Global Optimization of Clustering Problems

by

Mingfei Shi

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Mathematics)

The University of British Columbia
(Vancouver)

April 2022

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Global Optimization of Clustering Problems

submitted by **Mingfei Shi** in partial fulfillment of the requirements for the degree of **Master of Science** in **Mathematics**.

Examining Committee:

Yankai Cao, Assistant Professor, Chemical and Biological Engineering, UBC
Supervisor

Bhushan Gopaluni, Professor, Chemical and Biological Engineering, UBC
Supervisory Committee Member

Philip D. Loewen, Professor, Department of Mathematics, UBC
Supervisory Committee Member

Abstract

Clustering is a fundamental unsupervised machine learning task that aims to aggregate similar data into one cluster and separate those in diverse into different clusters. Cluster analysis can always be formulated as an optimization problem. Various objective functions may lead to different clustering problems. In this thesis, we concentrate on k-means and k-center problems. Each can be formulated as a mixed-integer nonlinear programming problem. The work about k-means clustering optimization has been published on ICML 2021 [30]. Moreover, we also submitted the work about global optimization of k-center clustering to ICML 2022 and the paper has been accepted in Phase 1 of reviewing. This thesis provides a practical global optimization algorithm for these two tasks based on a reduced-space spatial branch and bound (BB) scheme. This algorithm can guarantee convergence to the global optimum by only branching on the centers of clusters, which is independent of the dataset's cardinality. We also design several methods to construct lower and upper bounds at each node in the BB scheme. In addition, for k-center problem, a set of feasibility-based bounds tightening techniques are proposed to narrow down the domain of centers and significantly accelerate the convergence. To demonstrate the capacity of this algorithm, we present computational results on UCI datasets and compare our proposed algorithms with the off-the-shelf global optimal solvers and classical local optimal algorithms. For k-means clustering, the numerical experiments demonstrated our algorithm's ability to handle datasets with up to 200,000 samples. Besides, for k-center clustering, the serial implementation of the algorithm on the dataset with 14 million samples and 3 features can attain the global optimum to an optimality gap of 0.1% within 2 hours.

Lay Summary

Clustering is a fundamental unsupervised machine learning task that plays a vital role in various fields of applications, such as customer grouping, data summarization, and facility location determination. However, investigating directly on the global solution of the k-means and k-center clustering is still in deficiency, especially for large datasets.

The first goal of this thesis is to design a practical global optimization algorithm for k-means and k-center clustering problems, which can guarantee convergence to the global optimum. The second goal of this thesis is to accelerate the convergence of this algorithm.

Preface

This thesis is an original work done in collaboration with and under the guidance of my supervisor Professor Yankai Cao.

This thesis concentrates on two clustering problems, i.e., k-means clustering and k-center clustering. The work about k-means clustering problem has been published on ICML 2021 [30] of which I am an author. Besides, the work about k-center clustering has been accepted in Phase 1 of reviewing at ICML 2022 at the time of this submission.

As the author of this thesis, I was responsible for the formulations, models and methods presented in this thesis. Moreover, I was also responsible for the computational experiments of k-center clustering in Chapter 4. I also wrote the codes related to k-center problem, and the complete code used for the k-center problem can be found in https://github.com/mingfei-shi/Kcenter. The computational experiments of k-means problem in Section 3.3 were conducted by another author of [30] and used with permission.

Table of Contents

List of Tables

List of Figures

Acknowledgements

First, I would like to offer my enduring gratitude to my supervisor, Professor Yankai Cao, for his insightful guidance and patient advising.

Furthermore, I would also like to thank Kaixun Hua and Jiayang Ren for their support. This work would not have been possible without them.

Chapter 1

Introduction

Clustering is a fundamental unsupervised machine learning task that plays a vital role in various fields of applications, such as customer grouping [2], data summarization [27, 34], and facility location determination [26]. Clustering aims to aggregate similar data into one cluster and separate those in diverse into different clusters [45]. Cluster analysis can always be formulated as an optimization problem. Various objective functions are designed and lead to different algorithms to find clusters [40], such as k-means and k-center clustering, which are the main concentrations of this thesis.

1.1 k-Means Clustering

k-means problem is a fundamental target of the clustering problems that minimize the within-cluster sum-of-squared-error or in short the minimum sum-of-squares criteria. It tends to minimize the distance between points to their corresponding cluster centers to achieve the best cohesion and separation of the resulted clusters [48]. There are many heuristic methods proposed for solving the k-means clustering task. For instance, the k-means clustering algorithm [39] provides a coordinate descent based method to produce a result for the k-means problem. However, due to the non-convexity of the k-means objectives, the classic k-means algorithm is sensitive to the initialization and easy to fall under the local minimum [52]. To overcome this limitation, several modifications to the classical k-means clustering algorithm

have been proposed in order to obtain the global optimal solutions for the k-means clustering [1, 38, 50, 52]. However, none of these works provide the deterministic guarantee of locating the global minimum. Investigating directly on the global solution of the k-means problem is still in deficiency.

It is well-known that clustering problems can be reformulated as mixed integer programming problems [21, 35, 47]. Specifically, the k-means clustering can be treated as a Mixed Integer Second Order Cone Programming (MISOCP) problem. Branch and bound (BB) scheme is the most widely used algorithm for solving these optimization problems to global optimality and is well implemented in several popular solving packages, like BARON [49], ANTIGONE [42], and SCIP [22]. The BB paradigm depends on the efficient evaluation of lower and upper bounds of the optimal solution. It is able to reduce the gap between lower and upper bounds based on two key principles, that is to partition the search spaces into smaller regions that can be solved recursively (e.g. branching), and to prune the search regions that it can prove will not contain an optimal solution (e.g. bounding) [43]. Nevertheless, the direct application of the BB scheme does not scale well with the size of samples because branching may need to be performed on all variables to guarantee convergence and the number of variables increases linearly as the number of samples. Thus, it is infeasible to find the global optimum of the clustering tasks using the classical BB scheme if the number of samples is moderate (e.g. 100 data samples).

To handle the issue, in the thesis, we reformulate the k-means clustering as a two-stage optimization problem. This is because several approaches have been proposed in the stochastic programming community focusing on improving the scalability of global search by exploiting the structure of two-stage problems. These methods include generalized Benders decomposition [24], nonconvex Benders decomposition [36, 37], and Lagrangian relaxation [10, 31, 33]. In this thesis, we adapted our recent work of [8] on the reduced-space BB scheme for two-stage optimization problems and tailored it for the k-means problem. The novelty of the approach proposed in [8] is that it guarantees the convergence to the global optimum by only branching on first-stage variables (branching on second-stage variables is performed implicitly during the computation of bounds). In the context of the clustering task, the centers of clusters are regarded as the first-stage variables,

while the binary variables indicating the class of each data sample are treated as the second-stage variables. It implies that the number of variables need to be partitioned on is independent of the dataset's cardinality.

1.2 k-Center Clustering

Another clustering problem we focus on in this thesis is the k-center problem. It picks a subset of k samples as centers to represent k clusters, and each sample is assigned to its closest center to form the cluster. The distance from a sample to its closest center is called the within-cluster distance. The objective of the k-center problem is to minimize the maximum within-cluster distance of the dataset [32]. As an NP-hard problem [25], there is no way to achieve an optimal solution in a polynomial-time unless $P = NP$ [23]. Therefore, many heuristic approximation algorithms are developed to quickly get a near-optimal feasible solution.

The 2-approximation greedy algorithm is one of the most effective heuristic ways to solve the k-center problem [25]. It starts from a randomly selected center, and then the new centers are chosen as the farthest points to the previously selected centers. [5] applied a complete-linkage-based algorithm that can outperform traditional algorithms with proper ordering heuristics for sample. However, the later experiments showed that it seldom finds optimal solutions [3]. Parametric pruning [28] transferred the k-center problem as finding a minimum dominating set in a pruned graph. It developed corresponding heuristics to solve the dominating set problems. An experimental comparison of these heuristic algorithms indicated the 2-approximation greedy algorithm as the fastest in practical [41]. Nevertheless, none of the above algorithms can guarantee a global optimal solution for the k-center problem.

The exact algorithms that attempt to solve the k-center problem to global optimum lie in many directions. Early seminal works focus on reducing searching space for the optimal solution. For example, [16] proposed an iterative algorithm that solves the k-center problem by performing a binary search over possible solution values. In each iteration, a set-covering problem is solved. [19] leveraged the binary search scheme and designed a new integer linear programming formulation of the k-center problem and generated tighter lower bounds than pure LP relaxations. It

was the first algorithm that could solve a dataset of size up to 1817 samples. Another direction to attain global optimum includes adapting a constraint programming framework. [18] proposed a general constraint clustering algorithm that can solve the datasets with up to 5000 samples in 50 seconds. [7] introduces a block of covering constraints for the formulation of k-center problem and updates the lower and upper bound of the problem in the same iteration. However, their algorithm does not perform well on large datasets [13].

Recently, the strategy of iterative solving reduced subproblems has been proposed by researchers to solve the k-center problem to global optimality on large datasets. [3] presented a sampling-based exact algorithm, which alternates between an exact procedure on a small sample of points and a heuristic procedure to test the optimality of the current solution. Their computational experiment shows that this algorithm can obtain a reasonable solution for a dataset containing 581,012 observations within 4 hours. However, this work does not report the optimality gap, an important index to evaluate the solution quality. [12] and [11] proposed an iterative algorithm based on relaxation but only considered a small subset of the data samples. They set up heuristics to iteratively update the subset samples to approach the global optimal. Moreover, [13] designed a row generation algorithm that iteratively solves a smaller subproblem, and reported the solution of a dataset with 1 million samples to a gap of 6% within 9 hours. However, none of these methods provides a convergence guarantee that the algorithm will converge to an arbitrarily small gap within a finite number of steps. Therefore, these methods often lead to a nontrivial optimality gap, especially for large datasets.

1.3 Contribution of The Thesis Work

In this thesis, we design an exact global optimization algorithm based on a reduced-space spatial branch and bound scheme for k-means clustering and k-center clustering problem. We prove that our algorithm is guaranteed to converge to the global optimum by only branching on the centers of clusters. We formulate the k-means problem and k-center problem into a two-stage optimization problem respectively and we propose decomposable approaches to compute lower bounds. Moreover, to generate tighter lower bounds, we propose adaptive grouping and Lagrangian

decomposition to construct lower bounding problems for the k-means problem. As for the k-center problem, we design several bounds tightening techniques to significantly reduce the search space of the BB algorithm and accelerate the solution process. We demonstrate that the assignment of clusters of many samples can be determined without knowing the optimal solution. Bounds tightening techniques coupled with the decomposable lower bounding techniques enable our algorithm to be extremely scalable.

Besides, to evaluate the performance of our algorithm, for the k-means problem, we present an open-source implementation of the proposed algorithm in `Julia` on both synthetic and real-world datasets. Our algorithm and implementation enlarge the application of finding the global optimum of k-means problem to the scale of datasets with up to 210,000 samples (200 cores, under 3% optimality gap within the runtime of 4 hours). Remarkably, numerical experiments on the several real-world datasets show that our implementation can converge to a small gap ($< 0.1\%$) under 12 hours in serial or 1 hour in parallel with small number of clusters, while current state of the art need a long time ($\geq$ 50 hours) or cannot solve. Moreover, for the k-center problem, we provide an implementation of the algorithm and performed extensive computational experiments on 29 UCI datasets. Our algorithm is significantly faster than the off-the-shelf global optimal solvers and provides much better solutions than heuristic approaches. Notably, for the dataset with 14 million samples and 3 features, the serial implementation of the algorithm can attain the global solution to an optimality gap of 0.1% within 2 hours.

Chapter 2

Branch and Bound Scheme

This chapter introduces a reduced-space branch and bound algorithm for k-means and k-center clustering. Section 2.1 presents the notation and formulations used in this thesis. The details of the branch and bound algorithm are proposed in Section 2.2.

2.1 Notation and Formulations

Let $X = \{x_1, \ldots, x_S\} \in \mathbb{R}^{A \times S}$ be a dataset with S samples and A attributes, in which $x_i = [x_{i,1}, ..., x_{i,A}] \in \mathbb{R}^A$ is the ith sample and $x_{i,a}$ is the ath attribute of ith sample. The followings are the notations used in this thesis.

- $s \in \mathscr{S} := \{1, \cdots, S\}$ is the data sample set.
- $k \in \mathscr{K} := \{1, \cdots, K\}$ is the cluster set.
- $\mu := [\mu^1, \cdots, \mu^K]$ is the center of each cluster.
- $\underline{\mu}$ is the lower bound of centers, i.e., $\underline{\mu}_a^k = \min_s X_{s,a}$, $\forall k \in \mathscr{K}$, $a \in \{1, \cdots, A\}$.
- $\bar{\mu}$ is the upper bound of centers, i.e., $\bar{\mu}_a^k = \max_s X_{s,a}$, $\forall k \in \mathscr{K}$, $a \in \{1, \cdots, A\}$.
- $M_0 := \{\mu \mid \underline{\mu} \leq \mu \leq \bar{\mu}\}$ is a closed set that represents the domain of centers.
- d_s^k is the distance between sample x_s and center μ^k.
- d_s^* is the distance between x_s and the center of its cluster.

- b_s^k is a binary variable. b_s^k is equal to 1 if sample x_s belongs to the kth cluster, and 0 otherwise.
- λ_s^k is a binary variable. λ_s^k is equal to 1 if μ^k is chosen on x_s, and 0 otherwise.
- Denote $d_s := [d_s^1, \cdots, d_s^K, d_s^*]$, $d := [d_1, \cdots, d_S]$, $b_s := [b_s^1, \cdots, b_s^K]$, $b := [b_1, \cdots, b_S]$, $\lambda_s := [\lambda_s^1, \cdots, \lambda_s^K]$, $\lambda := [\lambda_1, \cdots, \lambda_S]$.
- N, N_1 and N_2 are arbitrary large values.
- z^{km} is the optimal value of k-means problem. z^{kc} is the optimal value of k-center problem.
- Denote $M^k := \{\mu^k : \underline{\mu}^k \leq \mu^k \leq \bar{\mu}^k\}$ where $\underline{\mu}^k$ and $\bar{\mu}^k$ are the lower and upper bound of μ^k respectively.
- Denote $\delta(M_0) := ||\bar{\mu} - \underline{\mu}||_\infty$ as the diameter of the box set M_0.
- α is an upper bound and β is a lower bound.
- $d(x_s, T)$ is the distance from sample x_s to set T.

k-means clustering is a minimum sum-of-squares clustering task, which can be defined as:

$$\min_{\mu} \sum_{s \in \mathscr{S}} \sum_{k \in \mathscr{K}} b_s^k ||x_s - \mu^k||_2^2 \tag{2.1}$$

On the other hand, in the k-center clustering problem, the objective is to minimize the maximum within-cluster distance of the dataset, which can be formulated as:

$$\min_{\mu \in X} \max_{s \in \mathscr{S}} \min_{k \in \mathscr{K}} ||x_s - \mu^k||_2^2 \tag{2.2}$$

where we use $\mu \in X$ to denote the "centers on samples" constraint that the center of each cluster is restricted to some data sample.

It is well-known that clustering problems can be treated as mixed integer programming problems. Correspondingly, k-means problem has the following opti-

mization formulation:

$$\min_{\mu,d,b} \sum_{s\in\mathscr{S}} d_s^* \tag{2.3a}$$

$$\text{s.t. } d_s^k \geq ||x_s - \mu^k||_2^2 \tag{2.3b}$$

$$-N(1-b_s^k) \leq d_s^* - d_s^k \leq N(1-b_s^k) \tag{2.3c}$$

$$d \geq d_{s,*} \tag{2.3d}$$

$$\sum_{k\in\mathscr{K}} b_s^k = 1 \tag{2.3e}$$

$$b_s^k \in \{0,1\} \tag{2.3f}$$

$$s \in \mathscr{S}, k \in \mathscr{K} \tag{2.3g}$$

Constraint 2.3c is a big-M formulation and ensures that $d_s^* = d_s^k$ if $b_s^k = 1$, and Constraint 2.3e guarantees that sample x_s belongs to only one cluster.

Similarly, k-center clustering has a mixed integer nonlinear programming formulation as follows:

$$\min_{\mu,d,b,\lambda} d_* \tag{2.4a}$$

$$\text{s.t. } d_s^k \geq ||x_s - \mu^k||_2^2 \tag{2.4b}$$

$$-N_1(1-b_s^k) \leq d_s^* - d_s^k \leq 0 \tag{2.4c}$$

$$d_* \geq d_s^* \tag{2.4d}$$

$$\sum_{k\in\mathscr{K}} b_s^k = 1 \tag{2.4e}$$

$$b_s^k \in \{0,1\} \tag{2.4f}$$

$$-N_2(1-\lambda_s^k) \leq x_s - \mu^k \leq N_2(1-\lambda_s^k) \tag{2.4g}$$

$$\sum_{s\in\mathscr{S}} \lambda_s^k = 1 \tag{2.4h}$$

$$\lambda_s^k \in \{0,1\} \tag{2.4i}$$

$$b_s^k \geq \lambda_s^k \tag{2.4j}$$

$$s \in \mathscr{S}, k \in \mathscr{K} \tag{2.4k}$$

Constraint 2.4c is a big-M formulation and ensures that $d_s^* = d_s^k$ if $b_s^k = 1$ and

$d_s^* \leq d_s^k$ otherwise, and Constraint 2.4e guarantees that sample x_s belongs to only one cluster. We also adapt Constraint 2.4g, 2.4h and 2.4j to represent $\mu \in X$, the restriction that centers are selected on data samples for each cluster. Specifically, Constraint 2.4g uses a big-M formula to make sure that $\mu^k = x_s$ if $\lambda_s^k = 1$, and Constraint 2.4h confirms that each center can only be selected on one sample. Constraint 2.4j ensures that if x_s is the center of the kth cluster, then obviously it is assigned to the kth cluster.

There are two differences between k-means and k-center clustering. One is that they have various objectives. The other is that k-center problem has the "centers on samples" constraint, while k-means clustering does not have such constraint.

k-means and k-center clustering can be respectively presented as a two-stage problem with the same second-stage optimal problem:

$$Q_s(\mu) = \min_{k \in \mathscr{K}} ||x_s - \mu^k||_2^2 \tag{2.5}$$

With different objectives, the first-stage optimization of Problem 2.3 has the form :

$$z^{km} = \min_{\mu \in M_0} \sum_{s \in \mathscr{S}} Q_s(\mu). \tag{2.6}$$

and Problem 2.4 has the first-stage problem of the form:

$$z^{kc} = \min_{\mu \in M_0 \cap X} \max_{s \in \mathscr{S}} Q_s(\mu). \tag{2.7}$$

where μ are the so-called first-stage variables.

Since $Q_s(\mu)$ is the minimum of a finite number of continuous functions, we have:

Lemma 2.1.1. *$Q_s(\mu)$ is continuous on $\mu \in M_0$ for all $s \in \mathscr{S}$.*

Because of the compactness of M_0 and Lemma 2.1.1, the clustering Problem 2.6 can attain its minimum according to the generalized Weierstrass theorem.

For the k-center problem, it is obvious that we can attain the value of z^{kc} in Problem 2.7 since $\mu \in M_0 \cap X$, which is a finite set.

2.2 Reduced-space Branch and Bound Scheme

We tailor the reduced-space branch and bound scheme to solve Problem 2.6 and 2.7. The details of the algorithm are given in Algorithm 1.

Algorithm 1 Branch and Bound Scheme

Initialization
Initialize the iteration index $i \leftarrow 0$;
Set $\mathbb{M} \leftarrow \{M_0\}$, and tolerance $\varepsilon > 0$;
Compute initial upper and lower bounds $\alpha_i = \alpha(M_0)$, $\beta_i = \beta(M_0)$;
while $\mathbb{M} \neq \emptyset$ **do**
  Node Selection
  Select a set M satisfying $\beta(M) = \beta_i$ from $\mathbb{M}$ and delete it from $\mathbb{M}$;
  Update $i \leftarrow i + 1$;
  Branching
  Find two subsets M_1 and M_2 s.t. $\mathring{M}_1 \cap \mathring{M}_2 = \emptyset$ and $M_1 \cup M_2 = M$;
  if Solving k-means clustering **then**
    Update $\mathbb{M} \leftarrow \mathbb{M} \cup \{M_1\} \cup \{M_2\}$
  else if Solving k-center clustering **then**
    Update $\mathbb{M} \leftarrow \mathbb{M} \cup \{M_1\}$, if $M_1^k \cap X \neq \emptyset, \forall k \in \mathscr{K}$
    Update $\mathbb{M} \leftarrow \mathbb{M} \cup \{M_2\}$, if $M_2^k \cap X \neq \emptyset, \forall k \in \mathscr{K}$
  end if
  Bounding
  Compute upper and lower bound $\alpha(M_1)$, $\beta(M_1)$, $\alpha(M_2)$, $\beta(M_2)$;
  Let $\beta_i \leftarrow \min\{\beta(M') \mid M' \in \mathbb{M}\}$;
  $\alpha_i \leftarrow \min\{\alpha_{i-1}, \alpha(M_1), \alpha(M_2)\}$;
  Remove all M' from $\mathbb{M}$ if $\beta(M') \geq \alpha_i$;
  If $|\beta_i - \alpha_i| \leq \varepsilon$, STOP;
end while

Chapter 3

k-Means Clustering

Based on different objectives and constraints, k-means and k-center problems have various approaches to obtaining lower bounds and upper bounds used in Algorithm 1. This chapter only concentrates on the k-means problem and introduces several methods to generate lower bounds and upper bounds. We also evaluate the performance of our branch and bound (BB) scheme on k-means clustering.

Using the two-stage formulation, when solving the Problem 2.6 with BB algorithm, we solve the following problem at each node with respect to the partition set $M \subseteq M_0$:

$$z^{km}(M) = \min_{\mu \in M} \sum_{s \in \mathscr{S}} Q_s(\mu) \tag{3.1}$$

The Problem 3.1 is referred as the *primal node problem*. By replicating the center μ for each sample, and enforcing the non-anticipativity constraints 3.2b, we give the lifted Problem 3.2 as follow:

$$\min_{\mu_s \in M} \quad \sum_{s \in \mathscr{S}} Q_s(\mu_s) \tag{3.2a}$$

$$\text{s.t.} \quad \mu_s = \mu_{s+1}, s \in \{1, \cdots, S-1\} \tag{3.2b}$$

Formulations 3.1, 3.2 are equivalent.

3.1 Lower Bounds

In this section, we describe methods to generate lower bounds for the primal node Problem 3.1 and improvements to generate tighter bounds.

3.1.1 Basic Lower Bounding Problem

By relaxing the non-anticipativity constraints 3.2b, we can obtain the following lower bounding problem:

$$\beta(M) := \min_{\mu_s \in M} \sum_{s \in \mathscr{S}} Q_s(\mu_s) \tag{3.3}$$

This problem can be easily decomposed into S sub-problems:

$$\beta_s(M) = \min_{\mu \in M} Q_s(\mu). \tag{3.4}$$

Here, $\beta(M) = \sum_{s \in \mathscr{S}} \beta_s(M)$. Since the non-anticipativity constraints are relaxed, the feasible region of 3.3 is a superset of the feasible region of Problem 3.1. Thus $\beta(M) \leq z^{km}(M)$ is always satisfied.

Because the closed-form solution to the second-stage problem is $Q_s(\mu) = \min_k ||x_s - \mu^k||_2^2$, we have:

$$\beta_s(M) = \min_k \min_{\mu^k \in M^k} ||x_s - \mu^k||_2^2, \tag{3.5}$$

Therefore, subproblem $\beta_s(M)$ can be further decomposed into K subsubproblems:

$$\beta_s^k(M^k) = \min_{\mu^k \in M^k} ||x_s - \mu^k||_2^2. \tag{3.6}$$

Here $\beta_s(M) = \min_k \beta_s^k(M^k)$. Problem 3.6 is a Quadratic Programming (QP) problem and the closed-form solution to this problem can be derived: $\mu_a^k = \text{mid}\{\underline{\mu}_a^k, x_{s,a}, \bar{\mu}_a^k\}$, $\forall a \in \{1, \cdots, A\}$.

3.1.2 Adaptive Sample Grouping

Although the lower bound generated using the basic lower bound problems is enough to guarantee convergence as shown in Section 5.1, it might not be very tight because decomposition is performed on each individual sample. Specifically, for the root node with $M = M_0$, since each subproblem only contains one data sample x_s, it is easy to verify that $\mu^k = x_s$ is the global optimal solution to Problem 3.6 and the corresponding $\beta(M_0)$ equal to 0. Although this lower bound can be improved as M is partitioned into smaller sets during the BB scheme, we seek to generate tighter bounds for each node.

One approach to achieve this goal is sample grouping, that is to assign more than one sample (i.e. a group of samples) to each subproblem. Specifically, we divide the set of samples $\mathscr{S}$ into G groups $\{\mathscr{S}_1, \mathscr{S}_2, \cdots, \mathscr{S}_G\}$ and define the group set $\mathscr{G} = \{1, \cdots, G\}$, where $\mathscr{S}_g$ represents the subset of $\mathscr{S}$ containing samples belonging to group g, such that $\bigcup_{g=1}^{G} \mathscr{S}_g = \mathscr{S}$ and $\mathscr{S}_i \cap \mathscr{S}_g = \emptyset, \forall i, g \in \mathscr{G}, i \neq g$. Considering the lifted problem, instead of replicating the center for each sample, we can replicate it for each group and reformulate Problem 3.2 as follow:

$$\min_{\mu_g \in M} \quad \sum_{g \in \mathscr{G}} Q_g(\mu_g) \tag{3.7a}$$

$$\text{s.t.} \quad \mu_g = \mu_{g+1}, g \in \{1, \cdots, G-1\}. \tag{3.7b}$$

Here, $Q_g(\mu_g) = \sum_{s \in \mathscr{S}_g} Q_s(\mu_g)$. By relaxing the constraints 3.7b, we can obtain the following sample grouped lower bounding problem:

$$\beta^{SG}(M) := \min_{\mu_g \in M} \sum_{g \in G} Q_g(\mu_g). \tag{3.8}$$

This problem can be easily decomposed into G subproblems:

$$\beta_g^{SG}(M) := \min_{\mu_g \in M} Q_g(\mu_g), \tag{3.9}$$

or, equivalently,

$$
\begin{aligned}
\beta_g^{SG}(M) := \min_{\mu_g, d_{\mathscr{S}_g}, b_{\mathscr{S}_g}} & \sum_{s \in \mathscr{S}_g} d_s^* \\
\text{s.t.} \quad & -N(1-b_s^k) \le d_s^* - d_s^k \le N(1-b_s^k) \\
& d_s^k \ge ||x_s - \mu^k||_2^2 \\
& \sum_{k \in \mathscr{K}} b_s^k = 1 \\
& b_s^k \in \{0,1\} \\
& s \in \mathscr{S}_g, k \in \mathscr{K}
\end{aligned}
\tag{3.10}
$$

with $\beta^{SG}(M) = \sum_{g \in \mathscr{G}} \beta_g^{SG}(M)$.

Compared with the basic lower bounding problems 3.3, nonanticipativity constraints within each group are re-enforced in Problem 3.8 (i.e. all samples within the same group share the same copy of centers), while nonanticipativity constraints between groups are still relaxed. Therefore, sample grouping based decomposition leads to a tighter relaxation and provides a stronger lower bound than the decomposition based on each individual sample. Therefore, we have:

Proposition 3.1.1. $\beta(M) \le \beta^{SG}(M) \le z^{km}(M)$.

Although sample grouping provides a tighter relaxation, the computational cost also increases dramatically compared with the basic lower bounding problem. It is because each subproblem 3.10 is a MISOCP problem that needs to be solved to global optimality.

The way of assigning groups strongly influences the quality of lower bounds. However, finding the optimal groups is itself an NP-hard problem. Therefore we developed a heuristic to group samples. We first divide the samples into K_{SG} clusters (default of $K * G$) with the constraints $\mu \in M$, which is solved using local optimal solvers as shown in Section 3.2.2. Then samples within each cluster is evenly distributed to different groups. Since M is different at each node, groups are updated at every iteration.

3.1.3 Lagrangian Decomposition

Another approach to compute tighter lower bounds is through Lagrangian Decomposition, in which nonanticipativity constraints are not removed but dualized. They are multiplied by fixed Lagrange multipliers λ and added to the objective function. Here we discuss the Lagrangian Decomposition based on each individual sample, while it can also be combined with sample grouping. The relaxed problem based on Lagrangian Decomposition can be written in the following form:

$$\beta^{LD}(M,\lambda) := \min_{\mu \in M} \{ \sum_{s \in \mathscr{S}} Q_s(\mu_s) + \sum_{s=1}^{S-1} \lambda_s(\mu_s - \mu_{s+1}) \} \tag{3.11}$$

It is clear that the basic lower bounding problem 3.3 is a special case of problem 3.11 with $\lambda = 0$. Problem 3.11 can be decomposed into S sub-problems:

$$\beta_s^{LD}(M,\lambda) := \min_{\mu_s \in M} \{ Q_s(\mu_s) + (\lambda_s - \lambda_{s-1})\mu_s \} \tag{3.12}$$

with $\lambda_0 = \lambda_S = 0$ and $\beta^{LD}(M,\lambda) = \sum_{s \in \mathscr{S}} \beta_s^{LD}(M,\lambda)$.

It can be shown that the solution to Problem 3.11 provides a lower bound to the primal node problem 3.2 by noticing that the optimal solution to Problem 3.2 is also feasible with respect to Problem 3.11 with the same objective value. Note that the value of λ in Problem 3.11 is fixed and choosing a proper value of λ may produce a tighter lower bound. To achieve the tightest lower bound, we need to solve the Lagrangian dual problem:

$$\beta^{LD}(M) = \max_{\lambda} \beta^{LD}(M,\lambda). \tag{3.13}$$

Solving this dual problem is itself very challenging. The community usually approaches with heuristic methods, including the Sub-gradient Method [20], Volume Algorithm [6], and Progressive Hedging Algorithm [46]. In this thesis, we adapt the sub-gradient method for the update of λ.

Since the lower bound computed from basic lower bounding problem satisfying $\beta(M) = \beta^{LD}(M,0)$. Thus, we have the following proposition:

Proposition 3.1.2. $\beta(M) \leq \beta^{LD}(M) \leq z^{km}(M)$

Compared with the basic lower bound problem, although Lagrangian Decomposition has the potential to find a tighter bound if a close-to-optimal λ is computed from the Lagrangian dual problem, the computational cost of solving a MISOCP subproblem 3.12 is much higher than solving a QP subproblem 3.6, which has a closed-form solution.

3.2 Upper Bounds

This section describes two approaches to generate upper bounds for the primal node Problem 3.1. Both methods are included in our implementation.

3.2.1 Basic Upper Bounding Problem

An upper bound of the primal node Problem 3.1 can be obtained by fixing the first stage variable μ at a candidate solution $\hat{\mu} \in M$. We denote the solution of the upper bounding problem as follow:

$$\alpha(M) = \sum_{s \in \mathscr{S}} Q_s(\hat{\mu}). \tag{3.14}$$

Because the closed-form solution to $Q_s(\hat{\mu})$ is known, the computation of $\alpha(M)$ is trivial without the need to solve any optimization problem. Since $\hat{\mu}$ is an arbitrary feasible solution to the primal node Problem 3.1, it is obviously that $z^{km}(M) \leq \alpha(M), \forall \hat{\mu} \in M$. The choice of candidate solution is a critical decision. In our implementation, the algorithm will test $\hat{\mu}$ obtained from both lower bounding problems as discussed in Section 3.1 and local optimization as discussed in Section 3.2.2.

3.2.2 Local Optimal Solution

Another way of computing the upper bound is to solve the primal node Problem 3.1 to local optimality. One alternative formulation of the Problem 3.1, which is

equivalent to Problem 2.3 with the addition of $\mu \in M$, is of the following form:

$$\begin{aligned} \min_{\mu \in M, b} \quad & \sum_{s \in \mathscr{S}} \sum_{k \in \mathscr{K}} b_s^k ||x_s - \mu^k||_2^2 \\ & \sum_{k \in \mathscr{K}} b_s^k = 1 \\ & 0 \leq b_s^k \leq 1 \\ & s \in \mathscr{S}, k \in \mathscr{K} \end{aligned} \tag{3.15}$$

The reason why $b_s^k \in \{0,1\}$ can be replaced by $0 \leq b_s^k \leq 1$ is because, if μ^k is fixed, a sample x_s will always be assigned to the cluster with the nearest center. This Nonlinear Programming Problem (NLP) can be easily solved using off-the-shelf solvers like Ipopt [51] to local optimality. In our implementation, we run the local solver several trails using different initial values to get the best local optimal value as the node's upper bound.

3.3 Computational Experiments

In this section, we evaluate the performance of our algorithm on both synthetic and real-world datasets. Our algorithm is implemented in Julia 1.5.3. We compare the computational results (in serial and parallel) against those of the classic BB algorithm implemented in the state-of-the-art global optimizer CPLEX 20.1.0 [14], and classic k-means clustering algorithm implemented in Julia Package Clustering. The result of k-means clustering algorithm is generated by repeating with 100 trials using random initialization. The worst, average and best k-means objectives are recorded. Both our algorithm and CPLEX terminate with a time limit of 4 hours We also did experiments on our BB clustering algorithm (BB) with different convergence acceleration techniques. Here, closed-form solution (BB+CF), scenario-based sample grouping (BB+SG), and grouping based Lagrangian decomposition (BB+LD+SG) are applied for comparison. We also present a preliminary parallel implementation in which the subproblems in BB+LD+SG are assigned to multiple CPU cores. For sample grouping, we limited each group's size under $\min(162/d - k, 10 \times k)$ when decomposing the lower bounding problem. All serial experiments are executed on a 2.3GHz quad-core 10th-generation Intel Core i7

processor with 16G RAM. The parallel experiments are executed on the Niagara Cluster of Compute Canada. The complete code files for the k-means clustering problem can be found in https://github.com/kingsley1989/global_kmeans.

Algorithms are compared based on three criteria: upper bound (UB), optimality gap, and number of nodes being solved. UB represents the best found (feasible) solution reported at the termination of each algorithm. Optimality gap is measured by the difference between the best possible (lower bound or LB) and best known solutions (UB) and is calculated as $Gap = \frac{UB-LB}{LB}$. The optimality gap of provides a certificate that the best found solution is not worse than $\varepsilon\%$ from the optimal solution. Therefore, it serves as a benchmark index on whether searching for a better solution is still necessary (if the optimality gap is already very low, then the potential improvement is small). The optimality gap is a unique property of deterministic global optimization algorithms. Heuristic techniques such as k-means clustering algorithm do not have the capability to evaluate the quality of its solutions (in the deterministic sense). The number of nodes records how many iterations the BB procedure processed within specific time period.

3.3.1 Numerical Results of Synthetic Data

We first consider numerical performance on artificially generated datasets. To illustrate the scalability of the algorithms, we consider datasets with different numbers of samples. All datasets are generated with 3 Gaussian clusters randomly with seed = 1. Each data sample has two attributes. For each dataset, we solve two clustering problems ($k = 3$ and $k = 4$). The number of variables involved in the optimization problem increases linearly with the number of samples and clusters.

Table 3.1 compares the performance of our BB-based clustering algorithms (BB), CPLEX, and k-means algorithm. In terms of the best found (feasible) solution or UB, the BBClst based algorithms can always obtain the lowest sum-of-squares cost. For example, for dataset *Syn-2100* on problem $k = 3$, our algorithm can reduce the optimal cost by 47.13% compared with the best solution returned by CPLEX. Remarkably, the best k-means objectives out of 100 trials also obtain the lowest cost. This process of repeatedly using local optimal algorithms with multiple random initialization can be viewed as a stochastic global optimization strategy. However,

Table 3.1: Computational results of different algorithms on synthetic datasets.

DATASET	METHOD	$k=3$			$k=4$		
		UB	NODES	GAP(%)	UB	NODES	GAP(%)
SYN-300	k-MEANS (WORST)	6454.84	-	-	6379.28	-	-
	k-MEANS (AVERAGE)	4506.19	-	-	3416.73	-	-
	k-MEANS (BEST)	**4049.28**	-	-	**3170.97**	-	-
	CLASSIC BB (CPLEX)	6945.02	8955000	1046.8	6782.52	8181700	3981.6
	BB+CF	4049.28	397442	7.31	3170.97	360753	35.67
	BB+SG	4049.28	4475	0.88	3170.97	655	6.02
	BB+LD+SG	**4049.28**	**73**	$\leq$**0.1**$(2h)$	**3170.97**	**41**	**2.67**
	BB+LD+SG(20CORES)	**4049.28**	**73**	$\leq$**0.1**$(0.4h)$	**3170.97**	**481**	**0.29**
SYN-1200	k-MEANS (WORST)	24465.51	-	-	23796.32	-	-
	k-MEANS (AVERAGE)	16223.76	-	-	12578.29	-	-
	k-MEANS (BEST)	**14290.53**	-	-	**11812.94**	-	-
	CLASSIC BB (CPLEX)	26417.87	2708500	27677.8	26154.47	1649002	58723.5
	BB+CF	14290.53	352489	7.42	11812.94	353012	44.73
	BB+SG	14290.53	714	2.20	**11812.94**	**169**	**12.26**
	BB+LD+SG	**14290.53**	**32**	**1.21**	11812.94	7	20.81
	BB+LD+SG(20CORES)	**14290.53**	**159**	**0.1**$(0.6h)$	**11812.94**	**177**	**2.79**
SYN-2100	k-MEANS (WORST)	43141.42	-	-	41943.30	-	-
	k-MEANS (AVERAGE)	28111.86	-	-	21639.97	-	-
	k-MEANS (BEST)	**25033.48**	-	-	**20598.32**	-	-
	CLASSIC BB (CPLEX)	47349.02	930610	23709.5	47431.11	603200	333233.3
	BB+CF	25033.48	333910	7.26	20598.32	335763	41.61
	BB+SG	25033.48	233	3.20	**20598.32**	**107**	**12.41**
	BB+LD+SG	**25033.48**	**14**	**2.71**	20598.32	3	25.14
	BB+LD+SG(20CORES)	**25033.48**	**322**	**0.26**	**20598.32**	**124**	**3.49**
SYN-42000	k-MEANS (WORST)	847326.29	-	-	822601.88	-	-
	k-MEANS (AVERAGE)	570643.51	-	-	437109.88	-	-
	k-MEANS (BEST)	**501472.82**	-	-	**411815.14**	-	-
	CLASSIC BB (CPLEX)[1]	-	-	-	-	-	-
	BB+CF	501472.82	148583	9.82	411815.14	139891	75.21
	BB+SG	**501472.82**	**4**	**5.43**	411815.14	3	23.46
	BB+LD+SG	501472.82	1	7.45	**411815.14**	**1**	**21.70**
	BB+LD+SG(20CORES)	**501472.82**	**20**	**3.16**	**411815.14**	**4**	**12.99**

[1] NO FEASIBLE SOLUTION.

one key limitation of this strategy is that it cannot compute the optimality gap and evaluate if it is necessary to continue the search for better solutions. In contrast, the optimality gap provided by deterministic global optimization algorithms gives a certificate on the solution quality.

In terms of the optimality gap, CPLEX cannot converge into a comparatively low value within a budget of 4 hours, and the optimality gap remains large (over 10000%) at the end of the solution process and in extreme cases, can not find a feasible solution for problems on large dataset (e.g. *Syn-42000*) within 4 hours. On the contrary, our algorithm can always maintain a comparatively low optimality gap after four-hour execution. Specifically, for the problem with $k = 3$ on dataset *Syn-300* which has 300 samples, our algorithm ends with an optimality gap lower

than 0.1% in within 2 hours in serial and within 0.4 hours in parallel (20 cores). Even when we increase the number of samples to 42000, the optimality gap remains at 5.43% in serial and 3.16% in parallel (20 cores), under the same runtime.

Comparing our algorithms using different methods to generate lower bounds, we find that, scenario-based sample grouping may be beneficial for some k-means problem, while the Lagrangian decomposition component may accelerate or slow down the solution process depending on the number of variables and clusters.

The decomposition scheme of our algorithm enables an easy way for paralleling. To further illustrate the scalablity of our algorithm on the k-means problem. We tested BB+LD+SG on the synthetic dataset with 210,000 samples in parallel of 200 cores. The results are listed in Table 3.2. Here, we can see that even for dataset over 200,000 samples, which is 10 times larger the state of the art (2392 samples) by [4],

Table 3.2: Computational results of large synthetic dataset in parallel. (BB+LD+SG, 200 cores, $k = 3$)

DATASET	UB	NODES	GAP(%)
SYN-210,000	2.43×10^6	6	2.55

3.3.2 Numerical Results of Real-world Data

We then perform numerical experiments on several real-world datasets. First, we pick three datasets from UCI Machine Learning Repository [17] and [44] with different scales to demonstrate the performance and generalization ability of our algorithm. Two of them are well known benchmark instances. The *Iris* dataset has 150 samples and 4 attributes, while the *Seeds* dataset contains 210 samples and 7 attributes. The third dataset, *Hemi* dataset, contains 1955 experimental data samples. Each of its data sample represents a reaction condition with 7 attributes.

Table 3.3 summarizes the performance of different algorithms on these three datasets. In terms of the UB, our algorithms and k-means (best) attains the lowest cost for all datasets. In terms of the optimality gap, our algorithm reports a significantly lower optimality gap compared to the Classic BB method. Remarkably, our implementation BB+LD+SG can converge to 0.1% of the optimality gap within

1.5 hour execution for the *Iris* dataset in serial. For parallel execution with 20 cores, only 0.6 hour and 1 hour executions are needed for *Iris* and *Seeds* dataset to converge to 0.1%, respectively. In contrast, CPLEX halts at 328.63%.

Table 3.3: Computational results of different algorithms on real-world datasets. ($k = 3$)

DATASET	SAMPLE	DIMENSION	METHOD	UB	NODES	GAP(%)
IRIS	150	4	k-MEANS (WORST)	145.45	-	-
			k-MEANS (AVERAGE)	85.91	-	-
			k-MEANS (BEST)	**78.85**	-	-
			CLASSIC BB (CPLEX)	82.64	12313100	328.63
			BB+CF	78.85	382280	50.76
			BB+SG	78.85	876	1.58
			BB+LD+SG	**78.85**	**31**	**0.1**(1.5h)
			BB+LD+SG(20CORES)	**78.85**	**31**	**0.1**(0.6h)
SEEDS	210	7	k-MEANS (WORST)	916.21	-	-
			k-MEANS (AVERAGE)	591.46	-	-
			k-MEANS (BEST)	**587.32**	-	-
			CLASSIC BB (CPLEX)	626.37	6715463	850.57
			BB+CF	587.32	356273	69.49
			BB+SG	587.32	433	4.34
			BB+LD+SG	**587.32**	**47**	**0.26**
			BB+LD+SG(20CORES)	**587.32**	**89**	**0.1**(1h)
HEMI	1,955	7	k-MEANS (WORST)	16.98×10^6	-	-
			k-MEANS (AVERAGE)	10.20×10^6	-	-
			k-MEANS (BEST)	$\mathbf{9.75\times10^6}$	-	-
			CLASSIC BB (CPLEX)	18.38×10^6	478800	4950.51
			BB+CF	9.75×10^6	326896	59.48
			BB+SG	$\mathbf{9.75\times10^6}$	**74**	**21.75**
			BB+LD+SG	9.75×10^6	4	39.38
			BB+LD+SG(20CORES)	$\mathbf{9.75\times10^6}$	**112**	**2.23**

The rest two datasets are selected from [4] to compare the performance with current state of the art. The Padberg and Rinald's hole drilling dataset [44] who has 2392 samples and 2 attributes is the dataset with maximum sample size in [4]. The glass identification dataset [17] is also a well-know benchmark for clustering problem. It has 214 samples and 9 attributes. Table 3.4 expounds the results of each dataset for problem $k = 2$.

Specifically, Aloise's algorithm is superior when k is large or $n/k \approx 10$ [4]. In contrast, we offer a different approach that can process large datasets with relatively

small number of clusters. As shown in Table 3.4, for problem $k = 2$, Aloise's algorithm cannot solve the glass identification data and is very slow for Padberg and Rinald's data. Yet, our BB clustering algorithm can even hit the gap lower than 0.1% within 12 hours on both datasets.

Table 3.4: Comparison on datasets with Aloise et al.[4]. (BB+LD+SG, $k = 2$)

METHOD	UB	NODES	GAP(%)
Padberg and Rinald's Dataset ($n = 2,392, d = 2$)			
ALOISE ET AL.	2.967×10^{10}	1	i[1](50h)
SERIAL	2.967×10^{10}	7	1.32 (4h)
SERIAL	2.967×10^{10}	253	0.1 (11h)
20 CORES	2.967×10^{10}	247	0.1 (1h)
Glass Identification ($n = 214, d = 9$)			
ALOISE ET AL. [2]	-	-	-
SERIAL	819.63	85	28.65 (4h)
SERIAL	819.63	339	0.1 (9h)
20 CORES	819.63	415	0.1 (1h)

[1] SOLVED AT THE ROOT NODE.
[2] CAN NOT BE SOLVED.

The numerical experiments have shown our algorithm's ability to handle datasets with up to thousands of data samples. Numerical results also illustrate that the algorithm can provide the deterministic clustering results with good generations on diverse real-world datasets.

Chapter 4

k-Center Clustering

In this chapter, we adapt the BB Algorithm 1 on the k-center problem. Section 4.1 and 4.2 propose approaches to compute lower bounds and upper bounds. Section 4.3 designs several bounds tightening techniques to significantly reduce the search space of the BB algorithm and accelerate the solution process. Moreover, we provide an open-source Julia implementation of the algorithm and performed extensive computational experiments on 29 UCI datasets for the k-center problem in Section 4.4.

4.1 Lower Bounds

As mentioned in Section 2.1, when using two-stage formulations, k-means and k-center clustering have the same second-stage optimal Problem 2.5. We can adapt similar method discussed in Section 3.1.1 to generate lower bounds for the k-center problem.

At each node in a BB algorithm, we deal with a subset of M_0, which is denoted as M, and solve the following problem with respect to M:

$$z^{kc}(M) = \min_{\mu \in M \cap X} \max_{s \in \mathscr{S}} Q_s(\mu) \tag{4.1}$$

This problem is referred as the primal node problem. It can be equivalently reformulated as the following problem by duplicating μ across samples and enforcing

them to be equal. This gives the lifted form:

$$\min_{\mu_s \in M \cap X} \max_{s \in \mathscr{S}} Q_s(\mu_s) \tag{4.2a}$$

$$\text{s.t.} \quad \mu_s = \mu_{s+1}, s \in \{1, \cdots, S-1\} \tag{4.2b}$$

By removing the "centers on samples" constraint $\mu \in M$ and the Constraints 4.2b, we attain a lower bounding formulation as follow:

$$\beta(M) := \min_{\mu_s \in M} \max_{s \in \mathscr{S}} Q_s(\mu_s). \tag{4.3}$$

With constraints relaxed, the feasible region of Problem 4.3 is a superset of the primal feasible region. Therefore, it is obvious that $\beta(M) \leq z^{kc}(M)$.

In Problem 4.3, since each sample is independent, it is obvious that:

$$\beta(M) = \max_{s \in \mathscr{S}} \min_{\mu_s \in M} Q_s(\mu_s). \tag{4.4}$$

Clearly, it can be decomposed into S subproblems of the form:

$$\beta_s(M) = \min_{\mu \in M} Q_s(\mu), \tag{4.5}$$

with $\beta(M) = \max_{s \in \mathscr{S}} \beta_s(M)$. Denote $M^k := \{\mu^k : \underline{\mu}^k \leq \mu^k \leq \bar{\mu}^k\}$ where $\underline{\mu}^k$ and $\bar{\mu}^k$ are the lower and upper bound of μ^k respectively. Since the closed-form solution to $Q_s(\mu)$ is $Q_s(\mu) = \min_{k \in \mathscr{K}} ||x_s - \mu^k||_2^2$, we have

$$\beta_s(M) = \min_{k} \min_{\mu^k \in M^k} ||x_s - \mu^k||_2^2, \tag{4.6}$$

which can be further decomposed into K subsubproblems:

$$\beta_s^k(M^k) = \min_{\mu^k \in M^k} ||x_s - \mu^k||_2^2. \tag{4.7}$$

with $\beta_s(M) = \min_k \beta_s^k(M^k)$. The analytical solution to Problem 4.7 can be derived as: $\mu_a^k = \text{mid}\{\underline{\mu}_a^k,\ x_{s,a},\ \bar{\mu}_a^k\}, \forall a \in \{1, \cdots, A\}$, which is the same as Problem 3.6.

4.2 Upper Bounds

Section 3.2.1 claims that an upper bound of the primal node can be obtained by fixing the centers at a candidate feasible solution $\hat{\mu}$, which can also be utilized for the k-center problem. In this way, we can compute the upper bound base on the following equation:

$$\alpha(M) = \max_{s \in \mathscr{S}} Q_s(\hat{\mu}). \tag{4.8}$$

where $\hat{\mu} \in M \cap X$. It is easy to see that $z^{kc}(M) \leq \alpha(M)$, $\forall \hat{\mu} \in M \cap X$. Since the closed-form solution to $Q_s(\hat{\mu})$ is known, the computation of the upper bound is cheap, with no need to solve any optimization problems, for a given candidate solution.

In our implementation, we use two methods to obtain candidate solutions. At the root node, we use a heuristic method called Farthest First Traversal to obtain a candidate solution $\hat{\mu} \in M_0 \cap X$. Using this method, we first randomly pick an initial point and then select each following point to be as far as possible from the set of previously selected points. Algorithm 2 describes the details of farthest first traversal. We use $FFT(M_0)$ to denote the upper bound obtained using this approach. At a child node with center region M, for each cluster, we select the data sample which is closest to the middle point of M^k as $\hat{\mu}^k$, and obtain the corresponding upper bound $\alpha(M)$.

Algorithm 2 Farthest First Traversal

Initialization
Randomly pick $s \in S$;
Denote T as the set of K points selected by farthest first traversal;
Set $T \leftarrow \{x_s\}$;
while $|T| < K$ **do**
  Compute $x_s \in \arg\max_{x_s \in X} d(x_s, T)$ to find x_s which is the farthest away from set T;
  $T \leftarrow T \cup \{x_s\}$;
end while

4.3 Bounds Tightening Techniques

Although the lower bound introduced in Section 4.1 is enough to guarantee convergence, it might not be very tight, leading to a tremendous amount of iterations.

Therefore here we propose bounds tightening techniques to reduce the search space and speed up the BB procedure. Since Algorithm 1 only branches on the space of centers μ, we focus on reducing the region of μ to accelerate the solution process, while guaranteeing the optimal solution of the problem is not excluded.

4.3.1 Cluster Assignment

In this subsection, we propose several strategies to decide the assignment of some samples (e.g. which cluster the sample is assigned to), that is to determine the value of b_s^k in Problem 2.4 for some s and k, before finding the optimal solution of the whole problem. This information will be used in the next subsection to reduce the region of μ.

Denote α as the current best upper bound of the optimal value achieved using methods described in Section 4.2. Then from Objective 2.4a and Constraint 2.4c in Formulation 2.4, we have $d_s^* \leq \alpha$. Based on Constraint 2.4b and 2.4c, we can conclude that if $b_s^k = 1$, then $||x_s - \mu^k||_2^2 \leq \alpha$. Therefore, we have Lemma 4.3.1.

Lemma 4.3.1. *If sample x_s is in the kth cluster, then $||x_s - \mu^k||_2^2 \leq \alpha$, where α is an upper bound of the k-center problem.*

Lemma 4.3.1 tells us that if $||x_s - \mu^k||_2^2 > \alpha$, then we can infer $b_s^k = 0$, that is sample x_s cannot be assigned to the kth cluster. Besides inferring from the distance between samples and centers, cluster assignments may also be determined from the distance of two data samples, as shown in the following Lemma 4.3.2.

Lemma 4.3.2. *If two samples x_i and x_j are in the same cluster, then $||x_i - x_j||_2^2 \leq 4\alpha$ where α is an upper bound of the k-center problem.*

Lemma 4.3.2 is obvious by noticing that if x_i and x_j all belong to the kth cluster, then based on Lemma 4.3.1 we have $||x_i - \mu^k||_2^2 \leq \alpha$ and $||x_j - \mu^k||_2^2 \leq \alpha$. Thus $||x_i - x_j||_2^2 = ||x_i - \mu^k + \mu^k - x_j||_2^2 \leq (||x_i - \mu^k||_2 + ||\mu^k - x_j||_2)^2 \leq 4\alpha$. Lemma 4.3.2 tells us that if $||x_i - x_j||_2^2 > 4\alpha$, then x_i and x_j are not in the same cluster.

Based on these two Lemmas, the followings are three approaches to assign samples to clusters.

K Farthest Points

By Lemma 4.3.2, if there are K points and the distance between any two of these points x_i and x_j satisfying $||x_i - x_j||_2^2 > 4\alpha$, then we can conclude that each point belongs to a distinct cluster. If we can find the K points satisfying this property at the root node, we can arbitrarily assign these points to different clusters. In other words, we can denote the cluster containing the kth point as kth cluster. We call these K points as initial seeds. In order to find these initial seeds, every two samples must be as far as possible. Therefore, in our implementation, we use the heuristic Farthest First Traversal (FFT) (Algorithm 2) to obtain K farthest points. For about half of the case studies shown in Section 4.4, we can obtain the initial seeds using FFT. However, for other datasets, initial seeds can not be obtained using FFT, or maybe the initial seeds do not even exist.

Center-Based Assignment

By Lemma 4.3.1, if $||x_s - \mu^k||_2^2 > \alpha$, then we can conclude that x_s is not in cluster k, or $b_s^k = 0$. If we can determine that $b_s^k = 0, \forall k \in \mathscr{K} \setminus \{k'\}$, then $b_s^{k'} = 1$. However, the value of μ here is not known before solving the overall problem. One observation is that if the node M contains the optimal solution, then we have $\beta_s^k(M^k) = \min_{\mu^k \in M^k} ||x_s - \mu^k||_2^2 \leq ||x_s - \mu^k||_2^2$. Therefore, if $\beta_s^k(M^k) > \alpha$, then by Lemma 4.3.1, sample x_s is definitely not in the kth cluster and $b_s^k = 0$. In summary, for sample x_s, if $\forall k \in \mathscr{K} \setminus \{k'\}$, $\beta_s^k(M^k) > \alpha$, then x_s is assigned to cluster k' with $b_s^{k'} = 1$. Figure 4.1 illustrates an example in two-dimensional space with a total of three clusters.

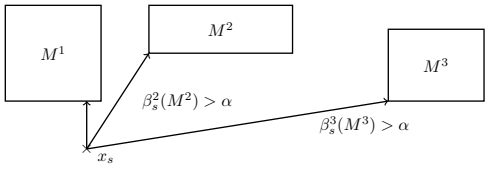

Figure 4.1: Center-based assignment with 3 clusters. In this example, $\beta_s^2(M^2) > \alpha$ ($b_s^2 = 0$) and $\beta_s^3(M^3) > \alpha$ ($b_s^3 = 0$). Therefore, we assign x_s to the first cluster ($b_s^1 = 1$).

This center-based method can be adapted at every node of the BB scheme. Since

$\beta_s^k(M^k)$ is already computed when obtaining the lower bound, there is no additional cost of distance computation or solving any optimization problem. Nevertheless, we do not need to apply this method at the root node, since $M^1 = \cdots = M^K$ initially. As the BB scheme continues branching on μ, M^k becomes more and more different from those of other clusters, then the cluster of more samples can be determined.

Sample-based Assignment

Besides using centers as a benchmark to allocate data points, assigned samples also give us the assignment of undetermined samples. By Lemma 4.3.2, if $||x_i - x_j||_2^2 > 4\alpha$, then x_i and x_j are not in the same cluster. If we already know that x_j belongs to cluster k, then obviously x_i cannot be assigned to cluster k, or $b_i^k = 0$. Using this relation, if all the other $K-1$ clusters are excluded, x_i will be assigned to the only one cluster left. An example of the sample-based assignment is depicted in Figure 4.2.

There is a prerequisite to using this method. For each cluster, there must be at least one sample that has already been assigned to the cluster so that we are able to compare the distance with the upper bound. Based on this condition, sample-based assignment is utilized only after the algorithm has already determined at least one sample for each cluster.

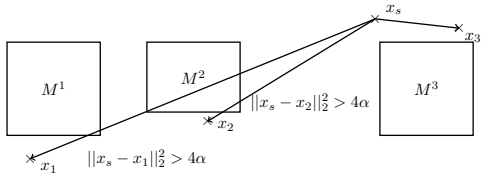

Figure 4.2: Sample-based assignment with 3 clusters. Assume we have already known that x_1, x_2, x_3 belong to cluster $1, 2$ and 3, respectively. x_s is the sample to be determined. In this example, $||x_s - x_1||_2^2 > 4\alpha$ ($b_s^1 = 0$) and $||x_s - x_2||_2^2 > 4\alpha$ ($b_s^2 = 0$). Therefore, x_s is assigned to cluster 3 ($b_s^3 = 1$).

4.3.2 Feasibility-Based Bounds Tightening

In this subsection we adapt the Feasibility-Based Bounds Tightening (FBBT) technique to reduce the space of μ. For a sample j, denote $B_\alpha(x_j)=\{x|\ ||x-x_j||_2^2 \leq \alpha\}$ as the ball with center x_j and radius $\sqrt{\alpha}$. By using methods described in subsection 4.3.1, assume we already know that sample j belongs to cluster k, by Lemma 4.3.1, then $\mu^k \in B_\alpha(x_j)$. We use $\mathscr{J}_A^k$ to denote the index of all samples assigned to cluster k, i.e., $\mathscr{J}_A^k = \{j \in S \mid b_j^k = 1\}$, then $\mu^k \in B_\alpha(x_j), \forall j \in \mathscr{J}_A^k$.

Besides this, we also know that $\mu^k \in M^k \cap X$. Denote $\mathscr{S}_+^k$ as the index set of samples satisfy all these constraints, $S_+^k(M) := \{s \in S \mid x_s \in M^k, x_s \in B_\alpha(x_j), \forall j \in \mathscr{J}_A^k\}$. In this way, we can obtain a tightened box containing all feasible solutions of kth medoid, $\hat{M}^k=\{\mu^k|\underline{\hat{\mu}}^k \leq \mu^k \leq \hat{\bar{\mu}}^k\}$, with the bounds of ath attribute in kth medoid to be $\underline{\hat{\mu}}_a^k = \min\limits_{s\in S_+^k(M)} x_{s,a}^k$ and $\hat{\bar{\mu}}_s^k = \max\limits_{s\in S_+^k(M)} x_{s,a}^k$. Figure 4.3 gives an example of bounds tightening using this method. One challenge of this ball-based bounds tightening method is that it need to compute the distance of x_s and x_j for all $s \in S$ and $j \in \mathscr{J}_A^k$. If we know the assignments of most of the samples, we need to do at most S^2 times of distance calculation. Note that we only need to do $S*K$ times of distance calculation to compute a lower bound. To reduce the computational time, in our implementation, we set a threshold on the maximum number of balls utilized to tighten bounds.

Another strategy to reduce the computation burden is to utilize the relaxation of $B_\alpha(x_j)$. For any ball $B_\alpha(x_j)$, the closed set $R_\alpha(x_j) = \{x \mid x_j - \sqrt{\alpha} \leq x \leq x_j + \sqrt{\alpha}\}$ is the smallest box containing $B_\alpha(x_j)$. Then we have $\mu^k \in R_\alpha(x_j), \forall j \in \mathscr{J}_A^k$. Since $R_\alpha(x_j)$ and M^k are all boxes, we can easily compute the tighten bounds $\hat{M}^k = \cap_{j\in\mathscr{J}_A^k} R_\alpha(x_j) \cap M^k$. Figure 4.4 gives an example of box-based bounds tightening using this method. Obviously, the bounds generated in Figure 4.3 is much tighter while the method in Figure 4.4 is much faster. Consequently, if $|\mathscr{J}_A^k|$ is small for all clusters, the ball-based bounds tightening method gives more satisfactory results. While if $|\mathscr{J}_A^k|$ is large for any k, box-based bounds tightening provides a cheap alternative.

4.3.3 Symmetry Breaking

Another way to get tighter bounds is based on the symmetry breaking constraints. We add the condition $\mu_1^1 \leq \mu_1^2 \leq \cdots \leq \mu_1^K$ in the BB algorithm 1, in which μ_a^k

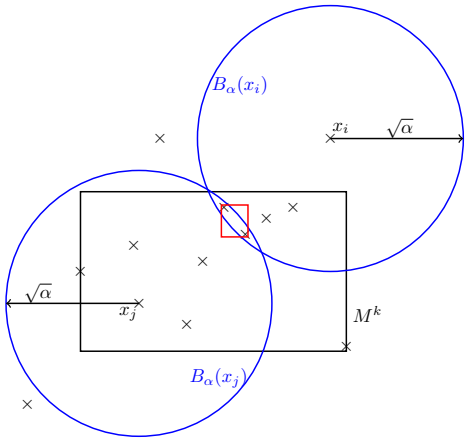

Figure 4.3: Ball-based bounds tightening in two-dimensional space. In this example, suppose it is determined that two points x_i and x_j belong to the kth cluster. We first compute the index set of samples within all balls and original box, $S_+^k(M) := \{s \in S \,|x_s \in M^k \cap B_\alpha(x_i) \cap B_\alpha(x_j)\}$. We then generate the smallest box containing these samples in $S_+^k(M)$. The red rectangle is the tighter bounds we obtained.

denotes ath attribute of kth center. This constraint can also help the algorithm to reduce the search space. For example with M^k=$\{\mu|\underline{\mu} \leq \mu \leq \bar{\mu}\}$ as the original box of μ, we can update $\underline{\mu}_1^K$ to be $max(\underline{\mu}_1^1, \underline{\mu}_1^2, \cdots, \underline{\mu}_1^K)$. Note that both symmetry breaking constraints and FFT-based inital seeds serve the function of breaking symmetry by providing a certain order for the clusters, so they cannot be combined together.

4.4 Computational Experiments

In this section, we evaluate the performance of our algorithms on the 29 datasets from UCI Machine Learning Repository [17] and [44]. The number of samples varies from 150 to 14,057,567. The attribute numbers of the datasets range from 2 to 68. The branch and bound scheme is tested with different methods to accelerate convergence, including closed-form solution (BB+CF), closed-form with symmetric breaking (BB+CF+SB), and closed-form with FBBT (BB+CF+FBBT). It should be noted that we also apply symmetric breaking in BB+CF+FBBT when initial seeds

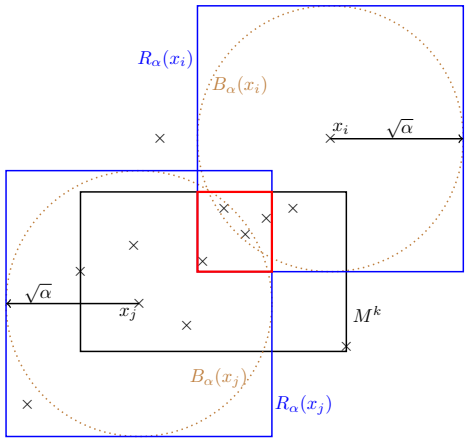

Figure 4.4: Box-based bounds tightening in two-dimensional space. In this example, we first generate two boxes with $R_\alpha(x_i) := \{x|\ x_i - \sqrt{\alpha} \leq x \leq x_i + \sqrt{\alpha}\}$ and $R_\alpha(x_j) = \{x|\ x_j - \sqrt{\alpha} \leq x \leq x_j + \sqrt{\alpha}\}$. We then create a tighten bounds with $\hat{M}^k$=$R_\alpha(x_i) \cap R_\alpha(x_j) \cap M^k$. The red rectangle is the tighter bounds we want.

are not found from FFT at the root node as discussed in Section 4.3.1.

We compare the numerical results of our algorithm with the state-of-art global optimizer CPLEX 20.1.0 [15] and the heuristic algorithm, Farthest First Traversal (FFT). Since the initial point is selected randomly in FFT, we repeat FFT 100 trails with different initialization to avoid this randomness and obtain the worst, best, and average results. We implement CPLEX and our algorithms in Julia 1.6.1 on the Niagara Compute Canada with a time limit of 4 hours. In all the experiments, the number of Cluster K is set to 3. The complete code files for the k-center clustering problem can be found in https://github.com/mingfei-shi/Kcenter.

The computational results are compared by three criteria, including upper bound (UB), optimality gap, the number of solved nodes in BB scheme. UB measures the best feasible solution. The optimality gap represents the relative difference between the best feasible lower bound (LB) and UB. It is defined as $Gap = \frac{UB-LB}{LB}$. The optimality gap is a unique property for the deterministic global optimization algorithm. The heuristic algorithm (FFT) does not have such a property, which means FFT can not evaluate the quality of its solutions. The number of solved nodes

is the iteration number of BB scheme before we find the best possible LB and UB.

4.4.1 Numerical Results of Small Scale Datasets

Table 4.3 is the computational performance of datasets with less than 1000 samples. In this table, even the best results of Farthest First Traversal can be far away from optimal. For example, in BM dataset, FFT obtains the best UB of 15245 while our algorithm and CPLEX give a UB of 10539 with zero gap. Our algorithms obtain the same UB as CPLEX, buy CPLEX takes significantly more computational time than our algorithms and cannot even close the optimality gap to 50% within 4 hours for GLASS, UK, ABS, TR, and SGC datasets.

As for the comparison of our algorithms, BB+CF, BB+CF+SB, and BB+CF+FBBT can all generate the best UB and a satisfactory gap ($\leq 0.2\%$) for all the datasets with $S \leq 1000$. Among them, BB+CF+FBBT needs the minimum nodes to solve the problems. Moreover, the number of nodes needed to reach the optimal solution is smaller when initial seeds are found, which shows that the initial seeds can be an ideal start of center-based and sample-based assignment.

For about half of the datasets, initial seeds can be obtained from FFT. Here we propose an index to check in advance whether we can find initial seeds by using FFT. The index is a ratio that measures the density of a dataset. It is computed as $Ratio = \frac{DIST}{UB}$, where $DIST$ is the minimum distance between K centers. Note here the values of K centers are from the best reported feasible solution. If the samples in the dataset are far from each other, this Ratio will be relatively large. The lower this ratio is, the denser the dataset is. We find that for the dataset with a ratio larger than 2, the initial seeds can be obtained. Regarding the Ratio, the greater the Ratio is, the more likely a dataset can obtain an initial assignment.

4.4.2 Numerical Results of Large Scale Datasets

Table 4.2 shows the results of datasets with more than 1000 samples. Like the results of small scale datasets, FFT fails to get the lowest UB in all the datasets. CPLEX can neither obtain a satisfactory optimality gap nor give a feasible solution within 4 hours. Compared with FFT and CPLEX, our algorithms can obtain the best upper bounds in all the datasets and reach a satisfactory gap ($\leq 0.1\%$) in most datasets

within the run-time of 4 hours. All three versions of our algorithms maintain a balanced memory usage during the whole execution process, while for datasets with size over 50,000, CPLEX occurs the out of memory error on Compute Canada.

In Table 4.1, we illustrate the numerical results of datasets with millions of samples. For all the datasets in Table 4.1, BB+CF+FBBT can converge to small gaps ($\leq 0.1\%$) and provide the best optimal solution after 4 hours of running. To our best knowledge, it is the first time that the k-center problem is solved under a relatively small gap ($\leq 0.1\%$) within 4 hours on datasets with over **14 million** samples.

Table 4.1: Computational results of datasets with millions of samples.

DATASET	SAMPLE	DIM-ENSION	METHOD	UB	NODES	GAP (%)	RATIO	TIME (S)
USC1990[1]	2,458,285	68	FFT (WORST)	6.68×10^{11}	-	-	-	-
			FFT (AVERAGE)	4.23×10^{11}	-	-	-	-
			FFT (BEST)	2.44×10^{11}	-	-	4.00	-
			CPLEX [3]	-	-	-	-	-
			BB+CF	1.69×10^{11}	916	3.75	-	14400
			BB+CF+SB	1.68×10^{11}	284	$\leq$0.1	-	4604
			BB+CF+FBBT	$\mathbf{1.68 \times 10^{11}}$	**i**[4]	**$\leq$0.1**	**4.00**	**317**
GAS_METHANE[1]	4,178,504	18	FFT (WORST)	2.84×10^{8}	-	-	-	-
			FFT (AVERAGE)	2.18×10^{8}	-	-	-	-
			FFT (BEST)	1.60×10^{8}	-	-	2.36	-
			CPLEX [3]	-	-	-	-	-
			BB+CF	1.04×10^{8}	1220	45.08	-	14400
			BB+CF+SB	1.03×10^{8}	1348	31.16	-	14400
			BB+CF+FBBT	$\mathbf{1.02 \times 10^{8}}$	**33**	**$\leq$0.1**	**1.78**	**679**
GAS_CO[2]	4,208,261	18	FFT (WORST)	1.71×10^{9}	-	-	-	-
			FFT (AVERAGE)	1.28×10^{9}	-	-	-	-
			FFT (BEST)	8.81×10^{8}	-	-	2.26	-
			CPLEX [3]	-	-	-	-	-
			BB+CF	5.66×10^{8}	1095	14.66	-	14400
			BB+CF+SB	5.46×10^{8}	909	$\leq$0.1	-	11877
			BB+CF+FBBT	$\mathbf{5.46 \times 10^{8}}$	**62**	**$\leq$0.1**	**1.72**	**878**
PHONES_ACCELEROMETER[1]	13,062,475	6	FFT (WORST)	5.09×10^{28}	-	-	-	-
			FFT (AVERAGE)	4.48×10^{28}	-	-	-	-
			FFT (BEST)	2.04×10^{28}	-	-	6.12	-
			CPLEX [3]	-	-	-	-	-
			BB+CF	1.46×10^{28}	51	$\leq$0.1	-	2038
			BB+CF+SB	1.46×10^{28}	51	$\leq$0.1	-	2252
			BB+CF+FBBT	$\mathbf{1.46 \times 10^{28}}$	**i**[4]	**$\leq$0.1**	**3.55**	**276**
PHONES_GYROSCOPE[1]	13,932,632	6	FFT (WORST)	5.09×10^{28}	-	-	-	-
			FFT (AVERAGE)	4.51×10^{28}	-	-	-	-
			FFT (BEST)	2.03×10^{28}	-	-	6.14	-
			CPLEX [3]	-	-	-	-	-
			BB+CF	1.46×10^{28}	51	$\leq$0.1	-	2195
			BB+CF+SB	1.46×10^{28}	51	$\leq$0.1	-	2214
			BB+CF+FBBT	$\mathbf{1.46 \times 10^{28}}$	**i**[4]	**$\leq$0.1**	**4.74**	**305**
AADP[2]	14,057,567	3	FFT (WORST)	4923.15	-	-	-	-
			FFT (AVERAGE)	3900.94	-	-	-	-
			FFT (BEST)	3824.00	-	-	1.72	-
			CPLEX [3]	-	-	-	-	-
			BB+CF	2660.10	602	55.91	-	14400
			BB+CF+SB	2660.10	619	49.32	-	14400
			BB+CF+FBBT	**2546.92**	**196**	**$\leq$0.1**	**0.50**	**4321**

[1] CAN ASSIGN INITIAL CLUSTER THROUGH FFT AT THE ROOT NODE.
[2] CAN NOT ASSIGN INITIAL CLUSTER THROUGH FFT AT THE ROOT NODE.
[3] OUT OF MEMORY WHEN USING NIAGARA COMPUTE CANADA.
[4] SOLVED AT THE ROOT NODE.

Table 4.2: Computational results of large-scale datasets. ($1,000 < S < 1,000,000$)

Data-set	Sam-ple	Dim-ension	Method	UB	Nodes	Gap (%)	Ratio	Time (s)
hemi[1]	1955	7	FFT (worst)	251438.77	-	-	-	-
			FFT (average)	149300.03	-	-	-	-
			FFT (best)	105657.63	-	-	4.04	-
			CPLEX	180123.59	-	87.66	-	14400
			BB+CF	64872.92	1275	≤0.1	-	18
			BB+CF+SB	**64872.92**	**223**	**≤0.1**	-	**11**
			BB+CF+FBBT	**64872.92**	**13**	**≤0.1**	**4.08**	**15**
pr2392[2]	2392	2	FFT (worst)	1.05×10^8	-	-	-	-
			FFT (average)	6.96×10^7	-	-	-	-
			FFT (best)	3.99×10^7	-	-	2.44	-
			CPLEX	5.55×10^7	-	100	-	14400
			BB+CF	2.93×10^7	59327	≤0.1	-	297
			BB+CF+SB	2.93×10^7	11631	≤0.1	-	67
			BB+CF+FBBT	$\mathbf{2.93 \times 10^7}$	**241**	**≤0.1**	**0.93**	**16**
TRR[2]	5454	24	FFT (worst)	161.83	-	-	-	-
			FFT (average)	136.19	-	-	-	-
			FFT (best)	101.55	-	-	1.06	-
			CPLEX	165.64	-	100	-	14400
			BB+CF	89.78	357283	271.06	-	14400
			BB+CF+SB	**88.29**	**358090**	**169.97**	-	**14400**
			BB+CF+FBBT	88.30	277504	178.99	0.51	14400
AC[2]	7195	22	FFT (worst)	5.88	-	-	-	-
			FFT (average)	4.27	-	-	-	-
			FFT (best)	3.59	-	-	1.17	-
			CPLEX[3]	-	-	-	-	-
			BB+CF	2.75	313564	74.30	-	14400
			BB+CF+SB	**2.78**	**317577**	**62.82**	-	**14400**
			BB+CF+FBBT	2.78	267412	63.65	0.60	14400
rds_cnt[1]	10000	4	FFT (worst)	53361.0	-	-	-	-
			FFT (average)	38772.90	-	-	-	-
			FFT (best)	20449.0	-	-	3.97	-
			CPLEX	80656.01	-	100	-	14400
			BB+CF	13924.00	639	≤0.1	-	25
			BB+CF+SB	13924.00	631	≤0.1	-	27
			BB+CF+FBBT	**13924.00**	**i**[5]	**≤0.1**	**3.97**	**13**
HTRU2[1]	17898	8	FFT (worst)	172984.66	-	-	-	-
			FFT (average)	90363.45	-	-	-	-
			FFT (best)	71117.75	-	-	3.95	-
			CPLEX[3]	-	-	-	-	-
			BB+CF	52367.35	11247	≤0.1	-	627
			BB+CF+SB	52367.35	2717	≤0.1	-	171
			BB+CF+FBBT	**52367.35**	**69**	**≤0.1**	**3.90**	**12**
GT[2]	36733	11	FFT (worst)	7576.07	-	-	-	-
			FFT (average)	6068.98	-	-	-	-
			FFT (best)	4565.01	-	-	1.32	-
			CPLEX[3]	-	-	-	-	-
			BB+CF	3071.83	135677	38.06	-	14400
			BB+CF+SB	3071.83	134432	34.29	-	14400
			BB+CF+FBBT	**2976.81**	**20708**	**≤0.1**	**0.95**	**2053**
rds[2]	50000	3	FFT (worst)	0.15	-	-	-	-
			FFT (average)	0.13	-	-	-	-
			FFT (best)	0.11	-	-	2.01	-
			CPLEX[3]	-	-	-	-	-
			BB+CF	0.08	131874	5.01	-	14400
			BB+CF+SB	0.08	162164	1.75	-	14400
			BB+CF+FBBT	**0.08**	**519**	**0.61**	**1.04**	**55**
KEGG[2]	53413	23	FFT (worst)	1.12×10^7	-	-	-	-
			FFT (average)	1.09×10^7	-	-	-	-
			FFT (best)	8.14×10^6	-	-	3.70	-
			CPLEX[4]	-	-	-	-	-
			BB+CF	4.98×10^6	87	≤0.1	-	41
			BB+CF+SB	4.98×10^6	63	≤0.1	-	31
			BB+CF+FBBT	$\mathbf{4.98 \times 10^6}$	**25**	**≤0.1**	**6.05**	**22**
rng_agr[1]	199843	7	FFT (worst)	1.22×10^{11}	-	-	-	-
			FFT (average)	7.01×10^{10}	-	-	-	-
			FFT (best)	4.78×10^{10}	-	-	3.82	-
			CPLEX[4]	-	-	-	-	-
			BB+CF	3.16×10^{10}	36749	5.05	-	14400
			BB+CF+SB	3.16×10^{10}	35650	4.17	-	14400
			BB+CF+FBBT	$\mathbf{3.14 \times 10^{10}}$	**1627**	**≤0.1**	**3.83**	**159**
urbanGB[1]	360177	2	FFT (worst)	19.36	-	-	-	-
			FFT (average)	15.11	-	-	-	-
			FFT (best)	10.75	-	-	2.01	-
			CPLEX[4]	-	-	-	-	-
			BB+CF	5.48	16323	≤ 0.1	-	10713
			BB+CF+SB	5.48	2555	≤ 0.1	-	2033
			BB+CF+FBBT	**5.48**	**117**	**≤0.1**	**2.40**	**40**
spnet3D[1]	434876	3	FFT (worst)	2205.35	-	-	-	-
			FFT (average)	1191.77	-	-	-	-
			FFT (best)	827.39	-	-	3.99	-
			CPLEX[4]	-	-	-	-	-
			BB+CF	569.91	21814	0.32	-	14400
			BB+CF+SB	569.80	7782	≤0.1	-	6509
			BB+CF+FBBT	**569.80**	**85**	**≤0.1**	**3.99**	**29**

[1] Can assign initial cluster through FFT at the root node.
[2] Can Not assign initial cluster through FFT at the root node.
[3] No feasible solution.
[4] Out of memory when using Niagara compute canada.
[5] Solved at the root node.

Table 4.3: Computational results of small-scale datasets. ($S \leq 1,000$)

Data-set	Sam-ple	Dim-ension	Method	UB	Nodes	Gap (%)	Ratio	Time (s)
iris[1]	150	4	FFT (worst)	6.68	-	-	-	-
			FFT (average)	4.79	-	-	-	-
			FFT (best)	3.66	-	-	2.48	-
			CPLEX	2.04	-	≤0.1	-	98
			BB+CF	2.04	12967	≤0.1	-	17
			BB+CF+SB	**2.04**	**2453**	**≤0.1**	-	**12**
			BB+CF+FBBT	**2.04**	**i**[4]	**≤0.1**	**2.86**	**14**
seeds[1]	210	7	FFT (worst)	38.47	-	-	-	-
			FFT (average)	24.54	-	-	-	-
			FFT (best)	14.95	-	-	1.77	-
			CPLEX	10.44	-	≤0.1	-	14400
			BB+CF	10.44	7155	≤0.1	-	17
			BB+CF+SB	**10.44**	**1269**	**≤0.1**	-	**11**
			BB+CF+FBBT	**10.44**	**19**	**≤0.1**	**1.61**	**15**
glass[2]	214	9	FFT (worst)	51.71	-	-	-	-
			FFT (average)	41.23	-	-	-	-
			FFT (best)	27.52	-	-	1.98	-
			CPLEX	35.57	-	64.78	-	1210
			BB+CF	**27.52**	**5559**	**≤0.1**	-	**15**
			BB+CF+SB	**27.52**	**1445**	**≤0.1**	-	**11**
			BB+CF+FBBT	**27.52**	**191**	**≤0.1**	**1.98**	**14**
BM[1]	249	6	FFT (worst)	34335.0	-	-	-	-
			FFT (average)	22890.61	-	-	-	-
			FFT (best)	15245.0	-	-	1.39	-
			CPLEX	10539.0	-	≤0.1	-	14400
			BB+CF	10539.0	14187	≤0.1	-	22
			BB+CF+SB	**10539.0**	**4333**	**≤0.1**	-	**14**
			BB+CF+FBBT	**10539.0**	**47**	**≤0.1**	**1.77**	**15**
UK[2]	258	5	FFT (worst)	1.36	-	-	-	-
			FFT (average)	0.96	-	-	-	-
			FFT (best)	0.72	-	-	1.19	-
			CPLEX	0.74	-	58.30	-	1910
			BB+CF	0.53	315495	≤0.1	-	258
			BB+CF+SB	0.53	87687	≤0.1	-	71
			BB+CF+FBBT	**0.53**	**17770**	**≤0.1**	**0.32**	**25**
HF[1]	299	12	FFT (worst)	6.70×10^{10}	-	-	-	-
			FFT (average)	4.82×10^{10}	-	-	-	-
			FFT (best)	2.53×10^{10}	-	-	4.26	-
			CPLEX[3]	-	-	-	-	-
			BB+CF	$\mathbf{1.72 \times 10^{10}}$	339	**≤0.1**	-	**10**
			BB+CF+SB	$\mathbf{1.72 \times 10^{10}}$	**241**	**≤0.1**	-	**10**
			BB+CF+FBBT	$\mathbf{1.72 \times 10^{10}}$	**i**[4]	**≤0.1**	**4.03**	**12**
Who[2]	440	8	FFT (worst)	7.27×10^9	-	-	-	-
			FFT (average)	5.77×10^9	-	-	-	-
			FFT (best)	4.50×10^9	-	-	1.58	-
			CPLEX[1]	-	-	-	-	-
			BB+CF	$\mathbf{3.49 \times 10^9}$	**3407**	**≤0.1**	-	**15**
			BB+CF+SB	$\mathbf{3.49 \times 10^9}$	**1287**	**≤0.1**	-	**12**
			BB+CF+FBBT	$\mathbf{3.49 \times 10^9}$	**383**	**≤0.1**	**1.18**	**15**
HCV[2]	602	12	FFT (worst)	287219.97	-	-	-	-
			FFT (average)	216190.32	-	-	-	-
			FFT (best)	174716.17	-	-	2.51	-
			CPLEX	141440.68	-	≤0.1	-	965
			BB+CF	**141440.68**	**291**	**≤0.1**	-	**10**
			BB+CF+SB	**141440.68**	**103**	**≤0.1**	-	**10**
			BB+CF+FBBT	**141440.68**	**39**	**≤0.1**	**0.90**	**14**
Abs[2]	740	21	FFT (worst)	39540.58	-	-	-	-
			FFT (average)	26395.93	-	-	-	-
			FFT (best)	18311.19	-	-	1.90	-
			CPLEX	20089.19	-	100	-	14400
			BB+CF	13905.40	33449	≤ 0.1	-	153
			BB+CF+SB	13905.40	13541	≤0.1	-	68
			BB+CF+FBBT	**13905.40**	**585**	**≤0.1**	**1.10**	**16**
TR[2]	980	10	FFT (worst)	14.50	-	-	-	-
			FFT (average)	10.44	-	-	-	-
			FFT (best)	8.01	-	-	1.53	-
			CPLEX	7.82	-	58.50	-	14400
			BB+CF	5.94	741851	≤0.1	-	2953
			BB+CF+SB	5.94	242753	≤0.1	-	876
			BB+CF+FBBT	**5.94**	**39118**	**≤0.1**	**0.34**	**128**
SGC[1]	1000	21	FFT (worst)	3.64×10^7	-	-	-	-
			FFT (average)	1.96×10^7	-	-	-	-
			FFT (best)	1.33×10^7	-	-	3.97	-
			CPLEX	2.97×10^7	-	100	-	14400
			BB+CF	$\mathbf{9.45 \times 10^6}$	411	**≤0.1**	-	**12**
			BB+CF+SB	$\mathbf{9.45 \times 10^6}$	**411**	**≤0.1**	-	**12**
			BB+CF+FBBT	$\mathbf{9.45 \times 10^6}$	**i**[4]	**≤0.1**	**3.97**	**12**

[1] Can assign initial cluster through FFT at the root node.
[2] Can Not assign initial cluster through FFT at the root node.
[3] No feasible solution.
[4] Solved at the root node.

Chapter 5

Convergence Analysis of The Branch and Bound Scheme

In this chapter, we establish the convergence of the BB scheme constructed using only the basic lower and upper bounding problems. The basic lower and upper bounding problems of k-means clustering are presented in Section 3.1.1 and Section 3.2.1, respectively. Moreover, for the k-center clustering, Section 4.1 and Section 4.2 introduce the basic lower and upper bounding problems, respectively. Along BB process, it can generate a monotonically nonincreasing sequence $\{\alpha_i\}$ and a monotonically nondecreasing sequence $\{\beta_i\}$ since the search space is narrowing along the path.

5.1 k-Means Clustering

For the k-means problem, the number of feasible solutions is infinite. In this case, a branch and bound scheme is said to be convergent if $\lim_{i\to\infty} \alpha_i = \lim_{i\to\infty} \beta_i = z^{km}$. Besides, since sample grouping and Lagrangian Decomposition will only provide tighter bounds, they will not break the convergence.

Our proposed BB clustering algorithm can be regarded as a rooted tree. The root node is the original variable space M_0. This node is indexed at level 0. We denote M_{i_q} as the node at level q which is explored at iteration i_q. A node $M_{i_{q+1}}$ is a child node that connected to its parent node M_{i_q}, with $M_{i_{q+1}} \subset M_{i_q}$. The child node is

at level $q+1$ and is explored at iteration i_{q+1}. We denote $\{M_{i_q}\}$ as the sequence of the partition element that represents a path of the tree from the root node to the node M_{i_q} at the level q. If the scheme is convergent, then it produces a global ε-optimal solution in a finite number of steps.

In the proof analysis, we adapt the basic results from the work in [9] and the seminal work in the Chapter IV of [29]. Unlike these works in which the general constraints might implicitly reduce the feasible regions, any point $\mu \in M$ in the clustering problem is a feasible solution. Therefore, we modified definitions and theories accordingly in this thesis.

Lemma 5.1.1 (Corollary IV.1 [29])**.** *If a BB procedure is infinite, then it generates at least one infinitely decreasing sequence $\{M_{i_q}\}$ of successively refined partition elements, $M_{i_{q+1}} \subset M_{i_q}$.*

Definition 5.1.2 (Definition IV.10 [29])**.** A subdivision is called **exhaustive** if $\lim_{q\to\infty} \delta(M_{i_q}) = 0$, for all decreasing sub-sequences $\{M_{i_q}\}$ generated by the subdivision.

It is easy to see that if the element of μ that corresponds to $\delta(M)$ is selected for partitioning, the created subdivision is guaranteed to be exhaustive.

The next several conclusions help to prove the convergence of lower bounds.

Definition 5.1.3 (Definition IV.7 [29])**.** A lower bounding operation is called **strongly consistent**, if, at every iteration, any undeleted partition set can be further refined and if any infinite decreasing sequence $\{M_{i_q}\}$ of successively refined partition elements, contains a sub-sequence $\{M_{i_{q'}}\}$, satisfying, $\lim_{q'\to\infty} \beta(M_{i_{q'}}) = z^{km}(\overline{M})$, where $\overline{M} = \bigcap_q M_{i_q}$.

Lemma 5.1.4. *Given an exhaustive subdivision on μ, The lower bounding operation in Algorithm 1 is strongly consistent.*

Proof. With an exhaustive subdivision, M_{i_q} shrinks to a single point $\bar{\mu}$ and we thus have that $\bar{M} = \{\bar{\mu}\}$. We then prove the lemma by showing that $\lim_{q\to\infty} \beta(M_{i_q}) = z^{km}(\bar{M}) = \sum_{s\in\mathscr{S}} Q_s(\bar{\mu})$. Let $\tilde{\mu}_{i_q,s} \in \{Q_s(\mu_s) : \mu_s \in M_{i_q}\}$, since M_{i_q} shrinks to $\bar{\mu}$, we have $\lim_{q\to\infty} \tilde{\mu}_{i_q,s} = \bar{\mu}$. Based on the continuity of $Q_s(.)$ (Lemma 2.1.1), we have

$Q_s(\overline{\mu}) = \lim_{q\to\infty} Q_s(\tilde{\mu}_{i_q,s}) = \lim_{q\to\infty} \beta_s(M_{i_q})$. Take the sum over s, we obtain $\lim_{q\to\infty} \beta(M_{i_q}) = \sum_{s\in\mathscr{S}} Q_s(\overline{M})$. □

Definition 5.1.5 (Definition IV.6 [29])**.** A selection operation is said to be **bound improving**, if, after a finite number of steps, at least one partition element where the actual lower bounding is attained is selected for further partition.

The selection operation in Algorithm 1 is bound improving, because at each iteration, Algorithm 1 selects the node where the actual lower bounding is attained for further partition.

Lemma 5.1.6 (Theorem IV.3 [29])**.** *For a BB scheme using a lower bounding operation that is strongly consistent and using a selection operation that is bound improving, we have that* $\lim_{i\to\infty} \beta_i = z^{km}$.

Lemma 5.1.7. *Given an exhaustive subdivision on* μ*, Algorithm 1 satisfies* $\lim_{i\to\infty} \beta_i = z^{km}$.

Proof. This result can be obtained from Lemma 5.1.4 and 5.1.6. □

Finally, the convergence of the upper bounds is established through the following lemma.

Lemma 5.1.8. *Given an exhaustive subdivision on* μ*, Algorithm 1 generates a sequence* $\{\alpha_i\}$ *such that* $\lim_{i\to\infty} \alpha_i = z^{km}$.

Proof. Let $\mu^* \in M_0$ denote an optimal solution of the k-means problem. Because of the continuity of Q (implied by the continuity of Q_s), we have $Q(\mu) - Q(\mu^*) \leq K||\mu - \mu^*||$ for all $\mu \in M$. Given $\varepsilon > 0$, $r = \varepsilon/K$, for every point $\mu \in B_r(\mu^*)$, we have $Q(\mu) - Q(\mu^*) \leq K||\mu - \mu^*|| \leq \varepsilon$ holds.

Since the subdivision is exhaustive, after a finite number of iterations i, we have, either the partition considered satisfies $M_i \subseteq B_r(\mu^*)$, or the partition M_i which contain the solution μ^* is pruned. For the first case, since $M_i \subseteq B_r(\mu^*)$, we have $Q(\mu) - Q(\mu^*) \leq \varepsilon, \forall \mu \in M_i$. Then we have that $\alpha_i \leq \alpha(M_i) \leq Q(\mu^*) + \varepsilon$. In the second one, since M_i is pruned, then we have $\alpha_i \leq \alpha(M_i) \leq \beta(M_i) + \varepsilon$. Because $\mu^* \in M_i$, we also have $\beta(M_i) \leq Q(\mu^*)$. Hence, $\alpha_i \leq Q(\mu^*) + \varepsilon$. For both cases,

we have $z^{km} \leq \alpha_i \leq z^{km} + \varepsilon$. Since the above conclusion holds for arbitrary $\varepsilon > 0$, $\lim_{i \to \infty} \alpha_i = z^{km}$ holds. □

Combing Lemma 5.1.7 and 5.1.8, we obtain the following theorem:

Theorem 5.1.9. *Given an exhaustive subdivision on μ, Algorithm 1 converges in the sense that*

$$\lim_{i \to \infty} \alpha_i = \lim_{i \to \infty} \beta_i = z^{km}. \tag{5.1}$$

5.2 k-Center Clustering

For the k-center problem, the proof of the convergence for the BB scheme becomes obvious by noticing that the number of feasible solutions is finite because of the "centers on samples" constraint $\mu \in X$. We can show that $\{\alpha_i\}$ and $\{\beta_i\}$ both converge to z^{kc} in a finite number of steps.

Lemma 5.2.1. *Algorithm 1 terminates in a finite number of steps for the k-center problem.*

Since we add a partition M to the node set $\mathbb{M}$ only if $M^k \cap X \neq \emptyset, \forall k \in \mathscr{K}$, the BB procedure will not generate any infinitely decreasing sequence of successively refined partition elements. In the worse case, a sequence of successively refined partition elements will end at a leaf node in which $|M^k \cap X| = 1, \forall k \in \mathscr{K}$. Since each leaf node corresponds to a feasible solution, the number of leaf nodes is finite and so the number of total nodes visited in a BB procedure is finite. Moreover, at a leaf node M, it can be shown that $\beta(M) = \alpha(M)$. Therefore, the BB procedure will terminate in a finite number of steps by only branching on μ, with the lower bound and upper bound converging to the optimal solution.

Chapter 6

Conclusion

To conclude, throughout this thesis, we have proposed and implemented a scalable global optimization algorithm for the k-means and k-center problems. In our reduced-space algorithm, only the centers of clusters need to be branched, and lower bounding problems can be decomposed into smaller subproblems. We have proved this algorithm converges to a global optimal solution.

In the implementation of the algorithm for k-means problem, the numerical experiments demonstrated our algorithm's ability to handle datasets with up to 200,000 samples, which improves scale of solvable k-means problem by 100 times larger than state of the art. Numerical results also illustrate that the algorithm can provide the deterministic clustering results with good generations on diverse real-world datasets.

For the k-center problem, we have adapted feasibility-based bounds tightening and symmetric breaking to tighten bounds, which have accelerated the convergence of branch and bound scheme in the implementation. We test 29 datasets, even including 6 datasets with millions of samples. Our algorithm is able to solve datasets with up to 14 million samples in 4 hours and get an excellent optimality gap. For the largest case, we can attain an optimality gap of $\leq 0.1\%$ within 2 hours for the dataset with 14,057,567 samples and 3 attributes.

Bibliography

[1] A. F. Agarap and A. P. Azcarraga. Improving k-means clustering performance with disentangled internal representations. *arXiv preprint arXiv:2006.04535*, 2020. → page 2

[2] C. C. Aggarwal, J. L. Wolf, and P. S.-l. Yu. Method for targeted advertising on the web based on accumulated self-learning data, clustering users and semantic node graph techniques, Mar. 30 2004. US Patent 6,714,975. → page 1

[3] D. Aloise and C. Contardo. A sampling-based exact algorithm for the solution of the minimax diameter clustering problem. *Journal of Global Optimization*, 71(3):613–630, 2018. → pages 3, 4

[4] D. Aloise, P. Hansen, and L. Liberti. An improved column generation algorithm for minimum sum-of-squares clustering. *Mathematical Programming*, 131(1):195–220, 2012. → pages viii, 20, 21, 22

[5] C. J. Alpert and A. B. Kahng. Splitting an ordering into a partition to minimize diameter. *Journal of Classification*, 14(1):51–74, 1997. → page 3

[6] F. Barahona and R. Anbil. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming*, 87(3):385–399, 2000. → page 15

[7] H. Calik and B. C. Tansel. Double bound method for solving the p-center location problem. *Computers & operations research*, 40(12):2991–2999, 2013. → page 4

[8] Y. Cao and V. M. Zavala. A scalable global optimization algorithm for stochastic nonlinear programs. *Journal of Global Optimization*, 75(2): 393–416, 2019. → page 2

[9] Y. Cao and V. M. Zavala. A scalable global optimization algorithm for stochastic nonlinear programs. *Journal of Global Optimization*, 75(2): 393–416, 2019. → page 37

[10] C. C. CarøE and R. Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1-2):37–45, 1999. → page 2

[11] D. Chen and R. Chen. New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems. *Computers & Operations Research*, 36(5):1646–1655, 2009. → page 4

[12] R. Chen and G. Y. Handler. Relaxation method for the solution of the minimax location-allocation problem in euclidean space. *Naval Research Logistics (NRL)*, 34(6):775–788, 1987. → page 4

[13] C. Contardo, M. Iori, and R. Kramer. A scalable exact algorithm for the vertex p-center problem. *Computers & Operations Research*, 103:211–220, 2019. → page 4

[14] I. I. Cplex. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009. → page 17

[15] I. I. Cplex. V20.1.0: User's Manual for CPLEX. *International Business Machines Corporation*, 2020. → page 31

[16] M. S. Daskin. A new approach to solving the vertex p-center problem to optimality: Algorithm and computational results. *Communications of the Operations Research Society of Japan*, 45(9):428–436, 2000. → page 3

[17] D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml. → pages 20, 21, 30

[18] K.-C. Duong, C. Vrain, et al. Constrained clustering by constraint programming. *Artificial Intelligence*, 244:70–94, 2017. → page 4

[19] S. Elloumi, M. Labbé, and Y. Pochet. A new formulation and resolution method for the p-center problem. *INFORMS Journal on Computing*, 16(1): 84–94, 2004. → page 3

[20] M. L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management science*, 27(1):1–18, 1981. → page 15

[21] N. Freed and F. Glover. A mixed-integer programming approach to the clustering problem. *Communications in Statistics-Simulation and Computation*, 12(5):595–607, 1983. → page 2

[22] G. Gamrath, D. Anderson, K. Bestuzheva, W.-K. Chen, L. Eifler, M. Gasse, P. Gemander, A. Gleixner, L. Gottwald, K. Halbig, et al. The scip optimization suite 7.0. 2020. → page 2

[23] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979. → page 3

[24] A. M. Geoffrion. Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260, 1972. → page 2

[25] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. → page 3

[26] P. Hansen, J. Brimberg, D. Urošević, and N. Mladenović. Solving large p-median clustering problems by primal–dual variable neighborhood search. *Data Mining and Knowledge Discovery*, 19(3):351–375, 2009. → page 1

[27] Z. R. Hesabi, Z. Tari, A. Goscinski, A. Fahad, I. Khalil, and C. Queiroz. Data summarization techniques for big data—a survey. In *Handbook on Data Centers*, pages 1109–1152. Springer, 2015. → page 1

[28] D. S. Hochbaum and D. B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985. → page 3

[29] R. Horst and H. Tuy. *Global optimization: Deterministic approaches*. Springer Science & Business Media, 2013. → pages 37, 38

[30] K. Hua, M. Shi, and Y. Cao. A Scalable Deterministic Global Optimization Algorithm for Clustering Problems. In *International Conference on Machine Learning*, pages 4391–4401. PMLR, 2021. → pages iii, v

[31] R. Karuppiah and I. E. Grossmann. A lagrangean based branch-and-cut algorithm for global optimization of nonconvex mixed-integer nonlinear programs with decomposable structures. *Journal of global optimization*, 41 (2):163–186, 2008. → page 2

[32] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009. → page 3

[33] A. Khajavirad and J. J. Michalek. A deterministic lagrangian-based global optimization approach for quasiseparable nonconvex mixed-integer nonlinear programs. *Journal of Mechanical Design*, 131(5), 2009. → page 2

[34] M. Kleindessner, P. Awasthi, and J. Morgenstern. Fair k-center clustering for data summarization. In *International Conference on Machine Learning*, pages 3448–3457. PMLR, 2019. → page 1

[35] N. Komodakis, N. Paragios, and G. Tziritas. Clustering via lp-based stabilities. *Advances in neural information processing systems*, 21:865–872, 2008. → page 2

[36] C. Li and I. E. Grossmann. A generalized benders decomposition-based branch and cut algorithm for two-stage stochastic programs with nonconvex constraints and mixed-binary first and second stage variables. *Journal of Global Optimization*, 75(2):247–272, 2019. → page 2

[37] X. Li, A. Tomasgard, and P. I. Barton. Nonconvex generalized benders decomposition for stochastic separable mixed-integer nonlinear programs. *Journal of optimization theory and applications*, 151(3):425, 2011. → page 2

[38] A. Likas, N. Vlassis, and J. J. Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003. → page 2

[39] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. → page 1

[40] T. S. Madhulatha. An overview on clustering methods. *arXiv preprint arXiv:1205.1117*, 2012. → page 1

[41] J. Mihelič and B. Robic. Solving the k-center problem efficiently with a dominating set algorithm. *CIT*, 13:225–234, 09 2005. → page 3

[42] R. Misener and C. A. Floudas. Antigone: algorithms for continuous/integer global optimization of nonlinear equations. *Journal of Global Optimization*, 59(2-3):503–526, 2014. → page 2

[43] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016. → page 2

[44] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1): 60–100, 1991. → pages 20, 21, 30

[45] W. Pan, X. Shen, and B. Liu. Cluster analysis: Unsupervised learning via supervised learning with a non-convex penalty. *Journal of Machine Learning Research*, 14(7), 2013. → page 1

[46] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1): 119–147, 1991. → page 15

[47] B. Sağlam, F. S. Salman, S. Sayın, and M. Türkay. A mixed-integer programming approach to the clustering problem with an application in customer segmentation. *European Journal of Operational Research*, 173(3): 866–879, 2006. → page 2

[48] H. Späth. *Cluster analysis algorithms for data reduction and classification of objects*. Horwood, 1980. → page 1

[49] M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical programming*, 103(2):225–249, 2005. → page 2

[50] G. Tzortzis and A. Likas. The minmax k-means clustering algorithm. *Pattern Recognition*, 47(7):2505–2516, 2014. → page 2

[51] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006. → page 17

[52] J. Xu and K. Lange. Power k-means clustering. In *International Conference on Machine Learning*, pages 6921–6931, 2019. → pages 1, 2