 851275 | 851275 | 12 | 0% | 20 | 0 | 13 | 0% | 97 |

TAB. 2: Results on the largest TSP instances for our method and Zebra on our computer. TL=36000 seconds. ♦ means that the computer ran out of memory.

References


