Les Cahiers du GERAD ISSN: 0711–2440

Accelerating Benders decomposition for the p-median problem through variable aggregation

R. S. H. Willemsen, D. Aloise, R. Jans

G–2025–27

April 2025

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2025
– Bibliothèque et Archives Canada, 2025

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2025
– Library and Archives Canada, 2025

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Accelerating Benders decomposition for the p-median problem through variable aggregation

Rick S. H. Willemsen [a]

Daniel Aloise [b, d]

Raf Jans [c, d]

[a] *Econometric Institute, Erasmus University, 3062 PA Rotterdam, The Netherlands*

[b] *Department of Computer Engineering, Polytechnique Montréal, Montréal, (Qc), Canada, H3T 1J4*

[c] *Department of Logistics and Operations Management, HEC Montréal, Montréal (Qc), Canada, H3T 2A7*

[d] *GERAD, Montréal (Qc), Canada, H3T 1J4*

daniel.aloise@gerad.ca
raf.jans@hec.ca

April 2025
Les Cahiers du GERAD
G–2025–27

Abstract : The p-median problem is a classical location problem where the goal is to select p facilities while minimizing the sum of distances from each location to its nearest facility. Recent advancements in solving the p-median and related problems have successfully leveraged Benders decomposition methods. The current bottleneck is the large number of variables and Benders cuts that are needed. We consider variable aggregation to reduce the size of these models. We propose to partially aggregate the variables in the model based on a start solution; aggregation occurs only when the corresponding locations are assigned to the same facility in the initial solution. In addition, we propose a set of valid inequalities tailored to these aggregated variables. Our computational experiments indicate that our model, post-initialization, provides a stronger lower bound, thereby accelerating the resolution of the root node. Furthermore, this approach seems to positively impact the branching procedure, leading to an overall faster Benders decomposition method.

Acknowledgements: The authors would like to thank dr. Chandra Irawan for providing the BIRCH instances.

1 Introduction

The p-median problem is one of the fundamental problems in location science (Laporte et al., 2019), where the aim is to select p facilities while minimizing the sum of distances from each location to its nearest facility. The p-median problem was introduced by Hakimi (1964) and is known to be NP-hard (Kariv and Hakimi, 1979). It has numerous practical applications (Rahmaniani et al., 2017; Laporte et al., 2019). Particularly, in the data mining and machine learning literature, the problem is referred to as the k-medoids problem (see e.g. Pinheiro et al. (2020)).

Benders decomposition is one of the most well-known solution methods to solve difficult combinatorial problems (Benders, 1962). Benders decomposition formulations for the p-median problem date back to Cornuejols et al. (1980) and Magnanti and Wong (1981). Benders decomposition has shown to be successful on a multitude of location problems (Fischetti et al., 2016b,a; Cordeau et al., 2019; Coniglio et al., 2022; Gaar and Sinnl, 2022; Ramírez-Pico et al., 2023; Ljubić et al., 2024). In particular, Duran-Mateluna et al. (2023) use a Benders decomposition method to solve the p-median problem for instances up to $238,000$ locations.

The goal in the p-median problem is to cover N locations by opening p out of M possible facilities, while minimizing the sum of distances from each location to its nearest facility. To simplify notation and without loss of generality, we assume that $N = M$. Let d_{ij} denote the distance from location i to a facility j and let $N_i \leq N$ be the number of unique distances from i to any facility; the sorted distances are denoted by $D_i^0 \leq D_i^1 \leq D_i^2 \leq \cdots \leq D_i^{N_i}$, where $D_i^0 = -\infty$ is the distance to an artificial facility. The binary variable y_j is equal to 1 if facility j is selected, and 0 otherwise. The continuous variable θ_i represents the distance from location i to the closest selected facility. Let $[N]$ be the set $\{1, 2, \ldots, N\}$ and $[0, N]$ be the set $\{0, 1, \ldots, N\}$. A compact Benders reformulation (Duran-Mateluna et al., 2023) can be written as

$$\min \sum_{i=1}^{N} \theta_i, \tag{1}$$

$$\text{s.t.} \sum_{j=1}^{N} y_j = p, \tag{2}$$

$$\theta_i \geq D_i^{n_i+1} - \sum_{j:d_{ij} \leq D_i^{n_i}} (D_i^{n_i+1} - d_{ij}) y_j, \qquad i \in [N], n_i \in [0, N_i - 1], \tag{3}$$

$$y_j \in \{0, 1\}, \qquad j \in [N]. \tag{4}$$

In (1) we minimize the sum of the distances from location i to the nearest facility. Exactly p facilities are selected due to (2), while the so-called Benders cuts are given in (3). In a Benders decomposition approach the method starts with a subset of Benders cuts. Afterwards, violated Benders cuts are identified and added to the model until no more violations can be identified.

The current bottleneck of a Benders decomposition approach for the p-median problem is that the mathematical model becomes too large, since the number of variables and Benders cuts is proportional to the number of locations N. In the worst case, N Benders cuts are added in every iteration of the Benders decomposition. Instead of solving the *disaggregated* model, one potential solution is to define an *aggregate* variable $\bar{\theta} = \sum_{i=1}^{N} \theta_i$ and aggregated Benders cuts

$$\bar{\theta} \geq \sum_{i=1}^{N} [D_i^{n_i+1} - \sum_{j:d_{ij} \leq D_i^{n_i}} (D_i^{n_i+1} - d_{ij}) y_j], \qquad \forall (n_1, n_2, \ldots, n_N) : l \in [N], n_l \in [0, N_l - 1],$$

which leads to a so-called fully aggregated mathematical formulation.

A fully aggregated Benders formulation has been investigated in location problems. For instance, by Fischetti et al. (2016b) and Ljubić et al. (2024), for the uncapacitated facility location problem and the discrete ordered median problem, respectively. They present both disaggregated (multicut) and fully aggregated (single cut) formulations that are solved using Benders decomposition. An aggregated model may be preferred since the reduced formulation can be solved faster. However, such a model tends to converge more slowly, as the aggregate constraints are less restrictive. For several related location problems, effective aggregation strategies have been developed by leveraging problem-specific structures. For the uncapacitated hub location problem, Contreras et al. (2011) design cuts tailored to the structure involving hubs. Ramírez-Pico et al. (2023) examined the so-called adaptive Benders cuts for two-stage stochastic programming. Their approach starts with a limited set of scenarios, where each Benders cut corresponds to one scenario. These cuts are then dynamically disaggregated, generating additional scenarios (and more Benders cuts).

In this paper, we propose a *partially* aggregated Benders decomposition method for the p-median problem, meaning that some locations may be aggregated. We propose three key steps. First, during initialization, we partially aggregate locations based on a start solution, such that location that are likely to be assigned to the same facility are in the same aggregate variable and aggregate cut. Second, to accelerate the Benders decomposition, we propose a set of valid inequalities to strengthen the formulation in initial iterations. Third, if branching is required to obtain an integer solution, we introduce aggregated binary decision variables to branch on.

Our contribution can be summarized as follows. First, we propose a partially aggregated Benders decomposition framework for the p-median problem, which contains both disaggregate and fully aggregated Benders decomposition as special cases. Second, we show how to adjust the state-of-the-art Benders decomposition method in order to solve the partially aggregated Benders decomposition. Last, we demonstrate the effectiveness of our method on both benchmark instances and newly introduced instances. In particular, our model, post-initialization, has a stronger lower bound compared to the disaggregate formulation, enabling faster resolution of the root node. Additionally, our proposed solution approach seems to positively impact the performance during the branch-and-cut phase.

The remainder of this paper is organized as follows. Related work is discussed in Section 2. In Section 3 we present a disaggregated Benders decomposition and a framework for the aggreggated variant. Section 4 outlines our solution approach for the aggregated Benders decomposition. Our computational experiments are presented in Section 5. Finally, conclusions are drawn in Section 6.

2 Related work

Several mixed-integer programming formulations have been developed to model the p-median problem (ReVelle and Swain, 1970; Cornuejols et al., 1980; Magnanti and Wong, 1981; Elloumi, 2010) and connections between these formulations are still being investigated (Duran-Mateluna et al., 2023; Agra and Requejo, 2024). Numerous methods have been proposed to solve these p-median formulations to optimality. Senne et al. (2005) introduced a branch-and-price algorithm that incorporates stabilized column generation and Lagrangian relaxation, enabling them to solve instances up to 900 locations. Avella et al. (2007) propose a branch-and-cut-and-price algorithm with delayed column and row generation, allowing for the resolution of instances with nearly 3800 locations. García et al. (2011) consider a column generation approach that is embedded in a branch-and-bound framework, successfully solving instances up to $85,900$ locations and $p = 70,000$ facilities. Ren et al. (2022) implement a (parallel) branch-and-bound method integrated with Langrangian relaxation, achieving a 0.1% optimality gap for instances with up to $100,000$ locations on a single core and instances with up to 1 million locations when using 6000 cores.

Due to the large number of locations and facilities, heuristics have been designed to aggregate locations, such that optimization models remain applicable (Irawan and Salhi, 2015a). The p-median problem can be partitioned into smaller subproblems that can be solved within a reasonable amount of time. However, this results in a loss of information, as the model no longer utilizes the original location data. The difference between the optimal objective value and the one obtained using an aggregation heuristic is known as the aggregation error. Hillsman and Rhoda (1978) formally defined the aggregation error for the p-median (and related) problems. Current and Schilling (1987) were the first to study the elimination of aggregation errors, while Goodchild (1979) demonstrated that aggregation errors have a large impact on the results, potentially leading to inaccurate objective values. For Euclidean p-median problems in the plane, smaller error bounds have been proven (Qi and Shen, 2010). Aggregation techniques have been used to enhance heuristics for solving large scale p-median problems, e.g., Avella et al. (2012), Irawan et al. (2014), Irawan and Salhi (2015b) and Salhi and Irawan (2015). Although aggregation is typically done at the location level, it can also be applied at the facility level (Avella et al., 2012). Further details can be found in Irawan and Salhi (2015a).

In this paper, we show how to apply aggregation in the current state-of-the-art Benders decomposition method from Duran-Mateluna et al. (2023), without any aggregation error.

3 Benders decomposition framework

In this section, we present a disaggregated Benders decomposition formulation, which is used by the current state-of-the art Benders decomposition method to solve the p-median problem. Afterwards, we introduce an aggregated Benders decomposition framework, which contains as special cases the disaggregated and fully aggregated Benders decomposition.

3.1 Disaggregated Benders decomposition

Due to the large number of Benders cuts of the form (3), Duran-Mateluna et al. (2023) propose to solve (1)–(4) using Benders decomposition. The main idea of Benders decomposition is to iteratively solve a restricted master problem (RMP) on a subset of Benders cuts of the form (3), denoted by $\mathcal{B}_i$ for each location i. A solution to the RMP may violate a Benders cut not yet in $\mathcal{B}_i$. In this case, we apply a separation algorithm to identify violated Benders cuts, which are subsequently added to the RMP. This iterative process continues until a feasible, possibly fractional, solution is obtained. We apply branch-and-Benders cut (Rahmaniani et al., 2017) to obtain an optimal solution, so we keep track of one set of cuts throughout the branching tree. The RMP can be formulated as

$$
\begin{aligned}
&\min \sum_{i=1}^{N} \theta_i, && && (5)\\
&\text{s.t.} \sum_{j=1}^{N} y_j = p, && && (6)\\
&\qquad \theta_i \text{ satisfies } \mathcal{B}_i, && i \in [N], && (7)\\
&\qquad y_j \geq 0, && j \in [N], && (8)
\end{aligned}
$$

Note that (5) and (6) are equal to (1) and (2), respectively. The integrality constraints on y_i are relaxed in (8).

Let $(\mathbf{y}, \boldsymbol{\theta})$ be a feasible, possibly fractional, solution to the RMP. We must verify whether there exists Benders cuts that are not yet in the RMP and violate the given solution. For a fixed solution, we can consider a separation problem for each location in $[N]$. For an efficient implementation to identify violated Benders cuts we refer to Algorithm 1 in Duran-Mateluna et al. (2023), which runs in $\mathcal{O}(N)$ for each location in $[N]$.

3.2 Aggregated Benders decomposition

In this section we present a (partially) aggregated Benders decomposition framework. A priori we define which distance variables θ_i are aggregated. Let $P = \{Q_1, \ldots, Q_R\}$ be a partition of the set of locations into R disjoint subsets, satisfying $Q = \cup_{r=1,\dots,R} Q_r = [N]$ and $Q_r \cap Q_s = \emptyset, \forall r \neq s$. We denote the aggregated variables by $\bar{\theta}_{Q_r}$, such that $\bar{\theta}_{Q_r} = \sum_{i \in Q_r} \theta_i$. When $R = N$ we obtain the disaggregated Benders decomposition, whereas setting $R = 1$ leads to a fully aggregated Benders decomposition. For each set Q_r we define *aggregated* Benders cuts

$$\bar{\theta}_{Q_r} \geq \sum_{i \in Q_r} [D_i^{n_i+1} - \sum_{j: d_{ij} \leq D_i^{n_i}} (D_i^{n_i+1} - d_{ij}) y_j], \quad \forall (n_1, n_2, \dots, n_N) : l \in Q_r, n_l \in [0, N_l - 1]. \tag{9}$$

This leads to the following Aggregated Restricted Master Problem (ARMP)

$$\min \sum_{r=1}^{R} \bar{\theta}_{Q_r}, \tag{10}$$

$$\text{s.t.} \sum_{i=1}^{N} y_i = p, \tag{11}$$

$$\bar{\theta}_{Q_r} \text{ satisfies } \bar{\mathcal{B}}_r, \qquad r \in [R], \tag{12}$$

$$y_i \geq 0, \qquad i \in [N]. \tag{13}$$

In (10) we minimize the aggregated distance variables. As before, (11) ensures that we select exactly p facilities, while (12) enforces the feasibility of the current set of aggregated Benders cuts ($\bar{\mathcal{B}}_r$) for each set Q_r.

4 Aggregated Benders decomposition for Euclidean distances

In the remainder of this paper we assume that the locations lie in a k-dimensional Euclidean space. To solve the aggregated Benders decomposition we propose to add three additional steps to the existing disaggregated Benders decomposition from Duran-Mateluna et al. (2023). First, based on a start solution we partially aggregate the distance variables. Second, we introduce a set of valid inequalities for the aggregate distance variables in order to strengthen the lower bound in the initial iterations of the Benders decomposition. Third, we add aggregate binary variables before starting the branching procedure.

4.1 Constructing a partition

Suppose we have a feasible start solution, represented by a partition $P = \{Q_1, \dots, Q_p\}$. Each set Q_r contains one selected facility $m_r \in Q_r$. We define the set of locations outside Q_r as $Q_r^o = Q \setminus Q_r$, which we refer to as outside locations. The distance between a location i and a set of locations Q is defined as $d(i, Q) = \min_{j \in Q} d_{ij}$. The shortest distance from Q_r^o to the selected facility m_r is $\rho_r = d(m_r, Q_r^o)$, which we call the *radius* of set Q_r. Based on the radius, we split each Q_r into a group of central locations Q_r^c and remaining locations Q_r^e, which are defined below.

Definition 4.1 (Central locations)**.** Given a set Q_r and a radius ρ_r the central locations are in the set $Q_r^c = \{i \in Q_r : d_{im_r} < \frac{1}{3}\rho_r\}$.

Definition 4.2 (Remaining locations)**.** The remaining locations are given by the set $Q_r^e = Q_r \setminus Q_r^c$.

The concepts of central and remaining locations are visualized in Figure 1. In this figure, the yellow locations represent set Q_r, where we assume that $m_r = 1$ is the selected facility. The locations in Q_r are partitioned into a set of central locations Q_r^c and a set of remaining locations Q_r^e, based on radius ρ_r. The radius ρ_r is calculated as the shortest distance from the selected facility $m_r = 1$ to the nearest location outside the cluster, which is in this case a location in $Q_{r'}$ (represented by blue dots).

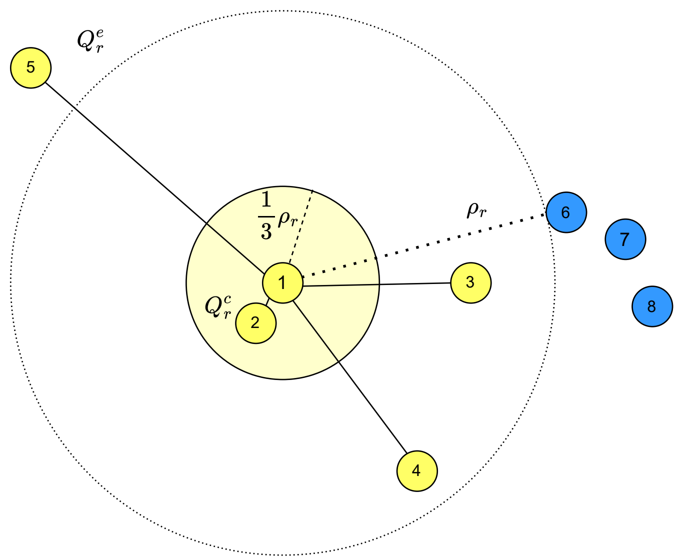


Figure 1: An example of a set $Q_r = \{1,2,3,4,5\}$ (corresponding to yellow dots) which is divided into two groups $Q_r^c = \{1,2\}$ and $Q_r^e = \{3,4,5\}$ based on a radius ρ_r, determined by the distance from $m_r = 1$ to the closest location from another cluster $Q_{r'} = \{6,7,8\}$ (corresponding to blue dots).

For each set Q_r we have one aggregated variable $\bar{\theta}_{Q_r^c}$ corresponding to set Q_r^c and several non-aggregated variables θ_i for each location $i \in Q_r^e$. Thus, when constructing the ARMP in (10)–(13), the model contains both aggregated and disaggregated Benders cuts.

The motivation behind the division into two groups Q_r^c and Q_r^e is as follows. Consider a good initial solution with m_r as the selected facility and suppose that in an optimal solution, m_r is not selected as facility. If another location from the set Q_r^c is selected as a facility, we can construct (strong) lower bounds for the partial objective $\bar{\theta}_{Q_r^c}$. Similarly, when a location from the set Q_r^e or Q_r^o is selected as facility we derive (weaker) lower bounds on $\bar{\theta}_{Q_r^c}$. These bounds can be added to the formulation through valid inequalities.

4.2 Valid inequalities

The main idea of the valid inequalities is to impose, for each $r \in [R]$, a lower bound on the (partial) objectives $\bar{\theta}_{Q_r^c}$ and $\theta_s, \forall s \in Q_r^e$. We propose three types of valid inequalities, which are added for each set Q_r.

4.2.1 Valid inequalities 1

We introduce three types of constants. First, for each central location $q \in Q_r^c$ let us define

$$\delta_{rq}^c = \sum_{i \in Q_r^c} d_{iq},$$

i.e., δ_{rq}^c is the 1-median objective for the central locations $i \in Q_r^c$ when $q \in Q_r^c$ is selected as the unique facility in Q_r. Second, for each remaining location $q \in Q_r^e$, we define

$$\delta_{rq}^e = \sum_{i \in Q_r^c} \min\{d_{iq}, d(i, Q_r^o)\}$$

The term δ_{rq}^e is a lower bound on the partial objective for the central locations $i \in Q_r^c$ when $q \in Q_r^e$ is selected as the unique facility in Q_r. Since outside locations in set Q_r^o may also be selected as a facility we take the minimum between distance d_{iq} and the distance from i to its nearest outside location, represented by $d(i, Q_r^o)$. Third, let D_r^c be the sum of distances from each location $i \in Q_r^c$ to its closest outside location, i.e.

$$D_r^c = \sum_{i \in Q_r^c} d(i, Q_r^o).$$

Consider the following valid inequality

$$\bar{\theta}_{Q_r^c} \geq D_r^c - \sum_{q \in Q_r^c} (D_r^c - \delta_{rq}^c) y_q - \sum_{q \in Q_r^e} (D_r^c - \delta_{rq}^e) y_q. \tag{14}$$

The valid inequality has a similar interpretation to a disaggregated Benders cut (3). When exactly one facility q is selected, the terms D_r^c cancel out and the remaining terms δ_{rq}^c and δ_{rq}^e represent a lower bound on the partial objective $\bar{\theta}_{Q_r^c}$. When no facility is selected in Q_r, a lower bound of D_r^c remains. By construction it holds that $D_r^c \geq \delta_{rq}^c$ and $D_r^c \geq \delta_{rq}^e$. For some locations $q \in Q_r^e$ it holds that $D_r^c = \delta_{rq}^e$, meaning that selecting these locations as facility does not change the lower bound in the valid inequality.

In Lemma 4.3 we prove that when exactly one location $q \in Q_r$ is selected as facility and this location belongs to the central locations Q_r^c, then the bound is tight: $\bar{\theta}_{Q_r^c} = \delta_{rq}^c$.

Lemma 4.3. *Assume without loss of generality that $y_q = 1$ for some $q \in Q_r^c$. If $\sum_{i \in Q_r^c} y_i = \sum_{i \in Q_r} y_i = y_q = 1$, then $\bar{\theta}_{Q_r^c}$ is equal to δ_{rq}^c.*

Proof. The distance d_{ij} between any two central locations $i, j \in Q_r^c$ is at most $\frac{2}{3}\rho_r$. Similarly, the distance d_{ij} from a central location $i \in Q_r^c$ to an outside location $j \in Q_r^o$ is at least $\frac{2}{3}\rho_r$. Since $y_q = 1$ and no other location is selected, it must hold that all locations in Q_r^c are assigned to the same facility q. Thus, given that q is selected as facility in an optimal solution the partial objective $\bar{\theta}_{Q_r^c}$ is equal to δ_{rq}^c, which is the 1-median objective of set Q_r^c with q as facility. □

In the next theorem, we prove the validity of (14).

Theorem 4.4. *Given a feasible solution $(\boldsymbol{\theta}, \mathbf{y})$, where the elements of vector $\mathbf{y}$ are integral, the valid inequalities (14) are correct.*

Proof.

We convert the solution vector $\mathbf{y}$ to a matrix $\mathbf{X}$, where x_{iq} takes the value 1 when location i is assigned to q and 0 otherwise. Note that $y_q \geq x_{iq}$. If q is selected as facility, $y_q = 1$, this does not always imply that i is assigned to q. Conversely, $y_q = 0$ implies that $x_{iq} = 0$ for all locations i.

We aim to prove the correctness of valid inequality (14), which is rewritten as

$$\bar{\theta}_{Q_r^c} + \sum_{q \in Q_r^c} (D_r^c - \delta_{rq}^c) y_q + \sum_{q \in Q_r^e} (D_r^c - \delta_{rq}^e) y_q - D_r^c \geq 0. \tag{15}$$

The realized partial objective can be expressed in terms of x_{iq} variables as

$$\begin{aligned}
\bar{\theta}_{Q_r^c} &= \sum_{q \in (Q_r^c \cup Q_r^e \cup Q_r^o)} \sum_{i \in Q_r^c} d_{iq} x_{iq}, \\
&\geq \sum_{q \in Q_r^o} \sum_{i \in Q_r^c} d(i, Q_r^o) x_{iq} + \sum_{q \in (Q_r^c \cup Q_r^e)} \sum_{i \in Q_r^c} d_{iq} x_{iq}
\end{aligned}$$

Using the definitions of D_r^c and δ_{rq}^c we bound the first summation in the left-hand side of (15) as

$$\sum_{q \in Q_r^c} (D_r^c - \delta_{rq}^c) y_q,$$

$$
\begin{aligned}
&= \sum_{q\in Q_r^c}(\sum_{i\in Q_r^c} d(i,Q_r^o) - \sum_{i\in Q_r^c} d_{iq})y_q,\\
&= \sum_{q\in Q_r^c}\sum_{i\in Q_r^c}(d(i,Q_r^o) - d_{iq})y_q,\\
&\geq \sum_{q\in Q_r^c}\sum_{i\in Q_r^c}(d(i,Q_r^o) - d_{iq})x_{iq},
\end{aligned}
$$

where the inequality holds because $y_q \geq x_{iq}$. Similarly, the second summation in the left-hand side of (15) can be bounded as well

$$
\begin{aligned}
&\sum_{q\in Q_r^e}(D_r^c - \delta_{rq}^e)y_q,\\
&= \sum_{q\in Q_r^e}[\sum_{i\in Q_r^c} d(i,Q_r^o) - \sum_{i\in Q_r^c}\min\{d_{iq}, d(i,Q_r^o)\}]y_q,\\
&= \sum_{q\in Q_r^e}\sum_{i\in Q_r^c}[d(i,Q_r^o) - \min\{d_{iq}, d(i,Q_r^o)\}]y_q,\\
&\geq \sum_{q\in Q_r^e}\sum_{i\in Q_r^c}[d(i,Q_r^o) - \min\{d_{iq}, d(i,Q_r^o)\}]x_{iq},\\
&\geq \sum_{q\in Q_r^e}\sum_{i\in Q_r^c}[d(i,Q_r^o) - d_{iq}]x_{iq}.
\end{aligned}
$$

The first inequality holds because $y_q \geq x_{iq}$. When leaving out the minimization we obtain the second inequality.

Using the results above, the left hand side of (15) can be written as

$$
\begin{aligned}
&\bar{\theta}_{Q_r^c} + \sum_{q\in Q_r^c}(D_r^c - \delta_{rq}^c)y_q + \sum_{q\in Q_r^e}(D_r^c - \delta_{rq}^e)y_q - D_r^c\\
&\geq \sum_{q\in Q_r^o}\sum_{i\in Q_r^c} d(i,Q_r^o)x_{iq} + \sum_{q\in(Q_r^c\cup Q_r^e)}\sum_{i\in Q_r^c} d_{iq}x_{iq} + \sum_{q\in Q_r^c}\sum_{i\in Q_r^c}[d(i,Q_r^o) - d_{iq}]x_{iq} + \sum_{q\in Q_r^e}\sum_{i\in Q_r^c}[d(i,Q_r^o) - d_{iq}]x_{iq} - D_r^c\\
&= \sum_{q\in Q_r^o}\sum_{i\in Q_r^c} d(i,Q_r^o)x_{iq} + \sum_{q\in Q_r^c}\sum_{i\in Q_r^c} d(i,Q_r^o)x_{iq} + \sum_{q\in Q_r^e}\sum_{i\in Q_r^c} d(i,Q_r^o)x_{iq} - D_r^c\\
&= \sum_{i\in Q_r^c}\sum_{q\in(Q_r^c\cup Q_r^e\cup Q_r^o)} d(i,Q_r^o)x_{iq} - D_r^c\\
&= \sum_{i\in Q_r^c} d(i,Q_r^o)[\sum_{q\in(Q_r^c\cup Q_r^e\cup Q_r^o)} x_{iq}] - D_r^c\\
&= D_r^c - D_r^c\\
&= 0
\end{aligned}
$$

To conclude, valid inequality (14) is correct. □

4.2.2 Valid inequalities 2

We add valid inequalities for the disaggregated distance variables corresponding to each location $s \in Q_r^e$. Let D_s^0 be equal to the distance from s to its closest central location $q \in Q_r^c$, i.e., $D_s^0 = d(s, Q_r^c)$. In addition, we can calculate D_s^1 as follows

$$D_s^1 = \min_{q\in Q: d_{sq} > D_s^0} d_{sq}.$$

This leads to the second valid inequality

$$\theta_s \geq D_s^1 - \sum_{q\in Q: d_{sq}\leq D_s^0}(D_s^1 - d_{sq})y_q, \qquad \forall s \in Q_r^e, \tag{16}$$

which is a Benders cut up to distance D_s^0.

4.2.3 Valid inequalities 3

We add the following two types of valid inequalities

$$\bar{\theta}_{Q_r^c} \geq \sum_{q \in Q_r^c} d(q, Q \setminus \{q\})(1 - y_q), \tag{17}$$

$$\theta_s \geq d(s, Q \setminus \{s\})(1 - y_s), \qquad \forall s \in Q_r^e. \tag{18}$$

Valid inequality (17) states that when y_q is not selected as a facility, the partial objective $\bar{\theta}_{Q_r^c}$ must at least include the distance from q to its nearest neighbor. Similarly, (18) enforces that when y_s is not selected, the distance θ_s must be at least equal to the distance from s to its nearest neighbor.

4.3 Solution approach

We solve the ARMP using a similar approach to Duran-Mateluna et al. (2023), where we make use of their separation algorithm to identify violated Benders cuts. They propose to solve the root node using Algorithm 1. A start solution, represented by the vector $\mathbf{y}^h$, is fixed and the separation algorithm is executed in order to generate initial Benders cuts. The LP relaxation and the separation algorithm are iteratively run until no more violated Benders cuts are found. Note that in each iteration the lower bound is updated and an upper bound can be obtained by using a simple rounding heuristic.

Algorithm 1 Solving the root node.

Input: Start solution $\mathbf{y}^h$
Output: Best lower and upper bound LB^* and UB^*

```
1: Run the separation algorithm with y^h
2: while a violated cut has been identified do
3:     Add the violated cuts to the ARMP
4:     Solve the ARMP to obtain a solution y and a lower bound LB
5:     Run the separation algorithm with y
6:     Update LB* ← LB
7:     Calculate UB using a rounding heuristic on y
8:     Update UB* = min{UB*, UB}
9: end while
```

Our proposed solution approach is summarized in Algorithm 2. Based on a start solution, represented by a partition P, we partially aggregate the distance variables $\boldsymbol{\theta}$ and add a set of valid inequalities (14), (16)–(18), to strengthen the formulation. The root node is solved as described in Algorithm 1. When a fractional solution is obtained, we perform two improvement procedures outlined by Duran-Mateluna et al. (2023) to decrease the size of the model, namely constraint reduction and reduced cost fixing. Also, we add integrality constraints to the $\mathbf{y}$ variables. While we keep the original y_i variables, we also add aggregate integer $\bar{y}_{Q_r^c}$ variables, for the central locations in a set $Q_r^c \subseteq Q_r$. When the start solution is close to an optimal solution, it is likely that exactly one y_i variable is selected in the set Q_r^c. We expect that the inclusion of the aggregate variables $\bar{y}_{Q_r^c}$ helps the branching process. In summary, we propose Algorithm 2, which incorporates steps 1, 2 and 7 into a disaggregated Benders decomposition approach for the p-median problem.

Algorithm 2 Aggregated Benders decomposition.

Inputs:
Sorted distances $D_i^0 \leq D_i^1 \leq D_i^2 \leq \cdots \leq D_i^{N_i}$
A start solution, represented by $P = \{Q_1, \ldots, Q_p\}$

1: Partially aggregate the distance variables $\boldsymbol{\theta}$ based on P (see Section 4.1)
2: Add valid inequalities (14), (16)–(18) based on P (see Section 4.2)
3: Solve the root node using Benders decomposition (see Algorithm 1)
4: Stop if the solution is integral
5: Perform the improvement procedure of Duran-Mateluna et al. (2023)
6: Add integrality constraints to the y variables in (13)
7: Partially aggregate the decision variables $\bar{y}_{Q_r^c} := \sum_{q \in Q_r^c} y_q \in \mathbb{N}, \forall r \in [p]$
8: Apply branch-and-cut

4.4 Solving Benders decomposition with kd-tree

The separation algorithm requires the indices corresponding to the sorted distances, $D_i^0 \leq D_i^1 \leq D_i^2 \leq \cdots \leq D_i^{N_i}$. Duran-Mateluna et al. (2023) store these indices in a matrix $\mathbf{S}$ with a space complexity of $\mathcal{O}(N^2)$. In practice storing the entire $\mathbf{S}$ may be unnecessary, as only the first K_i indices from $\mathbf{S}$ are typically utilized for each location i. To improve memory efficiency, we propose to store the N locations in a tree-based data structure to dynamically compute the required indices. Tree-based structures have been successfully applied to heuristic methods for Euclidean location problems (Salhi and Irawan, 2015). We utilize a kd-tree, which is a binary search tree with a space complexity of $\mathcal{O}(N)$. A kd-tree enables efficient retrieval of the nearest location in $\mathcal{O}(\log N)$ time. Additionally, it can be augmented with an efficient K-nearest neighbors algorithm.

In our approach, we dynamically identify the relevant indices during the separation algorithm. Specifically, for location i, we use a kd-tree to find its K_i-nearest neighbors. The process terminates when the separation algorithm can be solved with these K_i indices, otherwise we increase K_i (see Appendix A.3 for implementation details) and reattempt to solve the separation algorithm.

5 Computational results

In this section we present a computation study on several types of instances. First, we outline the instance types and parameters settings that are used. Next, we compare the disaggregated Benders decomposition with our aggregated one on a wide range of instances. We then investigate the impact of varying the quality of the start solution and assess the contribution of each proposed step and valid inequality in the aggregated Benders decomposition. Last, we show that using a kd-tree data structure can improve the performance of both the disaggregated and aggregated Benders decomposition.

5.1 Experimental setup

5.1.1 Overview of the instances

We evaluate our methods on the same Euclidean instances as Duran-Mateluna et al. (2023), namely TSP instances (Reinelt, 1991; Beasley, 1990) and BIRCH instances. We follow the notation of Duran-Mateluna et al. (2023) to categorize TSP instances as 'medium' and 'huge'. Additionally, we introduce three new sets of instances. First, we consider a set of huge TSP instances with low values of p, named TSP-huge-low-p. Second, we introduce a new category of even larger TSP instances, called TSP*. To the best of our knowledge, it is the first time in the literature that p-median instances of such magnitude are considered using an exact solver without massive parallelization. Third, we include a new set of CIRCLE instances, generated following the procedure described by Irawan et al. (2014). The instance categories that we consider are detailed in Table 1. For all instances, the locations are given by two-dimensional coordinates in Euclidean space.

The CIRCLE instances, introduced by Irawan et al. (2014), are constructed to ensure well-separated clusters, enabling a geometric argument to provide a proof of optimality for these p-median instances. These instances represent an ideal scenario for our algorithm, as the valid inequalities (14), (16)–(18) also provide bounds. If these valid inequalities are sufficiently strong, we expect our algorithm to identify an optimal solution within a few Benders iterations.

For the TSP instances, the distance between locations is calculated as the Euclidean distance rounded down to the nearest integer as done in García et al. (2011) and Duran-Mateluna et al. (2023). The maximum rounding error between the actual and rounded distance is given by $\epsilon_{rounding} \geq |d_{true} - d_{rounded}|$. We modify Definition 4.1 such that locations $q \in Q_r$ satisfying $d(q, m_r) < \frac{1}{3}\rho_r - \epsilon_{rounding}$ belong to Q_r^c. Since distances are rounded down to the nearest integer, the maximum rounding error is $\epsilon_{rounding} = 1$. For the BIRCH and CIRCLE instances we set $\epsilon_{rounding} = 10^{-6}$.

Table 1: Overview of the instances, including the number of observations (N), number of facilities (p), the rounding error ($\epsilon_{rounding}$) and whether the instances are considered for the first time in the literature.

instance category	N		p		$\epsilon_{rounding}$	new instances
	min	max	min	max		
TSP-medium	2103	5934	10	500	1	
TSP-huge	71009	238025	10000	200000	1	
TSP-huge-low-p	71009	238025	5	100	1	yes
TSP*	498378	744710	350000	700000	1	yes
BIRCH	25000	89600	25	64	10^{-6}	
CIRCLE	20000	80000	10	40000	10^{-6}	yes

5.1.2 Technical specifications

The experiments are carried out on an AMD Rome 7H12 processor 3.2 GHz with 1 TB RAM, although we limit the memory to 120 or 500 GB depending on the instance size. The MIP problems are solved using the commercial solver CPLEX 20.1. We use the same parameter settings for CPLEX as Duran-Mateluna et al. (2023), which are summarized in Table B1. The separation algorithm is implemented within the GenericCallback of CPLEX and gets called when a feasible integral solution is found, similar to Duran-Mateluna et al. (2023). Additionally, we apply the callback when a fractional solution is identified, which offers a slight improvement (see Table C1).

5.1.3 Computation times

We impose a time limit of 10 hours on the Benders decomposition method as described in Algorithm 2. This is stricter than the limit used in Duran-Mateluna et al. (2023), where a 10 hour time limit for the branch-and-cut phase is applied only after solving the root node.

Similar to Duran-Mateluna et al. (2023) we assume the instance data structure, e.g. matrix $\mathbf{S}$, and a start solution are given as input to the Benders decomposition method. These computation times are not included in the time limit. However, it is important to note that in some cases the calculation of the sorted distances needed to construct matrix $\mathbf{S}$ may be longer than the running time of the Benders decomposition method. Let T^{init}, T^{root} and $T^{B\&C}$ be, respectively, the time to initialize the aggregate model, the time to solve the root node, and the entire time of the branch-and-cut procedure (including the initialization and root node solving time). See Figure A1 for more details on the relation between the reported computation times.

Instances provably solved to optimality within the time limit are highlighted in bold, while instances from existing benchmark datasets that are solved to optimality for the first time in the literature are marked with a †.

5.1.4 Start heuristic

Start solutions are usually generated using a metaheuristic named PopStar (Resende and Werneck, 2004) or a k-means++ algorithm (see Appendix A.1 and A.2). In the k-means++($iter_1$, $iter_2$) algorithm, the parameters $iter_1$ and $iter_2$ specify the maximum number of restarts for the entire method and the number of inner iterations, respectively. As noted by Duran-Mateluna et al. (2023), the popStar heuristic becomes computationally expensive for large instances. To address this, we replace the popStar heuristic with k-means++(10,10) algorithm for large instances. Furthermore, due to the high memory requirements to store the matrix $\mathbf{S}$ we allocate up to 500 GB RAM to some instances. These settings are summarized in Table 2. Setting A of the Benders decomposition is applied to TSP-medium instances, while setting B is used for most other instances. Setting C is specifically applied when solving a Benders decomposition with a kd-tree.

Table 2: Overview of the different settings of the Benders decomposition for the distance data structure, memory limit and start heuristic.

setting	distance data structure	memory limit (in GB)	start heuristic
A	matrix $\mathbf{S}$	120	popStar
B	matrix $\mathbf{S}$	500	k-means++(10, 10)
C	kd-tree	120	k-means++(10, 10)

5.2 TSP instances

5.2.1 Comparison between disaggregated and aggregated Benders decomposition

Table 3 shows the results of the disaggregated and aggregated Benders decomposition with setting A on TSP-medium instances. While initializing the aggregated Benders decomposition takes on average 3 seconds, it reduces the average time required to solve the root node from 124 to 66 seconds. In addition, aggregation seems to reduce the time spent on the branch-and-cut, since the average computation time decreases from around 6200 to 4000 seconds. The disaggregated Benders is not able to solve five instances, compared to two when using the aggregated Benders.

The results with setting B for the TSP-huge instances are shown in Table 4. On average 220 seconds are spent on the initialization step of the aggregated Benders decomposition, which reduces the root node solving time from around 6200 seconds to 2000 seconds. Aggregation seems to have less impact on the branch-and-cut procedure compared to the TSP-medium instances.

In the literature, we were unable to find experiments that consider TSP-huge instances with low values of p. Table 5 presents the results on TSP-huge instances with values of p ranging from 5 to 100. The disaggregated Benders fails to complete a single Benders iteration for any of these instances. In contrast, the aggregated Benders decomposition successfully solves several Benders iterations, such that a gap can be calculated. Additionally, one instance with $p = 10$ is solved to optimality using branch-and-cut within the time limit.

5.2.2 Changing the start heuristic

We investigate the impact of the quality of the start solution on the solution time. The solutions obtained from k-means++(1,1) serve as a baseline, representing random start solutions. The solutions from k-means++(1,10) and k-means++(10,10) represent improved start solutions. We also evaluate popStar, a state-of-the-art heuristic that expected to outperform the k-means++ heuristics. Finally, we analyze an ideal scenario where the aggregated Benders decomposition is initialized with optimal solutions.

Table 6 shows the computation times for various start solutions. Changing the start solution has minimal impact on the time required to solve the root node, which averages between 63 and 74 seconds.

Table 3: Time (in s) and gap (in %) using disaggregated and aggregated Benders decomposition with setting A on TSP-medium instances. The number of iterations, number of aggregated variables and number of valid inequalities are denoted by iter, vars and cons, respectively. TL=36000 seconds.

instance			disaggregated Benders				aggregated Benders							
name	N	p	gap	T^{root}	$T^{B\&C}$	iter	UB	gap	T^{init}	T^{root}	$T^{B\&C}$	iter	vars	cons
d2103	2103	10	**0.00**	15	33	8	**687321**	**0.00**	3	4	11	6	179	3868
d2103	2103	20	**0.00**	21	37	10	**482926**	**0.00**	3	12	29	9	206	3834
d2103	2103	50	0.03	36	TL	12	**302190**†	**0.00**	2	22	12726	18	335	3636
d2103	2103	100	**0.00**	24	3728	12	**194664**	**0.00**	1	17	3057	17	395	3616
d2103	2103	200	**0.00**	4	7	11	**117753**	**0.00**	1	2	6	11	494	3616
d2103	2103	300	**0.00**	5	8	15	**90471**	**0.00**	0	2	3	17	360	4084
d2103	2103	400	**0.00**	2	10	10	**75324**	**0.00**	1	1	4	7	439	4118
d2103	2103	500	**0.00**	2	5	13	**64006**	**0.00**	1	1	5	14	524	4144
pcb3038	3038	10	**0.00**	68	68	8	**1211704**	**0.00**	2	15	17	6	301	5494
pcb3038	3038	20	**0.00**	84	498	10	**839494**	**0.00**	3	62	373	10	304	5508
pcb3038	3038	50	**0.00**	60	390	10	**506339**	**0.00**	4	32	249	10	346	5484
pcb3038	3038	100	**0.00**	55	236	11	**351500**	**0.00**	1	32	335	8	395	5486
pcb3038	3038	200	**0.00**	22	117	9	**237399**	**0.00**	1	12	109	9	416	5644
pcb3038	3038	300	**0.00**	12	23	11	**186833**	**0.00**	2	5	18	15	450	5776
pcb3038	3038	400	**0.00**	6	9	8	**156276**	**0.00**	2	2	6	10	489	5898
pcb3038	3038	500	**0.00**	4	6	11	**134798**	**0.00**	1	2	3	10	563	5948
fl3795	3795	10	**0.00**	13	13	7	**520940**	**0.00**	7	5	12	7	1678	4254
fl3795	3795	20	**0.00**	10	10	10	**319722**	**0.00**	5	2	7	7	1295	5040
fl3795	3795	50	**0.00**	8	8	12	**150940**	**0.00**	5	4	8	8	988	5714
fl3795	3795	100	**0.00**	9	9	12	**88299**	**0.00**	5	5	9	9	858	6074
fl3795	3795	200	**0.00**	14	1067	14	**53928**	**0.00**	2	11	843	25	871	6238
fl3795	3795	300	0.00	8	TL	17	**39586**	**0.00**	1	11	1575	35	940	6300
fl3795	3795	400	**0.00**	9	496	18	**31354**	**0.00**	1	8	274	23	1095	6190
fl3795	3795	500	**0.00**	7	7	15	**25976**	**0.00**	1	6	7	17	1116	6348
rl5934	5934	10	**0.00**	1203	10901	10	**9792218**	**0.00**	10	545	2665	10	517	10854
rl5934	5934	20	0.00	906	TL	11	6716215	0.00	5	596	TL	16	602	10704
rl5934	5934	50	0.00	631	TL	13	4030771	0.00	6	341	TL	17	678	10612
rl5934	5934	100	0.03	377	TL	12	**2722527**	**0.00**	7	219	30553	16	805	10458
rl5934	5934	200	**0.00**	139	747	9	**1805530**	**0.00**	4	83	1516	14	1066	10136
rl5934	5934	300	**0.00**	94	140	10	**1392419**	**0.00**	5	29	79	11	1346	9776
rl5934	5934	400	**0.00**	62	332	10	**1143940**	**0.00**	3	14	191	10	1536	9596
rl5934	5934	500	**0.00**	53	77	13	**972799**	**0.00**	2	11	26	13	1745	9378
avg			0.00	124	6218			0.00	3	66	3960			

However, the branch-and-cut procedure exhibits greater variability in computation time. Initializing the aggregated Benders decomposition with an optimal solution results in an average computation time of 3700 seconds. The best performing method, popStar, achieves computation times comparable to those obtained when initializing with an optimal solution. This is because popStar produces start solutions with objective values close to the optimal solutions.

5.2.3 Contribution of each step in the aggregated Benders decomposition

Recall that our approach adds three steps to the disaggregated Benders decomposition, namely partially aggregating the distance variables, adding valid inequalities and partially aggregating the decision variables. Table 7 reports the time spent when incrementally adding these steps with setting A on TSP-medium instances. We first analyze the time spent solving the root node. Activating the first step, which aggregates the θ variables, reduces the average time from 124 to 99 seconds. Incorporating the second step, adding valid inequalities, further reduces the time to an average of 66 seconds. A similar trend is observed in the branch-and-cut procedure. The average computation time decreases from around 6200 seconds to 5000 seconds with the first step, and further to 3900 seconds when activating the second step. The third step, adding the aggregate y variables, seems to have minimal impact on

Table 4: Time (in s) and gap (in %) using disaggregated and aggregated Benders decomposition with setting B on TSP-huge instances. The number of iterations, number of aggregated variables and number of valid inequalities are denoted by iter, vars and cons, respectively. TL=36000 seconds.

instance			disaggregated Benders				aggregated Benders							
name	N	p	gap	T^{root}	$T^{B\&C}$	iter	UB	gap	T^{init}	T^{root}	$T^{B\&C}$	iter	vars	cons
ch71009	71009	10000	0.04	11455	TL	12	4275352	0.02	46	2396	TL	11	14298	131800
ch71009	71009	20000	0.00	970	TL	17	2377850	0.00	39	551	TL	13	24164	124908
ch71009	71009	30000	**0.00**	524	632	19	**1464151**	**0.00**	45	414	612	14	33985	111266
ch71009	71009	40000	**0.00**	237	338	21	**879336**	**0.00**	32	377	526	12	43522	91274
ch71009	71009	50000	**0.00**	114	227	16	**463553**	**0.00**	20	115	251	12	53010	65656
ch71009	71009	60000	**0.00**	54	184	18	**167565**	**0.00**	13	32	177	14	62307	35330
pla85900	85900	10000	9.10	TL	TL	5	178470093	0.00	66	3506	TL	16	11924	167918
pla85900	85900	20000	0.00	12950	TL	15	118414166	0.00	63	4361	TL	11	21284	167692
pla85900	85900	30000	5.16	TL	TL	80	**86944715**†	**0.00**	65	210	1582	9	30640	159818
pla85900	85900	40000	**0.00**	46	168	27	**69944715**	**0.00**	49	204	677	9	40364	141964
pla85900	85900	50000	**0.00**	36	181	15	**52944715**	**0.00**	52	163	487	8	50194	117922
pla85900	85900	60000	**0.00**	79	239	21	**35944715**	**0.00**	41	129	406	9	60132	89418
pla85900	85900	70000	**0.00**	39	130	14	**18977475**	**0.00**	29	44	169	15	70086	57936
pla85900	85900	80000	**0.00**	12	114	26	**4512752**	**0.00**	17	14	150	24	80047	22796
usa115475	115475	10000	27.80	TL	TL	3	8765753	3.48	181	12097	TL	13	18743	213244
usa115475	115475	20000	0.01	6764	TL	13	5287528	0.01	155	2314	TL	13	27387	213600
usa115475	115475	30000	0.00	2642	TL	13	3815715	0.00	131	1481	TL	16	36006	210336
usa115475	115475	40000	**0.00**	1706	2114	17	**2876909**	**0.00**	125	1119	2565	13	44847	201514
usa115475	115475	50000	**0.00**	1109	1391	17	**2189144**	**0.00**	81	999	1555	16	53749	188094
usa115475	115475	60000	**0.00**	781	1061	14	**1651400**	**0.00**	69	990	1439	13	62897	169688
usa115475	115475	70000	**0.00**	629	943	16	**1214299**	**0.00**	77	878	1364	15	72168	146742
usa115475	115475	80000	**0.00**	348	690	24	**851481**	**0.00**	73	385	889	12	81534	120476
usa115475	115475	90000	**0.00**	202	541	16	**548097**	**0.00**	53	197	703	11	91034	90436
ara238025	238025	10000	0.32	9199	TL	12	1397282	3.69	665	6007	TL	12	34524	426868
ara238025	238025	20000	0.01	10280	TL	17	857548	0.00	854	6100	TL	13	50455	414808
ara238025	238025	30000	**0.00**	6812	28560	15	630983	0.00	699	3320	TL	12	65516	404188
ara238025	238025	40000	**0.00**	4630	5529	15	**494842**	**0.00**	629	2520	4560	11	78632	396704
ara238025	238025	50000	**0.00**	4183	5224	13	**401835**	**0.00**	627	1915	4283	13	88883	393260
ara238025	238025	60000	**0.00**	4722	5859	14	**334279**	**0.00**	541	2073	4733	11	97436	391192
ara238025	238025	70000	**0.00**	3800	5128	22	**283627**	**0.00**	510	2142	4510	13	104098	389844
ara238025	238025	80000	**0.00**	3827	5423	18	**244233**	**0.00**	421	2888	5768	16	109809	388040
ara238025	238025	90000	**0.00**	3985	7648	12	**214233**	**0.00**	411	3004	8293	22	115810	382198
ara238025	238025	100000	**0.00**	3795	7081	12	184241	0.01	369	3843	TL	11	121186	375986
ara238025	238025	150000	**0.00**	2091	3785	283	**88025**	**0.00**	307	3609	8116	9	156729	291012
ara238025	238025	200000	**0.00**	750	2497	13	**38025**	**0.00**	146	738	3805	13	201882	140130
avg			1.21	6223	12734			0.21	220	2032	12961			

the computation time. In Appendix C.2 we also provide a detailed comparison of the performance of each individual type of valid inequalities.

5.2.4 Lower bound in the first iteration

Figure 2 visualizes the lower bound in the first iteration of the Benders decomposition with setting B for different values of p on TSP-huge instances. The root node lower bound (red line) indicates the optimal objective value when solving the LP relaxation. Ideally, we would like to be as close as possible to this objective value. The figure shows that the objective value in the first iteration when using aggregated Benders (blue line) is equal to or higher compared to disaggregated Benders (black line).

This leads to the question whether the objective value in subsequent Benders iterations is also higher for aggregated Benders compared to disaggregated Benders. Figure 3 shows the objective value per iteration of the Benders decomposition for two selected instances (the other instances exhibit similar behaviors). For the instances with $p = 10000$ we notice that the objective value of the aggregated

Table 5: Time (in s) and gap (in %) using disaggregated and aggregated Benders decomposition with setting B on TSP-huge-low-p instances. The number of iterations, number of aggregated variables and number of valid inequalities are denoted by iter, vars and cons, respectively, for TL=36000 seconds.

instance			disaggregated Benders				aggregated Benders							
name	N	p	gap	T^{root}	$T^{B\&C}$	iter	UB	gap	T^{init}	T^{root}	$T^{B\&C}$	iter	vars	cons
ch71009	71009	5	-	TL	TL	0	279898768	2.49	26107	TL	TL	5	7643	126742
ch71009	71009	10	-	TL	TL	0	**184135561**	**0.00**	13962	7255	24698	10	7184	127670
ch71009	71009	50	-	TL	TL	0	82188868	7.49	2692	TL	TL	4	6666	128786
ch71009	71009	100	-	TL	TL	0	58002900	10.69	1627	TL	TL	4	6785	128648
pla85900	85900	5	-	TL	TL	0	9852614013	-	TL	TL	TL	0	6687	158436
pla85900	85900	10	-	TL	TL	0	7067159042	4.96	17240	TL	TL	3	6646	158528
pla85900	85900	50	-	TL	TL	0	3282348405	7.05	3441	TL	TL	3	7093	157714
pla85900	85900	100	-	TL	TL	0	2330305935	8.06	1712	TL	TL	3	6984	158032
usa115475	115475	5	-	TL	TL	0	514450906	-	TL	TL	TL	0	-	-
usa115475	115475	10	-	TL	TL	0	361392225	-	TL	TL	TL	0	-	-
usa115475	115475	50	-	TL	TL	0	152557601	7.97	20182	TL	TL	3	11410	208230
usa115475	115475	100	-	TL	TL	0	107188213	10.76	5158	TL	TL	3	10788	209574
ara238025	238025	5	-	TL	TL	0	80129403	-	TL	TL	TL	0	-	-
ara238025	238025	10	-	TL	TL	0	56557295	-	TL	TL	TL	0	-	-
ara238025	238025	50	-	TL	TL	0	24926972	-	TL	TL	TL	0	-	-
ara238025	238025	100	-	TL	TL	0	17482258	14.01	14955	TL	TL	2	19785	436680

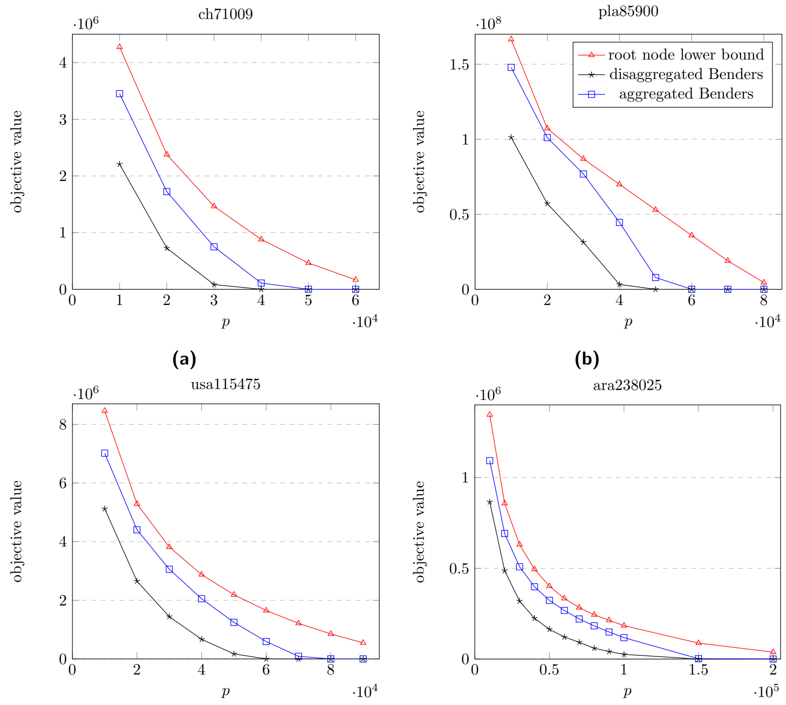
Table 6: Average time (in s) when using different start solutions for aggregated Benders decomposition on TSP-medium instances. Times are averaged over each instance with the same name. We also report the average gap between the start solution and optimal (or best-known) solution. TL=36000 seconds.

instance		disaggregated Benders		aggregated Benders									
		popStar		kmeans++ (1, 1)		kmeans++ (1, 10)		kmeans++ (10, 10)		popStar		optimal	
name	N	T^{root}	$T^{B\&C}$	T^{root}	$T^{B\&C}$	T^{root}	$T^{B\&C}$	T^{root}	$T^{B\&C}$	T^{root}	$T^{B\&C}$	T^{root}	$T^{B\&C}$
d2103	2103	14	4979	8	3663	9	4927	9	3452	8	1978	8	1694
pcb3038	3038	39	168	24	209	20	241	20	241	20	136	19	116
fl3795	3795	10	4701	10	4667	8	774	8	515	6	339	7	694
rl5934	5934	433	15025	254	18185	243	18226	214	15405	230	13375	235	12321
avg		124	6218	74	6681	70	6042	63	4903	66	3957	67	3706
avg gap		0.1%		9.0%		5.8%		4.7%		0.1%		0.0%	

Table 7: Average time (in s) while adding more steps to the aggregated Benders decomposition with setting A on TSP-medium instances. Times are averaged over each instance with the same name. TL=36000 seconds.

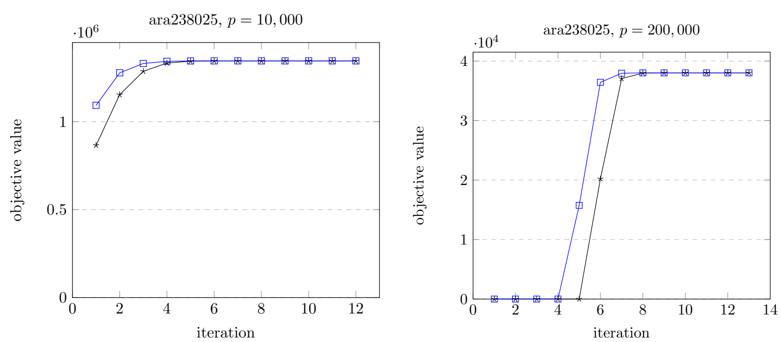
instance		disaggregated Benders		aggregate θ		add inequalities		aggregate y	
name	N	T^{root}	$T^{B\&C}$	T^{root}	$T^{B\&C}$	T^{root}	$T^{B\&C}$	T^{root}	$T^{B\&C}$
d2103	2103	14	4979	15	4889	8	1996	8	1978
pcb3038	3038	39	168	31	170	20	135	20	136
fl3795	3795	10	4701	9	1235	6	340	6	339
rl5934	5934	433	15025	341	13936	229	13291	230	13375
avg		124	6218	99	5057	66	3940	66	3957

Benders is higher for around 5 iterations. For the other instance with $p = 200000$ we notice that both lines start with an objective of 0, but the aggregated Benders jumps earlier to a non-negative objective value compared to disaggregated Benders.



(a) (b) (c) (d)

Figure 2: Lower bound in the first iteration for TSP-huge instances.



(a) (b)

Figure 3: Lower bound in each iteration for two TSP-huge instances.

5.3 BIRCH instances

In Table 8 we compare the disaggregated and aggregated Benders decomposition with setting B on the BIRCH instances. The time to solve the branch-and-cut procedure for aggregated Benders is on average 5100 seconds compared to 2700 seconds for disaggregated Benders. This can be explained by the observation that for the aggregated Benders most time is spent on initializing the aggregate model. This shows that adding all possible valid inequalities might not be a good idea. A possible improvement would be to add a limited number of valid inequalities.

Table 8: Time (in s) and gap (in %) using disaggregated and aggregated Benders decomposition with setting B on BIRCH instances. The number of iterations, number of aggregated variables and number of valid inequalities are denoted by iter, vars and cons, respectively. TL=36000 seconds.

instance			disaggregated Benders				aggregated Benders							
name	N	p	gap	T^{root}	$T^{B\&C}$	iter	UB	gap	T^{init}	T^{root}	$T^{B\&C}$	iter	vars	cons
BIRCH1	25000	25	**0.00**	143	144	5	**31229.3**	**0.00**	515	30	546	3	9251	31548
BIRCH2	36000	36	**0.00**	2975	2977	6	**45115.6**	**0.00**	662	50	714	5	12144	47784
BIRCH3	49000	49	**0.00**	380	383	5	**61384.1**	**0.00**	1540	70	1612	7	16736	64626
BIRCH4	64000	64	**0.00**	786	790	6	**80053.9**	**0.00**	3097	94	3195	6	21229	85670
BIRCH5	30000	25	**0.00**	191	193	6	**37563.6**	**0.00**	1502	61	1564	4	10974	38102
BIRCH6	43200	36	**0.00**	4612	4616	7	**54191.4**	**0.00**	1546	105	1654	6	13151	60170
BIRCH7	58800	49	**0.00**	18860	18864	6	**73626.8**	**0.00**	2361	123	2487	7	20113	77472
BIRCH8	76800	64	**0.00**	1301	1307	7	**96039.4**	**0.00**	4978	160	5142	7	23995	105738
BIRCH9	35000	25	**0.00**	511	514	6	**43902.1**	**0.00**	1377	84	1462	4	12802	44446
BIRCH10	50400	36	**0.00**	9741	9745	6	**63169.2**	**0.00**	1586	123	1713	5	16707	67458
BIRCH11	68600	49	**0.00**	752	758	6	**85833.5**	**0.00**	4727	193	4925	8	21725	93848
BIRCH12	89600	64	**0.00**	1612	1619	6	**112059.2**	**0.00**	15120	289	15415	6	28116	123096
BIRCH21	25000	25	**0.00**	272	273	6	**17696.2**	**0.00**	920	198	1119	20	10223	29604
BIRCH22	36000	36	**0.00**	371	373	8	**27423.0**	**0.00**	989	1047	2037	24	14343	43386
BIRCH23	49000	49	**0.00**	688	692	10	**44149.0**	**0.00**	1894	916	2813	33	16259	65580
BIRCH24	64000	64	**0.00**	834	839	11	**58832.6**	**0.00**	3002	1348	4353	29	24925	78278
BIRCH25	30000	25	**0.00**	402	405	8	**21829.9**	**0.00**	1042	460	1504	22	10984	38082
BIRCH26	43200	36	**0.00**	530	535	12	**32339.4**	**0.00**	2341	594	2938	11	15211	56050
BIRCH27	58800	49	**0.00**	1033	1038	9	**50857.9**	**0.00**	3549	837	4390	14	18498	80702
BIRCH28	76800	64	**0.00**	1842	14981	15	**66562.4**	**0.00**	3843	8397	14355	49	35007	83714
BIRCH29	35000	25	**0.00**	507	510	11	**24810.9**	**0.00**	1368	675	2045	26	13700	42650
BIRCH30	50400	36	**0.00**	687	691	13	**38102.6**	**0.00**	2826	6739	9568	36	19658	61556
BIRCH31	68600	49	**0.00**	1378	1385	14	**61850.6**	**0.00**	4421	2544	6969	33	24432	88434
BIRCH32	89600	64	**0.00**	2486	2496	20	**78777.0**	**0.00**	18731	12229	30967	53	38790	101748
avg			0.00	2204	2755			0.00	3497	1557	5145			

On average the disaggregated method needs 2200 seconds for solving the root node and around 500 seconds for the branch-and-cut, leading to a total running time of 2700 seconds. In contrast, the aggregated Benders requires on average 3500 seconds to add the valid inequalities, 1500 seconds to solve the root node and 100 seconds for the branch-and-cut, leading to a total computation time of 5100 seconds. Although the aggregated Benders speeds up the root node and branch-and-cut phases, the time spent adding inequalities is too high.

5.4 CIRCLE instances

The CIRCLE instances are designed to ensure that the clusters are well-separated, which facilitates the process of obtaining high-quality initial solutions. Note that this is the first time these instances have been addressed using an exact method in the literature. The main idea of our introduced valid inequalities is to calculate a lower bound that holds for several solutions close to the provided start solution. Thus, we expect our aggregation to perform well for the CIRCLE instances.

This expectation seems to be confirmed in Table 9, where we compare disaggregated and aggregated Benders decomposition with setting B. The time to initialize the model is on average 3300 seconds, this reduces the time to solve the root node from on average 8300 seconds for the disaggregated to 280 seconds for the aggregated Benders decomposition, representing an improvement of nearly a factor of 30. The total time spent on the branch-and-cut phase is reduced from on average 8300 seconds for the disaggregated to 3600 seconds for the aggregated Benders decomposition, reducing the computation time by more than half.

Table 9: Time (in s) and gap (in %) using disaggregated and aggregated Benders decomposition with setting B on CIRCLE instances. The number of iterations, number of aggregated variables and number of valid inequalities are denoted by iter, vars and cons, respectively. TL=36000 seconds.

instance			disaggregated Benders				aggregated Benders							
name	N	p	gap	T^{root}	$T^{B\&C}$	iter	UB	gap	T^{init}	T^{root}	$T^{B\&C}$	iter	vars	cons
C20000_10	20000	10	**0.00**	3604	3606	4	**15749249.9**	**0.00**	921	53	973	2	11000	18020
C20000_50	20000	50	**0.00**	188	188	4	**3298500.2**	**0.00**	145	4	149	3	9442	21216
C20000_100	20000	100	**0.00**	159	159	6	**1797000.1**	**0.00**	74	3	77	4	7191	25818
C20000_500	20000	500	**0.00**	177	177	7	**442500.2**	**0.00**	28	2	30	4	7890	25220
C20000_1000	20000	1000	**0.00**	104	105	7	**217499.9**	**0.00**	18	2	20	5	7465	27070
C20000_5000	20000	5000	**0.00**	24	26	6	**45000.0**	**0.00**	13	12	26	6	5001	39738
C20000_10000	20000	10000	**0.00**	16	20	43	**15000.0**	**0.00**	17	7	24	5	10000	36534
C80000_10	80000	10	-	TL	TL	0	245998500.1	0.00	32801	3199	TL	3	44000	72020
C80000_50	80000	50	-	TL	TL	0	**50396999.9**	**0.00**	8987	232	9219	3	38874	82352
C80000_100	80000	100	-	TL	TL	0	**26393999.9**	**0.00**	4528	65	4593	4	34362	91476
C80000_500	80000	500	**0.00**	6524	6526	8	**5985000.1**	**0.00**	893	20	913	4	32715	95570
C80000_1000	80000	1000	**0.00**	3491	3493	8	**2985000.0**	**0.00**	501	10	512	5	32279	97442
C80000_5000	80000	5000	**0.00**	1409	1417	7	**689999.7**	**0.00**	231	27	258	4	30191	109618
C80000_10000	80000	10000	**0.00**	971	986	7	**330000.0**	**0.00**	199	234	432	5	29209	121558
C80000_40000	80000	40000	**0.00**	381	436	57	**60000.0**	**0.00**	166	347	513	6	40000	146200
avg			0.00	8337	8343			0.00	3302	281	3583			

5.5 Solving Benders decomposition with kd-tree

5.5.1 TSP-huge instances

The computational results for solving Benders decomposition using a kd-tree on TSP-huge instances are presented in Table 10, comparing both disaggregated and aggregated Benders decompositions. Column T^{read} shows the time required to read the data and constructing the distance data structure, which is either the **S** matrix or a kd-tree. Column T^{heur} reports the time taken by the k-means++(10,10) heuristic.

Introducing a kd-tree increases the time needed for the branch-and-cut procedure, $T^{B\&C}$. This is due to the additional time taken by the separation algorithm, while the time to solve the master problem remains unchanged, as can be observed in Table C4. However, utilizing a kd-tree decreases the total computation time, consisting of T^{read}, T^{heur} and $T^{B\&C}$. The use of a kd-tree reduces T^{read} from approximately 1300 to 2 seconds. A similar effect is observed on T^{heur}, which decreases from around 2800 to 60 seconds.

In summary, applying a kd-tree eliminates the need to compute and store the sorted matrix **S**, making Benders decomposition with a kd-tree both time and memory efficient.

Table 10: Average time (in s) and gap (in %) using disaggregated and aggregated Benders decomposition with the matrix S or a kd-tree on TSP-huge instances. Times are averaged over each instance with the same name. TL=36000 seconds.

instance		disaggregated Benders								aggregated Benders							
		matrix **S** (setting B)				kdtree (setting C)				matrix **S** (setting B)				kdtree (setting C)			
name	N	gap	T^{read}	T^{heur}	$T^{B\&C}$	gap	T^{read}	T^{heur}	$T^{B\&C}$	gap	T^{read}	T^{heur}	$T^{B\&C}$	gap	T^{read}	T^{heur}	$T^{B\&C}$
ch71009	71009	0.01	318	745	12230	0.08	1	17	12387	0.01	318	744	12261	0.01	1	17	8835
pla85900	85900	0.00	405	1022	13604	0.00	2	27	13648	0.00	408	1026	9434	0.00	1	27	9459
usa115475	115475	0.00	768	1726	12749	0.01	1	40	16556	0.39	768	1719	12946	0.00	1	39	13101
ara238025	238025	0.03	2924	5806	12395	0.41	5	119	16171	0.31	2897	5921	15673	0.34	4	118	16770
avg		0.01	1347	2796	12734	0.16	2	60	15045	0.21	1338	2834	12961	0.12	2	59	12795

5.5.2 TSP* instances

Table 11 presents results for both disaggregated and aggregated Benders decomposition with a kd-tree (setting C) on TSP* instances. Instance *lrb744710* would theoretically require 2 TB of RAM to store matrix **S** when using the start-of-the-art Benders decomposition. In contrast, we solve several instances to optimality on a node with 120 GB of RAM (in practice even less memory is used). The disaggregated and aggregated Benders decomposition have similar computation times.

Table 11: Time (in s) and gap (in %) using disaggregated and aggregated Benders decomposition with a kd-tree (setting C) on TSP* instances. The number of iterations, number of aggregated variables and number of valid inequalities are denoted by iter, vars and cons, respectively. TL=36000 seconds.

instance			disaggregated Benders				aggregated Benders							
name	N	p	gap	T^{root}	$T^{B\&C}$	iter	UB	gap	T^{init}	T^{root}	$T^{B\&C}$	iter	vars	cons
lra498378	498378	350000	0.91	14195	TL	19	170044	0.18	1401	11735	TL	14	378838	467188
lra498378	498378	400000	**0.00**	8659	9109	44	**98378**	**0.00**	961	9201	13729	13	419166	326004
lra498378	498378	450000	**0.00**	3870	4004	19	**48379**	**0.00**	545	4467	6215	16	460110	166738
lrb744710	744710	600000	7.63	TL	TL	7	151951	100.63	2413	TL	TL	5	604540	535604
lrb744710	744710	650000	**0.00**	15615	15615	13	**94710**	**0.00**	1624	15871	17495	14	652888	359412
lrb744710	744710	700000	**0.00**	8448	8448	20	**44710**	**0.00**	877	8737	9614	20	701366	173186
avg			1.42	14465	18196			16.80	1303	14335	19842			

6 Conclusion & discussion

We consider the p-median problem, where the goal is to select p facilities such that the sum of distances between each location and its nearest facility is minimized. The bottleneck in the current state-of-the-art Benders decomposition is the large number of variables in the formulation.

We propose a partially aggregated Benders decomposition framework, which contains no and full aggregation as a special case. In our solution approach, we partially aggregate the variables based on a start solution. We aggregate variables for a group of so-called central locations, which are locations that we expect to be assigned to the same facility. We develop valid inequalities for the central locations and the remaining locations.

In our numerical experiments, we show that these valid inequalities strengthen the initial Benders iterations. Specifically, after initializing the model we often obtain an improved lower bound, resulting in a faster resolution of the root node. Across all types of instances, incorporating the valid inequalities reduces the average time required to solve the root node, in some cases by nearly a factor 30. Additionally, for certain instances, these valid inequalities positively influence the branch-and-cut phase, reducing the computation time by more than half.

However, for some instances, we observe that identifying a large number of valid inequalities can slow down the overall method. The additional time required to add these valid inequalities may offset the speedup gained from having stronger valid inequalities in the first Benders iterations. For further research, it may be interesting to strike a balance between the number of valid inequalities and the obtained speed up.

The approach we present may be generalized to instances with asymmetric distances. However, it may be challenging to define a group of central locations, particularly because the distance from a location to itself may be non-zero. Further research is required to develop a procedure for determining such a group of central locations and corresponding valid inequalities.

Appendix

A Details on the algorithms

A.1 The k-means++ algorithm

The k-means++ algorithm, also known as Lloyd's algorithm, is summarized in Algorithm 3. The method is initialized with a random selection of centroids, followed by an iterative assignment and recalculation of centroids. The maximum number of restarts and internal iterations are $iter_1$ and $iter_2$, respectively.

Algorithm 3 **The k-means++ algorithm**

1: Choose initial centroids
2: **while** objective has not converged or iteration limit, $iter_2$, has not been reached **do**
3: Assignment step: assign points to closest centroid
4: Update step: recalculate the centroids based on the assignment
5: **end while**

A.2 Solving k-means++ with kd-tree

The assignment step usually has a complexity of $\mathcal{O}(kN)$, which can be reduced by using a kd-tree data structure to $\mathcal{O}(k \log N)$. We construct a kd-tree containing the coordinates of all the centroids and assign each location $i \in [N]$ to its nearest centroid by performing a nearest neighbor look up.

After obtaining the centroid solution using the k-means++ algorithm, we convert it to a p-median solution. We construct a kd-tree containing all N locations. For each centroid, we efficiently identify the nearest location which is labeled as a facility. These facilities are inserted into a new kd-tree. Finally, we assign each location $i \in [N]$ to the nearest facility.

A.3 Solving the separation algorithm with kd-tree

When solving the separation algorithm with a kd-tree we initially limit our search to the first K_i indices, for each location in $[N]$. We initialize $K_i = 5$ for all locations. If the separation algorithm cannot be solved because the current K_i indices are insufficient, we perform the following update on K_i:

$$K_i = K_i + \max\{10, \lfloor 0.1 \cdot \max_{j \in [N]} \{K_j\} \rfloor\}.$$

This update increments K_i by 10 or by 10% of the current maximum K_i value across all locations, whichever is larger.

B Parameters

Table B1: Values of the CPLEX parameters as suggested by Duran-Mateluna et al. (2023).

parameter name	value	description
RelGap	10^{-10}	the relative tolerance to the best integer objective
AbsGap	0.9999	the absolute tolerance to the best integer objective
MIP Emphasis	BestBound	focus on proving optimality
BRDIR	1	branch up first

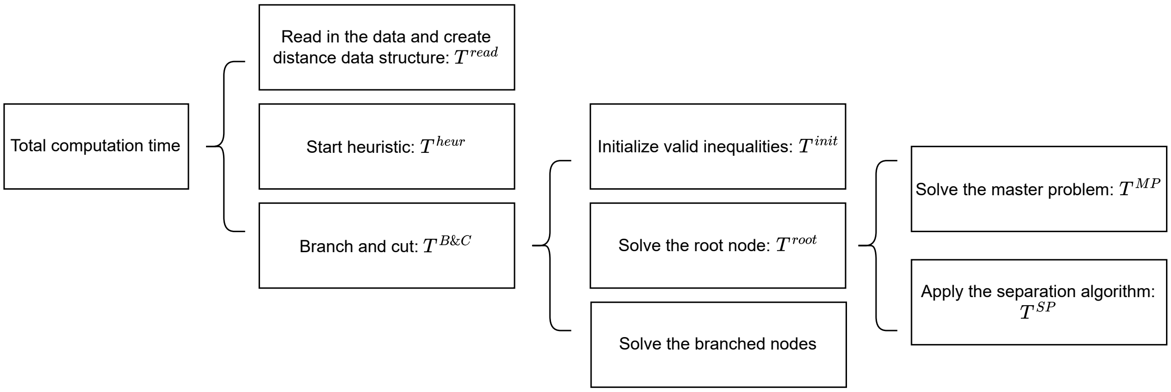


Figure A1: Overview of the notation for the reported computation times and how they relate to each other.

C Computational results

C.1 Different callback settings

Table C1: Time (in s) using disaggregated Benders decomposition with setting B on TSP-medium instances. The GeneralCallback in CPLEX gets called either when a candidate solution is found or additionally when a relaxation solution is identified. TL=36000 seconds.

instance			candidate only	candidate and relaxation
name	N	p	$T^{B\&C}$	$T^{B\&C}$
d2103	2103	10	32.9	32.6
d2103	2103	20	63.8	37.1
d2103	2103	50	TL	TL
d2103	2103	100	TL	3728.5
d2103	2103	200	7.3	7.2
d2103	2103	300	7.7	7.9
d2103	2103	400	5.1	10.0
d2103	2103	500	9.6	4.5
pcb3038	3038	10	67.3	68.4
pcb3038	3038	20	780.3	498.1
pcb3038	3038	50	270.7	390.5
pcb3038	3038	100	327.1	235.7
pcb3038	3038	200	140.3	116.8
pcb3038	3038	300	32.3	23.4
pcb3038	3038	400	14.1	8.7
pcb3038	3038	500	6.4	6.4
fl3795	3795	10	13.5	13.4
fl3795	3795	20	9.9	10.0
fl3795	3795	50	7.8	7.8
fl3795	3795	100	9.0	9.0
fl3795	3795	200	5648.9	1066.7
fl3795	3795	300	9060.8	TL
fl3795	3795	400	1186.1	495.9
fl3795	3795	500	7.5	7.5
rl5934	5934	10	9446.6	10901.4
rl5934	5934	20	TL	TL
rl5934	5934	50	TL	TL
rl5934	5934	100	TL	TL
rl5934	5934	200	1927.4	746.7
rl5934	5934	300	151.6	140.1
rl5934	5934	400	485.9	331.6
rl5934	5934	500	95.6	76.6
			6556.7	6218.2

C.2 Contribution of each type of valid inequality

We evaluate the effectiveness of the three types of proposed valid inequalities, namely (14), (16) and (17)–(18). Specifically, we examine the reduction in solving time at the root node and the branch-and-cut procedure. Additionally, we assess the quality of the lower bound (the objective value of the RMP) obtained in the first iteration of the Benders decomposition. The percentage improvement in the lower bound is calculated relative to the one obtained from the disaggregated Benders decomposition.

In Table C2 we compare the effectiveness of the individual valid inequalities for TSP-medium instances. Valid inequalities (14) are the strongest, improving the lower bound during the first iteration of the Benders decomposition by on average 25%, compared to the disaggregated Benders, while valid inequalities (16) result in the fastest computation times. Interestingly, these running times seem to suggest that adding valid inequalities (16) is sufficient, even though they have a smaller impact on the initial lower bound.

Table C2: Average time (in s) when adding one type of valid inequality to the aggregated Benders decomposition with setting A on TSP-medium instances. Times are averaged over each instance with the same name. We also report the number of valid inequalities added and the improvement (in %) in the lower bound achieved during the first iteration, compared to disaggregated Benders. The last column reports the improvement (in %) when adding all valid inequalities. TL=36000 seconds.

instance		valid inequalities (14)				valid inequalities (16)				valid inequalities (17)–(18)				all
name	N	T^{root}	$T^{B\&C}$	# cuts	improv	T^{root}	$T^{B\&C}$	# cuts	improv	T^{root}	$T^{B\&C}$	# cuts	improv	improv
d2103	2103	16	7185	196	28.7	8	2393	1737	16.4	13	4329	1932	17.7	46.0
pcb3038	3038	36	253	197	19.2	17	161	2630	14.3	33	211	2827	8.9	35.5
fl3795	3795	11	860	195	31.8	7	396	2690	15.8	12	881	2885	6.6	48.7
rl5934	5934	330	13398	198	21.9	185	12363	4897	15.3	358	15754	5095	1.9	33.1
avg		98	5424	196	25.4	54	3828	2988	15.4	104	5274	3185	8.8	40.8

The performance of TSP-huge instances differs noticeably from that of TSP-medium instances, as can be seen in Table C3. For larger instances, valid inequalities (14) and (17)–(18) improve the initial lower bound by 92% and 111%, respectively. In contrast, valid inequalities (16) decrease the initial lower bound, caused by the aggregation of variables.

Each individual type of valid inequality improves the solving time of the root node. Additionally, only valid inequalities (14) improve the solving time of the branch-and-cut procedure, suggesting that these valid inequalities are sufficient.

Table C3: Average time (in s) when adding one type of valid inequality to the aggregated Benders decomposition with setting A on TSP-huge instances. Times are averaged over each instance with the same name. We also report the number of valid inequalities added and the improvement (in %) in the lower bound achieved during the first iteration, compared to disaggregated Benders. The last column reports the improvement (in %) when adding all valid inequalities. TL=36000 seconds.

instance		valid inequalities (14)				valid inequalities (16)				valid inequalities (17)–(18)				all
name	N	T^{root}	$T^{B\&C}$	# cuts	improv	T^{root}	$T^{B\&C}$	# cuts	improv	T^{root}	$T^{B\&C}$	# cuts	improv	improv
ch71009	71009	1472	6475	14225	101.4	803	12221	32461	-3.3	1282	12165	46686	132.2	170.6
pla85900	85900	6688	9624	17525	153.6	1243	9364	40316	-14.7	1992	9370	57842	174.1	195.4
usa115475	115475	5313	12842	25128	72.6	4911	15960	61212	3.0	5375	12608	74494	79.4	119.2
ara238025	238025	3604	13778	47148	60.5	2828	14025	135945	0.8	5223	17235	183093	82.6	121.3
avg		4383	11336	29071	91.9	2654	13148	77130	-2.9	3848	13378	103154	111.2	146.2

C.3 Detailed computational results when solving the root node

Table C4: Time (in s) of solving the root node using disaggregated and aggregated Benders decomposition with the matrix S or a kd-tree on TSP-huge instances. The time to spent on the master problem and separation algorithm are denoted by T^{MP} and T^{SP}, respectively. TL=36000 seconds.

instance			disaggregated Benders						aggregated Benders					
			matrix **S** (setting B)			kdtree (setting C)			matrix **S** (setting B)			kdtree (setting C)		
name	N	p	T^{MP}	T^{SP}	T^{root}	T^{MP}	T^{SP}	T^{root}	T^{MP}	T^{SP}	T^{root}	T^{MP}	T^{SP}	T^{root}
ch71009	71009	10000	11453	1	11455	12939	135	13074	2394	1	2396	1620	102	1721
ch71009	71009	20000	969	1	970	1021	140	1162	549	1	550	538	100	638
ch71009	71009	30000	523	1	524	498	124	622	413	1	414	388	77	465
ch71009	71009	40000	235	1	237	232	92	325	376	1	377	383	56	439
ch71009	71009	50000	114	1	114	140	109	250	113	1	114	114	51	165
ch71009	71009	60000	53	1	54	36	108	144	31	1	31	30	56	87
pla85900	85900	10000	TL	1	TL	TL	25	TL	3503	2	3505	3285	67	3352
pla85900	85900	20000	12949	1	12950	13134	68	13202	4359	1	4360	3213	38	3251
pla85900	85900	30000	TL	3	TL	35715	305	TL	209	1	210	265	24	289
pla85900	85900	40000	45	1	46	39	73	113	203	1	204	198	21	219
pla85900	85900	50000	35	1	36	40	46	87	162	1	162	199	27	226
pla85900	85900	60000	77	1	79	46	60	107	128	1	129	82	29	111
pla85900	85900	70000	38	1	39	26	37	64	42	1	43	42	35	76
pla85900	85900	80000	10	1	12	12	77	89	11	1	12	13	51	64
usa115475	115475	10000	TL	1	TL	TL	15	TL	12091	4	12095	33004	69	33073
usa115475	115475	20000	6762	2	6764	8348	51	8399	2309	3	2312	2134	47	2181
usa115475	115475	30000	2640	1	2642	2594	45	2640	1476	3	1479	1555	44	1599
usa115475	115475	40000	1704	1	1706	1654	40	1694	1115	2	1117	1370	29	1399
usa115475	115475	50000	1107	1	1109	1138	36	1175	995	2	997	1005	27	1032
usa115475	115475	60000	779	1	781	790	32	823	986	2	988	1038	22	1060
usa115475	115475	70000	627	1	629	505	37	543	874	2	876	883	27	909
usa115475	115475	80000	346	1	348	336	44	381	382	2	384	337	16	353
usa115475	115475	90000	201	1	202	206	35	242	193	2	195	203	18	221
ara238025	238025	10000	9190	8	9199	8116	1039	9157	5995	9	6004	5576	1582	7158
ara238025	238025	20000	10272	6	10280	9140	623	9765	6091	6	6097	5235	755	5990
ara238025	238025	30000	6806	5	6812	6153	453	6608	3312	4	3317	2539	538	3077
ara238025	238025	40000	4624	4	4630	4083	414	4498	2513	4	2517	1592	428	2019
ara238025	238025	50000	4178	4	4183	3837	370	4208	1907	4	1911	1852	373	2226
ara238025	238025	60000	4717	4	4722	3412	306	3719	2067	3	2071	2073	514	2586
ara238025	238025	70000	3794	5	3800	3573	544	4120	2135	4	2139	2223	188	2411
ara238025	238025	80000	3821	4	3827	3446	342	3789	2879	4	2884	2750	239	2989
ara238025	238025	90000	3980	3	3985	3676	247	3924	2993	5	2999	2777	189	2966
ara238025	238025	100000	3791	3	3795	3467	226	3694	3836	3	3840	3331	191	3522
ara238025	238025	150000	2031	31	2091	2030	4254	6307	3603	3	3606	3298	160	3458
ara238025	238025	200000	747	2	750	785	222	1007	730	4	733	712	242	954
avg			5903	3	5908	5805	308	6112	2028	2	2030	2453	184	2637

References

Agra, A. and Requejo, C. (2024). Revisiting a cornuéjols-nemhauser-wolsey formulation for the p-median problem. EURO Journal on Computational Optimization, 12:100081.

Avella, P., Boccia, M., Salerno, S., and Vasilyev, I. (2012). An aggregation heuristic for large scale p-median problem. Computers & Operations Research, 39(7):1625–1632.

Avella, P., Sassano, A., and Vasil'ev, I. (2007). Computational study of large-scale p-median problems. Mathematical Programming, 109:89–114.

Beasley, J. E. (1990). Or-library: Distributing test problems by electronic mail. The Journal of the Operational Research Society, 41(11):1069–1072.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik, 4(1):238–252.

Coniglio, S., Furini, F., and Ljubić, I. (2022). Submodular maximization of concave utility functions composed with a set-union operator with applications to maximal covering location problems. Mathematical Programming, 196(1):9–56.

Contreras, I., Cordeau, J. F., and Laporte, G. (2011). Benders decomposition for large-scale uncapacitated hub location. Operations Research, 59(6):1477–1490.

Cordeau, J., Furini, F., and Ljubić, I. (2019). Benders decomposition for very large scale partial set covering and maximal covering location problems. European Journal of Operational Research, 275(3):882–896.

Cornuejols, G., Nemhauser, G. L., and Wolsey, L. A. (1980). A canonical representation of simple plant location problems and its applications. SIAM Journal on Algebraic Discrete Methods, 1(3):261–272.

Current, J. and Schilling, D. (1987). Elimination of source a and b errors in p-median location problems. Geographical Analysis, 19(2):95–110.

Duran-Mateluna, C., Ales, Z., and Elloumi, S. (2023). An efficient benders decomposition for the p-median problem. European Journal of Operational Research, 308(1):84–96.

Elloumi, S. (2010). A tighter formulation of the p-median problem. Journal of combinatorial optimization, 19(1):69–83.

Fischetti, M., Ljubić, I., and Sinnl, M. (2016a). Benders decomposition without separability: a computational study for capacitated facility location problems. European Journal of Operational Research, 253(3):557–569.

Fischetti, M., Ljubić, I., and Sinnl, M. (2016b). Redesigning benders decomposition for large-scale facility location. Management Science, 63(7):2146–2162.

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

García, S., Labbé, M., and Marín, A. (2011). Solving large p-median problems with a radius formulation. INFORMS Journal on Computing, 23(4):546–556.

Goodchild, M. (1979). The aggregation problem in location-allocation. Geographical Analysis, 11(3):240–255.

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

Hillsman, E. L. and Rhoda, R. (1978). Errors in measuring distances from populations to service centers. The Annals of Regional Science, 12:74–88.

Irawan, C., Salhi, S., and Scaparra, M. (2014). An adaptive multiphase approach for large unconditional and conditional p-median problems. European Journal of Operational Research, 237(2):590–605.

Irawan, C. A. and Salhi, S. (2015a). Aggregation and non aggregation techniques for large facility location problems-a survey. Yugoslav Journal of Operations Research, 25(3):313–341.

Irawan, C. A. and 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(3):537–554.

Kariv, O. and Hakimi, S. L. (1979). An algorithmic approach to network location problems. ii: The p-medians. SIAM Journal on Applied Mathematics, 37(3):539–560.

Laporte, G., Nickel, S., and da Gama, F. S. (2019). Introduction to Location Science. Springer.

Ljubić, I., Pozo, M. A., Puerto, J., and Torrejón, A. (2024). Benders decomposition for the discrete ordered median problem. European Journal of Operational Research, 317:858–874.

Magnanti, T. L. and Wong, R. T. (1981). Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. Operations Research, 29(3):464–484.

Pinheiro, D. N., Aloise, D., and Blanchard, S. J. (2020). Convex fuzzy k-medoids clustering. Fuzzy Sets and Systems, 389:66–92.

Qi, L. and Shen, Z. (2010). Worst-case analysis of demand point aggregation for the euclidean p-median problem. European Journal of Operational Research, 202(2):434–443.

Rahmaniani, R., Crainic, T. G., Gendreau, M., and Rei, W. (2017). The benders decomposition algorithm: A literature review. European Journal of Operational Research, 259(3):801–817.

Ramírez-Pico, C., Ljubić, I., and Moreno, E. (2023). Benders adaptive-cuts method for two-stage stochastic programs. Transportation Science, 57(5):1252–1275.

Reinelt, G. (1991). Tsplib-a traveling salesman problem library. ORSA Journal on Computing, 3(4):376–384.

Ren, J., Hua, K., and Cao, Y. (2022). Global optimal k-medoids clustering of one million samples. Advances in Neural Information Processing Systems, 35:982–994.

Resende, M. G. C. and Werneck, R. F. (2004). A hybrid heuristic for the p-median problem. Journal of Heuristics, 10:59–88.

ReVelle, C. S. and Swain, R. W. (1970). Central facilities location. Geographical analysis, 2(1):30–42.

Salhi, S. and Irawan, C. (2015). A quadtree-based allocation method for a class of large discrete euclidean location problems. Computers & Operations Research, 55:23–35.

Senne, E. L., Lorena, L. A., and Pereira, M. A. (2005). A branch-and-price approach to p-median location problems. Computers & Operations Research, 32(6):1655–1664.