Global Optimization of K-Center Clustering

Mingfei Shi [*] [1] **Kaixun Hua** [*] [1] **Jiayang Ren** [1] **Yankai Cao** [1]

Abstract

k-center problem is a well-known clustering method and can be formulated as a mixed-integer nonlinear programming problem. This work provides a practical global optimization algorithm for this task based on a reduced-space spatial branch and bound 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. In addition, 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 32 datasets. Notably, for the dataset with 14 million samples and 3 features, the serial implementation of the algorithm can converge to an optimality gap of 0.1% within 2 hours. Compared with a heuristic method, the global optimum obtained by our algorithm can reduce the objective function on average by 30.4%.

1. Introduction

Clustering is a fundamental unsupervised machine learning task that plays a vital role in various fields of applications, such as customer grouping (Aggarwal et al., 2004), data summarization (Kleindessner et al., 2019; Hesabi et al., 2015), and facility location determination (Hansen et al., 2009). Clustering aims to aggregate similar data into one cluster and separate those in diverse into different clusters (Pan et al., 2013). Cluster analysis can be formulated as an optimization problem. Various objective functions are designed and lead to different algorithms to find clusters (Madhulatha, 2012).

[*]Equal contribution [1]Department of Chemical and Biological Engineering, University of British Columbia, Vancouver, British Columbia, Canada. Correspondence to: Yankai Cao <yankai.cao@ubc.ca>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

This paper focuses on one of the most fundamental centroid-based clustering problems called 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 (Kaufman & Rousseeuw, 2009). As an NP-hard problem (Gonzalez, 1985), there is no way to achieve an optimal solution in a polynomial-time unless $P = NP$ (Garey & Johnson, 1979). Therefore, many heuristic approximation algorithms are developed to obtain a near-optimal solution quickly.

The 2-approximation greedy algorithm is one of the most effective heuristic ways to solve the k-center problem (Gonzalez, 1985). It starts from a randomly selected center, and then the new centers are chosen as the farthest points to the previously selected centers. Alpert & Kahng (1997) applied a complete-linkage-based algorithm that can outperform traditional algorithms with proper ordering heuristics for sample. However, the experiments in (Aloise & Contardo, 2018) showed that it seldom finds optimal solutions. Parametric pruning (Hochbaum & Shmoys, 1985) 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 (Mihelič & Robic, 2005). 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, Daskin (2000) 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. Elloumi et al. (2004) leveraged the binary search scheme, 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 adopting a constraint programming framework. Duong et al. (2017) proposed a general constraint clustering algorithm that can solve the datasets with up to 5000 samples in 50 seconds. Calik & Tansel (2013) introduces a block of covering constraints for the formulation of k-center problem and updates the lower and upper bounds in the same iteration. However, their algorithm does not perform well on large datasets (Contardo et al., 2019). Recently, the strategy of iteratively solving reduced subproblems has been proposed by researchers to solve the k-center problem to global optimality on large datasets. Aloise & Contardo (2018) 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. Chen & Handler (1987) and Chen & Chen (2009) 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, Contardo et al. (2019) 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.

In this paper, we proposed a different approach to solve the k-center problem on extremely large datasets, by adopting a reduced-space BB scheme recently proposed for two-stage stochastic programming problems (Cao & Zavala, 2019). This reduced-space BB scheme has already been successfully applied for the k-means clustering problem (Hua et al., 2021), in which the reduced-space BB scheme is combined with Lagrangian decomposition to guarantee convergence to the global ϵ-optimal solutions. This method can solve the k-means problem on a dataset with 210,000 samples (200 cores, 2.6% optimality gap within the runtime of 4 hours). Because the objective of k-center is to minimize the maximum within-cluster distance, instead of the average within-cluster distance, we cannot adopt the Lagrangian decomposition approach to compute the lower bound. Also, because of the "centers on samples" constraint in the k-center problem (k-means clustering does not have this constraint), the direct application of Hua's algorithm will lead to infeasible solutions. To address these challenges, we propose a tailored reduced-space BB clustering algorithm for the k-center problem and design several feasibility-based bounds tightening (FBBT) methods to reduce the search space of our BB algorithm efficiently. The novelty of our BB clustering algorithm is that by only branching on the space of centers, we can assure the convergence of our algorithm to the exact global optimum within a finite number of steps for the task of k-center clustering. Note that a classic BB algorithm is unable to solve the k-center problem with large datasets because it needs to branch on all integer variables, the dimension of which scales with the number of samples.

We highlight our contributions as follows:

- We design an exact global optimization algorithm based on a reduced-space spatial branch and bound scheme for the k-center clustering problem. We propose a decomposable approach to compute lower bounds. The lower bound can be analytically derived in a closed-form. The whole algorithm does not need to rely on solvers to solve any optimization sub-problems. We prove that our algorithm is guaranteed to converge to the global optimum by only branching on the centers of clusters.

- 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 can be determined for many samples without knowing the optimal solution. Bounds tightening techniques coupled with the decomposable lower bounding techniques enable our algorithm to be extremely scalable.

- We provide an open-source Julia implementation of the algorithm and perform extensive computational experiments on 32 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 obtain the global solution to an optimality gap of 0.1% within 2 hours.

2. k-center Formulation

To formulate k-center problem, given a dataset with S samples and A attributes, denoted as $X = \{x_1, \ldots, x_S\} \in \mathbb{R}^{A\times S}$, 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 k-center problem can be defined as:

$$\min_{\mu\in X} \max_{s\in\mathcal{S}} \min_{k\in\mathcal{K}} ||x_s - \mu^k||_2^2 \tag{1}$$

where $s \in \mathcal{S} := \{1, \cdots, S\}$ is the data sample set, $k \in \mathcal{K} := \{1, \cdots, K\}$ is the cluster set, $\mu := [\mu^1, \cdots, \mu^K]$ represents the center of each cluster. μ are the variables to be determined. We use $\mu \in X$ to denote the "centers on samples" constraint that the center of each cluster is restricted to some data samples.

Problem 1 can be reformulated as a problem with the form:

$$z = \min_{\mu \in M_0 \cap X} \max_{s \in S} Q_s(\mu). \tag{2}$$

where $Q_s(\mu) = \min_{k \in \mathcal{K}} ||x_s - \mu^k||_2^2$. We denote a closed set $M_0 := \{\mu \mid \underline{\mu} \leq \mu \leq \bar{\mu}\}$ in Equation 2 as the domain of centers, where $\underline{\mu}$ represents the lower bound of centers and $\bar{\mu}$ is the upper bound, i.e., $\underline{\mu}_a^k = \min_s X_{s,a}$, $\bar{\mu}_a^k = \max_s X_{s,a}$, $\forall k \in \mathcal{K}$, $a \in \{1, \cdots, A\}$. Here the constraint $\mu \in M_0$ can be inferred directly from data, and thus does not affect optimal solution. It is introduced to simplify the discussion of the BB framework. Since $\mu \in M_0 \cap X$, which is a finite set, we can attain the value of z in Problem 2. This formulation will be utilized in Section 3.1 to introduce the lower bounding problem in the branch and bound scheme.

Alternatively, the k-center problem also can be represented as a standard optimization problem with the following extensive form:

$$\min_{\mu,d,b,\lambda} \; d_* \tag{3a}$$

$$\text{s.t.} \;\; d_s^k \geq ||x_s - \mu^k||_2^2 \tag{3b}$$

$$-N_1(1 - b_s^k) \leq d_s^* - d_s^k \leq 0 \tag{3c}$$

$$d_* \geq d_s^* \tag{3d}$$

$$\sum_{k \in \mathcal{K}} b_s^k = 1 \tag{3e}$$

$$b_s^k \in \{0, 1\} \tag{3f}$$

$$-N_2(1 - \lambda_s^k) \leq x_s - \mu^k \leq N_2(1 - \lambda_s^k) \tag{3g}$$

$$\sum_{s \in \mathcal{S}} \lambda_s^k = 1 \tag{3h}$$

$$\lambda_s^k \in \{0, 1\} \tag{3i}$$

$$b_s^k \geq \lambda_s^k \tag{3j}$$

$$s \in \mathcal{S}, k \in \mathcal{K} \tag{3k}$$

where d_s^k represents the distance between sample x_s and center μ^k, d_s^* denotes the distance between x_s and the center of its cluster, N_1 and N_2 are both arbitrary large values. b_s^k and λ_s^k are two binary variables. b_s^k is equal to 1 if sample x_s belongs to the kth cluster, and 0 otherwise. λ_s^k is equal to 1 if μ^k, which is the center of the kth cluster, is chosen on x_s, and 0 otherwise.

Constraint 3c 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 3e guarantees that sample x_s belongs to only one cluster. We also adopt Constraint 3g, 3h and 3j to represent $\mu \in X$, the restriction that centers are selected on data samples for each cluster. Specifically, Constraint 3g uses a big M formula to make sure that $\mu^k = x_s$ if $\lambda_s^k = 1$, and Constraint 3h confirms that each center can only be selected on one sample. Constraint 3j ensures that if x_s is the center of the kth cluster, then obviously it is assigned to the kth cluster. Problem 3 is a convex mixed-integer nonlinear problem and can be solved by optimization solvers such as CPLEX (Cplex, 2020) or Gurobi (Optimization, 2020). However, within a certain time limit (e.g. 4 hours), these all-purpose solvers can only deal with small datasets.

3. Reduced-space Branch and Bound Scheme

In this section, we introduce a reduced-space BB algorithm for k-center problem with tailored lower bounding and upper bounding methods.

3.1. Lower Bounds

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:

$$z(M) = \min_{\mu \in M \cap X} \max_{s \in \mathcal{S}} Q_s(\mu) \tag{4}$$

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:

$$\min_{\mu_s \in M \cap X} \max_{s \in \mathcal{S}} Q_s(\mu_s) \tag{5a}$$

$$\text{s.t.} \quad \mu_s = \mu_{s+1}, s \in \{1, \cdots, S-1\} \tag{5b}$$

By removing the "centers on samples" constraint $\mu \in X$ and the Constraints 5b, we attain a lower bounding formulation as follow:

$$\beta(M) := \min_{\mu_s \in M} \max_{s \in \mathcal{S}} Q_s(\mu_s). \tag{6}$$

With constraints relaxed, the feasible region of Problem 6 is a superset of the primal feasible region. Therefore, it is obvious that $\beta(M) \leq z(M)$.

In Problem 6, since each sample is independent, it is obvious that:

$$\beta(M) = \max_{s \in \mathcal{S}} \min_{\mu_s \in M} Q_s(\mu_s). \tag{7}$$

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

$$\beta_s(M) = \min_{\mu \in M} Q_s(\mu), \tag{8}$$

with $\beta(M) = \max_{s \in \mathcal{S}} \beta_s(M)$. 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 $Q_s(\mu) = \min_{k\in\mathcal{K}} ||x_s - \mu^k||_2^2$, we have

$$\beta_s(M) = \min_{k\in\mathcal{K}} \min_{\mu^k\in M^k} ||x_s - \mu^k||_2^2, \tag{9}$$

which can be further decomposed into K subsubproblems:

$$\beta_s^k(M^k) = \min_{\mu^k\in M^k} ||x_s - \mu^k||_2^2. \tag{10}$$

with $\beta_s(M) = \min_k \beta_s^k(M^k)$. It can be easily derived that the analytical solution to Problem 10 is: $\mu_a^k = \text{mid}\{\underline{\mu}_a^k,\ x_{s,a},\ \bar{\mu}_a^k\}, \forall a \in \{1, \cdots, A\}$.

3.2. Upper Bounds

An upper bounds of Problem 4 can be obtained by fixing the centers at a candidate feasible solution $\hat{\mu} \in M \cap X$. In this way, we can compute the upper bound base on the following equation:

$$\alpha(M) = \max_{s\in\mathcal{S}} Q_s(\hat{\mu}). \tag{11}$$

It is easy to see that $z(M) \leq \alpha(M), \forall \hat{\mu} \in M \cap X$. Since the closed-form expression of $Q_s(\hat{\mu})$ is given, 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 1 describes the details of farthest first traversal, where $d(x_s, T)$ represents the minimum distance from sample x_s to any sample in set T. 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 1 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
```

3.3. Branch and Bound Scheme and Convergence Analysis

We tailor the reduced-space branch and bound scheme to solve Problem 2. The details of the algorithm are given in Algorithm 2. In the algorithm, We denote $relint(.)$ as the relative interior of a set.

Algorithm 2 Branch and Bound Scheme

```
Initialization
Initialize the iteration index $i \leftarrow 0$;
Set $\mathbb{M} \leftarrow \{M_0\}$, and tolerance $\epsilon > 0$;
Compute initial upper and lower bounds $\alpha_i = FFT(M_0)$, $\beta_i = \beta(M_0)$;
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. $relint(M_1) \cap relint(M_2) = \emptyset$ and $M_1 \cup M_2 = M$;
    Update $\mathbb{M} \leftarrow \mathbb{M} \cup \{M_1\}$, if $M_1^k \cap X \neq \emptyset, \forall k \in \mathcal{K}$
    Update $\mathbb{M} \leftarrow \mathbb{M} \cup \{M_2\}$, if $M_2^k \cap X \neq \emptyset, \forall k \in \mathcal{K}$
    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 $\beta_i - \alpha_i \leq \epsilon$, STOP;
end while
```

We can also establish the convergence of the branch-and-bound scheme in Algorithm 2. Along BB process, it can generate a monotonically nonincreasing sequence $\{\alpha_i\}$ and a monotonically nondecreasing sequence $\{\beta_i\}$. We can show that they both converge to z in a finite number of steps.

Lemma 3.1. *Algorithm 2 terminates in a finite number of steps.*

The proof of the convergence becomes obvious by noticing that the number of feasible solutions is finite because of the "centers on samples" constraint $\mu \in X$. Since we add a partition M to the node set $\mathbb{M}$ only if $M^k \cap X \neq \emptyset, \forall k \in \mathcal{K}$, 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 $|M^k \cap X| = 1, \forall k \in \mathcal{K}$. 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.

Note that the convergence of the k-center problem here is stronger than the convergence analysis in (Cao & Zavala, 2019) for two-stage nonlinear optimization problems or the convergence proof in (Hua et al., 2021) for k-means clustering problem. Both Cao & Zavala (2019) and Hua et al.

(2021) guarantee the convergence in the sense of $\lim_{i\to\infty} \alpha_i = \lim_{i\to\infty} \beta_i = z$. In a finite number of steps, they can only produce a global ϵ-optimal solution. While for the k-center problem, the algorithm can obtain an exact optimal solution (e.g. $\epsilon = 0$) in a finite number of steps, because the number of feasible solutions is finite.

4. Bounds Tightening Techniques

Although the lower bound introduced in Section 3.1 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 2 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.

4.1. 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 3 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 3.2. Then from Objective 3a and Constraint 3c in Formulation 3, we have $d_s^* \le \alpha$. Based on Constraint 3b and 3c, we can conclude that if $b_s^k = 1$, then $||x_s - \mu^k||_2^2 \le \alpha$. Therefore, we have Lemma 4.1.

Lemma 4.1. *If sample x_s is in the kth cluster, then $||x_s - \mu^k||_2^2 \le \alpha$, where α is an upper bound of the k-center problem.*

Lemma 4.1 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 4.2.

Lemma 4.2. *If two samples x_i and x_j are in the same cluster, then $||x_i - x_j||_2^2 \le 4\alpha$ where α is an upper bound of the k-center problem.*

Lemma 4.2 is obvious by noticing that if x_i and x_j all belong to the kth cluster, then based on Lemma 4.1 we have $||x_i - \mu^k||_2^2 \le \alpha$ and $||x_j - \mu^k||_2^2 \le \alpha$. Thus $||x_i - x_j||_2^2 = ||x_i - \mu^k + \mu^k - x_j||_2^2 \le (||x_i - \mu^k||_2 + ||\mu^k - x_j||_2)^2 \le 4\alpha$. Lemma 4.2 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.

4.1.1. K Farthest Points

By Lemma 4.2, 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 1) to obtain K farthest points. For about half of the case studies shown in Section 5, 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.

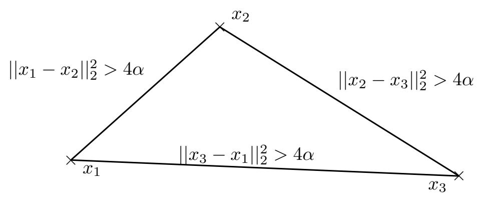
x_2
$||x_1 - x_2||_2^2 > 4\alpha$
$||x_2 - x_3||_2^2 > 4\alpha$
$||x_3 - x_1||_2^2 > 4\alpha$
x_1
x_3

Figure 1. K farthest points with 3 clusters. In this example, $||x_1 - x_2||_2^2 > 4\alpha$, $||x_2 - x_3||_2^2 > 4\alpha$ and $||x_3 - x_1||_2^2 > 4\alpha$. Therefore, we can arbitrarily assign x_1, x_2, x_3 to different 3 clusters.

4.1.2. Center Based Assignment

By Lemma 4.1, 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 $b_s^k = 0, \forall k \in \mathcal{K} \setminus \{k'\}$, 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 $\beta_s^k(M^k) = \min_{\mu^k \in M^k} ||x_s - \mu^k||_2^2 \le ||x_s - \mu^k||_2^2$. Therefore, if $\beta_s^k(M^k) > \alpha$. then by Lemma 4.1, sample x_s is definitely not in the kth cluster and $b_s^k = 0$. In summary, for sample x_s, if $\forall k \in \mathcal{K} \setminus \{k'\}$, $\beta_s^k(M^k) > \alpha$, then x_s is assigned to cluster k' with $b_s^{k'} = 1$. Figure 2 illustrates an example in two-dimensional space with a total of three clusters.

This center based method can be adopted 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

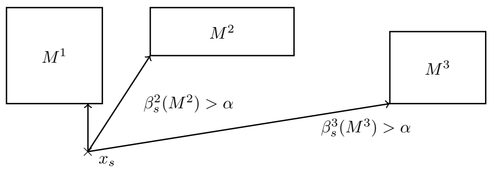

Figure 2. Center based assignment with 3 clusters. In this example, $\beta_s^2(M^2) > \alpha$ ($b_s^2 = 0$) and $\beta_s^3(M^3) > \alpha$ ($b_s^3 = 0$). Therefore, we assign x_s to the first cluster ($b_s^1 = 1$).

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.

4.1.3. 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 4.2, 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 3.

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. Based on this condition, sample based assignment is utilized only after the algorithm has already determined at least one sample for each cluster.

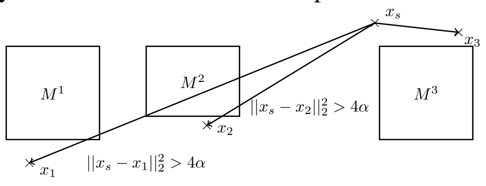

Figure 3. Sample based assignment with 3 clusters. Assume we have already known that x_1, x_2, x_3 belong to cluster $1, 2$ and 3, respectively. x_s is the sample to be determined. In this example, $||x_s - x_1||_2^2 > 4\alpha$ ($b_s^1 = 0$) and $||x_s - x_2||_2^2 > 4\alpha$ ($b_s^2 = 0$). Therefore,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

4.2. Feasibility-Based Bounds Tightening

In this subsection we adopt the Feasibility-Based Bounds Tightening (FBBT) technique to reduce the space of μ.

4.2.1. Ball-based Bounds Tightening

For a sample j, denote $B_\alpha(x_j)$=$\{x| \; ||x - x_j||_2^2 \le \alpha\}$ as the ball with center x_j and radius $\sqrt{\alpha}$. By using methods described in subsection 4.1, assume we already know that sample j belongs to cluster k, by Lemma 4.1, then $\mu^k \in B_\alpha(x_j)$. We use $\mathcal{J}_A^k$ to denote the index of all samples assigned to cluster k, i.e., $\mathcal{J}_A^k = \{j \in S \mid b_j^k = 1\}$, then $\mu^k \in B_\alpha(x_j), \forall j \in \mathcal{J}_A^k$. Besides this, we also know that $\mu^k \in M^k \cap X$. Denote $\mathcal{S}_+^k$ as the index set of samples satisfy all these constraints, $S_+^k(M) := \{s \in S \; |x_s \in M^k, x_s \in B_\alpha(x_j), \forall j \in \mathcal{J}_A^k\}$. In this way, we can obtain a tightened box containing all feasible solutions of kth medoid, $\hat{M}^k$=$\{\mu^k | \hat{\underline{\mu}}^k \le \mu^k \le \hat{\bar{\mu}}^k\}$, with the bounds of ath attribute in kth medoid to be $\hat{\underline{\mu}}_a^k$= $\min_{s \in S_+^k(M)} x_{s,a}^k$ and $\hat{\bar{\mu}}_s^k$= $\max_{s \in S_+^k(M)} x_{s,a}^k$. Figure 4 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 $j \in \mathcal{J}_A^k$. 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 (default: 50) utilized to tighten bounds.

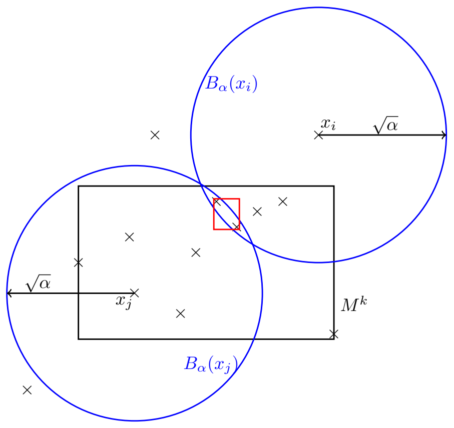

Figure 4. Ball-based bounds tightening in two-dimensional space. In this example, suppose it is determined that two points x_i and x_j belong to the kth cluster. We first compute the index set of samples within all balls and original box, $S_+^k(M) := \{s \in S \; |x_s \in M^k \cap B_\alpha(x_i) \cap B_\alpha(x_j)\}$. We then generate the smallest box containing these samples in $S_+^k(M)$. The red rectangle is the tightened bounds we obtain.

4.2.2. Box-based Bounds Tightening

Another strategy to reduce the computation burden is based on the relaxation of $B_\alpha(x_j)$. For any ball $B_\alpha(x_j)$, the closed set $R_\alpha(x_j) = \{x \mid x_j - \sqrt{\alpha} \le x \le x_j + \sqrt{\alpha}\}$ is the smallest box containing $B_\alpha(x_j)$. Then we have $\mu^k \in R_\alpha(x_j), \forall j \in \mathcal{J}_A^k$. Since $R_\alpha(x_j)$ and M^k are all boxes, we can easily compute the tighten bounds $\hat{M}^k$=$\bigcap_{j \in \mathcal{J}_A^k} R_\alpha(x_j) \cap M^k$. Figure 5 gives an example of box-based bounds tightening using this method. Obviously, the bounds generated in Figure 4 is much tighter while the method in Figure 5 is much faster. Consequently, if $|\mathcal{J}_A^k|$ is small for all clusters, the ball-based bounds tightening method gives more satisfactory results. While if $|\mathcal{J}_A^k|$ is large for any k, box-based bounds tightening provides a cheaper alternative.

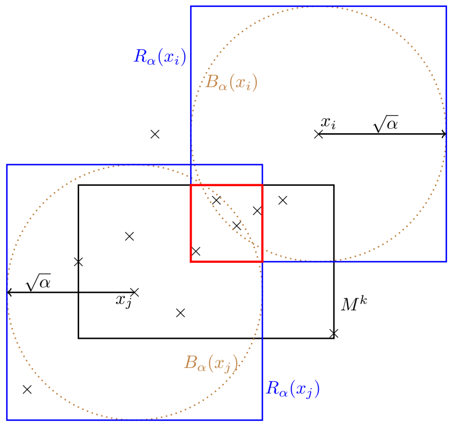

Figure 5. Box-based bounds tightening in two-dimensional space. In this example, we first generate two boxes with $R_\alpha(x_i) := \{x|\ x_i - \sqrt{\alpha} \leq x \leq x_i + \sqrt{\alpha}\}$ and $R_\alpha(x_j) = \{x|\ x_j - \sqrt{\alpha} \leq x \leq x_j + \sqrt{\alpha}\}$. We then create a tighten bounds with $\hat{M}^k{=}R_\alpha(x_i) \cap R_\alpha(x_j) \cap M^k$. The red rectangle is the tightened bounds we obtain.

4.3. 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 2, in which μ_a^k 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. In our implementation, symmetric breaking is used only when initial seeds are not found from FFT at the root node.

5. Computational Experiments

The branch and bound scheme is tested with different methods, including closed-form solution (`BB+CF`) and closed-form with FBBT (`BB+CF+FBBT`). We compare the numerical results of our algorithm with the state-of-art global optimizer CPLEX 20.1.0 (Cplex, 2020) and the heuristic algorithm, Farthest First Traversal. Since the initial point is selected randomly in FFT, we run 100 trails of FFT with different initialization to avoid this randomness and obtain the best (`FFT (BEST)`) and average (`FFT (AVERAGE)`)

Table 1. Computational results on synthetic datasets

DATASET	METHOD	UB	NODES	GAP (%)	TIME (S)
SYN-300[2]	FFT (AVERAGE)	152.79	-	-	-
	FFT (BEST)	118.72	-	-	-
	CPLEX+Q	82.71	86880	100	14400
	CPLEX+L3	66.97	36797	8.55	73
	CPLEX+L3+CUT	61.75	1330122	0.83	7668
	BB+CF	61.75	55191	≤ 0.1	46
	BB+CF+FBBT	**61.75**	**17**	$\leq$ **0.1**	**12**
SYN-1200[1]	FFT (AVERAGE)	216.58	-	-	-
	FFT (BEST)	135.83	-	-	-
	CPLEX+Q	84.81	1343190	1.61	14400
	CPLEX+L3	92.68	10021665	15.78	14400
	CPLEX+L3+CUT	92.68	336524	16.55	14400
	BB+CF	84.81	1155375	≤ 0.1	3609
	BB+CF+FBBT	**84.81**	**i**[3]	$\leq$ **0.1**	**11**
SYN-2100[1]	FFT (AVERAGE)	191.36	-	-	-
	FFT (BEST)	153.22	-	-	-
	CPLEX+Q	216.85	4083121	71.89	14400
	CPLEX+L3	144.78	2566049	36.53	14400
	CPLEX+L3+CUT	235.4	153026	100	14400
	BB+CF	95.10	1495899	≤ 0.1	11606
	BB+CF+FBBT	**95.10**	**i**[3]	$\leq$ **0.1**	**11**
SYN-42,000[2]	FFT (AVERAGE)	290.13	-	-	-
	FFT (BEST)	194.67	-	-	-
	CPLEX+Q	NO FEASIBLE SOLUTION			
	CPLEX+L3	378.14	39353	100	14400
	CPLEX+L3+CUT	800.45	14	100	14400
	BB+CF	142.33	172702	7.14	14400
	BB+CF+FBBT	**142.33**	**103**	$\leq$ **0.1**	**18**
SYN-210,000[1]	FFT (AVERAGE)	300.98	-	-	-
	FFT (BEST)	215.00	-	-	-
	CPLEX+Q	NO FEASIBLE SOLUTION			
	CPLEX+L3	NO FEASIBLE SOLUTION			
	CPLEX+L3+CUT	NO FEASIBLE SOLUTION			
	BB+CF	168.57	43626	7.56	14400
	BB+CF+FBBT	**168.57**	**i**[3]	$\leq$ **0.1**	**16**

[1] CAN ASSIGN INITIAL CLUSTER THROUGH FFT AT THE ROOT NODE.
[2] CAN NOT ASSIGN INITIAL CLUSTER THROUGH FFT AT THE ROOT NODE.
[3] SOLVED AT THE ROOT NODE.

results. We test the performance of our algorithm and CPLEX 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. All experiments are implemented in Julia 1.6.1 and the complete code files can be found in `https://github.com/YankaiGroup/Kcenter`.

The computational results are compared by three criteria, including upper bound (UB), optimality gap, and the number of solved BB nodes. UB measures the best feasible solution. The optimality gap represents the relative difference between the best 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. The number of solved nodes is the iteration number of BB scheme before the termination.

5.1. Numerical Results on Synthetic Datasets

We first consider numerical results on artificially generated datasets from (Hua et al., 2021). To illustrate the performance of the algorithms, we consider synthetic datasets with different numbers of samples. All datasets are generated with 3 Gaussian clusters randomly. Each data sample has two attributes.

Table 1 compares the performance of our reduced space BB algorithm, heuristic algorithm FFT, and CPLEX. `FFT (BEST)` fails to get the lowest UB in all the datasets. The direct usage of CPLEX on Problem 3 (`CPLEX+Q`) can not converge to a small optimality gap ($\leq 0.1\%$) within 4 hours on all the synthetic datasets. Compared with FFT and CPLEX, our algorithms can obtain the best upper bounds and reach a satisfactory gap ($\leq 0.1\%$) in all the datasets within the run-time of 4 hours.

Table 2. Computational results on small-scale datasets ($S \leq 1,000$)

Dataset	Sample	Dimension	Method	UB	Nodes	Gap (%)	Time (s)
			FFT (average)	4.79	-	-	-
			FFT (best)	3.66	-	-	-
iris[1]	150	4	CPLEX+Q	2.04	1512876	$\leq$0.1	548
			BB+CF	2.04	12967	$\leq$0.1	17
			BB+CF+FBBT	**2.04**	**i[3]**	**$\leq$0.1**	**14**
			FFT (average)	24.54	-	-	-
			FFT (best)	14.95	-	-	-
seeds[1]	210	7	CPLEX+Q	10.44	1093710	8.11	14430
			BB+CF	10.44	7155	$\leq$0.1	17
			BB+CF+FBBT	**10.44**	**19**	**$\leq$0.1**	**15**
			FFT (average)	41.23	-	-	-
			FFT (best)	27.52	-	-	-
glass[2]	214	9	CPLEX+Q		Out of memory		
			BB+CF	**27.52**	**5559**	**$\leq$0.1**	**15**
			BB+CF+FBBT	**27.52**	**191**	**$\leq$0.1**	**14**
			FFT (average)	22890.61	-	-	-
			FFT (best)	15245.0	-	-	-
BM[1]	249	6	CPLEX+Q		No feasible solution		
			BB+CF	10539.0	14187	$\leq$0.1	22
			BB+CF+FBBT	**10539.0**	**47**	**$\leq$0.1**	**15**
			FFT (average)	0.96	-	-	-
			FFT (best)	0.72	-	-	-
UK[2]	258	5	CPLEX+Q		Out of memory		
			BB+CF	0.53	315495	$\leq$0.1	258
			BB+CF+FBBT	**0.53**	**17770**	**$\leq$0.1**	**25**
			FFT (average)	4.82×10^{10}	-	-	-
			FFT (best)	2.53×10^{10}	-	-	-
HF[1]	299	12	CPLEX+Q		No feasible solution		
			BB+CF	$\mathbf{1.72 \times 10^{10}}$	339	**$\leq$0.1**	**10**
			BB+CF+FBBT	$\mathbf{1.72 \times 10^{10}}$	i[3]	**$\leq$0.1**	**12**
			FFT (average)	5.77×10^{9}	-	-	-
			FFT (best)	4.50×10^{9}	-	-	-
Who[2]	440	8	CPLEX+Q		No feasible solution		
			BB+CF	$\mathbf{3.49 \times 10^{9}}$	**3407**	**$\leq$0.1**	**15**
			BB+CF+FBBT	$\mathbf{3.49 \times 10^{9}}$	**383**	**$\leq$0.1**	**15**
			FFT (average)	216190.32	-	-	-
			FFT (best)	174716.17	-	-	-
HCV[2]	602	12	CPLEX+Q	141440.68	-	$\leq$0.1	965
			BB+CF	**141440.68**	**291**	**$\leq$0.1**	**10**
			BB+CF+FBBT	**141440.68**	**39**	**$\leq$0.1**	**14**
			FFT (average)	26395.93	-	-	-
			FFT (best)	18311.19	-	-	-
Abs[2]	740	21	CPLEX+Q	18647.59	940299	55.33	14400
			BB+CF	13905.40	33449	$\leq$ 0.1	153
			BB+CF+FBBT	**13905.40**	**585**	**$\leq$0.1**	**16**
			FFT (average)	10.44	-	-	-
			FFT (best)	8.01	-	-	-
TR[2]	980	10	CPLEX+Q	8.32	1556105	54.48	14400
			BB+CF	5.94	741851	$\leq$0.1	2953
			BB+CF+FBBT	**5.94**	**39118**	**$\leq$0.1**	**128**
			FFT (average)	1.96×10^{7}	-	-	-
			FFT (best)	1.33×10^{7}	-	-	-
SGC[1]	1000	21	CPLEX+Q	1.44×10^{7}	48015	100	14400
			BB+CF	$\mathbf{9.45 \times 10^{6}}$	411	**$\leq$0.1**	**12**
			BB+CF+FBBT	$\mathbf{9.45 \times 10^{6}}$	i[3]	**$\leq$0.1**	**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] Solved at the root node.

To make the comparison fair, we also show another two version of using CPLEX. In the first version (`CPLEX+L3`), Equation 3b in Problem 3 is linearized by adding three supporting hyperplane for each sample. However, after linearization, CPLEX still struggles for datasets with more than 1,200 samples. Moreover, three supporting hyperplane is not enough to obtain a solution close to the original optimal solution. But adding supporting hyperplanes will increase the computaional time. In the second version (`CPLEX+L3+CUT`), we manually add a callback of user cuts to enforce $\mu^k \in M^k$ for each BB node, but CPLEX's performance is not improved. We suspect that CPLEX has already inferred this from Constraint 3g. As these two versions does not significantly improve the performance of CPLEX, we do not show their performance in the following numerical experiments.

5.2. Numerical Results on Real-world datasets

We then evaluate the performance of our algorithms on 30 datasets from UCI Machine Learning Repository (Dua & Graff, 2017), one dataset called PR2392 from (Padberg & Rinaldi, 1991), and one dataset called HEMI from (Wang et al., 2022). The number of samples varies from 150 to 14,057,567. The number of attribute ranges from 2 to 68.

Table 2 demonstrates the computational performance of datasets with less than 1000 samples. Table 3 shows the results of datasets with more than 1000 samples. In Table 4, we illustrate the numerical results of datasets with millions of samples.

In these tables, even the best results of FFT can be far away from optimal. This is true even for very small datasets. For example, for `IRIS` dataset, `FFT (BEST)` obtains the best UB of 3.66 while our algorithm and `CPLEX+Q` give a UB of 2.04 with $\leq 0.1\%$ gap. Compared with `FFT (BEST)`, our algroithm can reduce the UB by 30.4%, on average for these 32 datasets.

For small datasets, our algorithms obtain the same UB as CPLEX. However, `CPLEX+Q` takes significantly more computational time than our algorithms. For all datasets with more than 740 samples, `CPLEX+Q` cannot even close the optimality gap to $\leq 50\%$ within 4 hours. For comparison, our algorithms, `BB+CF` and `BB+CF+FBBT` can all generate the best UB and a satisfactory gap ($\leq 0.1\%$) for the majority of the datasets.

The comparison of these two versions highlights the importance of FBBT, which can significantly reduce the computational time and the number of BB nodes to solve the problems. Remarkably, it can even solve several datasets in the root node (`NODES`=$\mathbf{i}^3$). Moreover, the number of nodes needed to reach the optimal solution is much smaller when initial seeds are found, which shows that the initial seeds can be essential for cluster assignment and bounds tightening. For about half of the datasets, initial seeds can be obtained from FFT. Both versions of our algorithms maintain a balanced memory usage during the whole execution process, while for datasets with size over 50,000 samples, `CPLEX+Q` returns the out of memory error.

For most of the datasets with millions of samples in Table 4, `BB+CF+FBBT` can converge to small gaps ($\leq 0.1\%$) and provide the best optimal solution after 4 hours of running. To the best of our 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 over **14 million** samples.

Table 3. Computational results on large-scale datasets ($1,000<S<1,000,000$)

Dataset	Sample	Dimension	Method	UB	Nodes	Gap (%)	Time (s)
hemi[1]	1955	7	FFT (average)	149300.03	-	-	-
			FFT (best)	105657.63	-	-	-
			CPLEX+Q	407892.47	141533	100	14400
			BB+CF	64872.92	1275	$\leq$0.1	18
			BB+CF+FBBT	**64872.92**	**13**	**$\leq$0.1**	**15**
pr2392[2]	2392	2	FFT (average)	6.96×10^7	-	-	-
			FFT (best)	3.99×10^7	-	-	-
			CPLEX+Q	5.30×10^7	151218	100	14400
			BB+CF	2.93×10^7	59327	$\leq$0.1	297
			BB+CF+FBBT	$\mathbf{2.93 \times 10^7}$	**241**	**$\leq$0.1**	**16**
TRR[2]	5454	24	FFT (average)	136.19	-	-	-
			FFT (best)	101.55	-	-	-
			CPLEX+Q	166.61	0	100	14400
			BB+CF	89.78	357283	271.06	14400
			BB+CF+FBBT	88.30	277504	178.99	14400
AC[2]	7195	22	FFT (average)	4.27	-	-	-
			FFT (best)	3.59	-	-	-
			CPLEX+Q	No feasible solution			
			BB+CF	2.75	313564	74.30	14400
			BB+CF+FBBT	2.78	267412	63.65	14400
rds_cnt[1]	10000	4	FFT (average)	38772.90	-	-	-
			FFT (best)	20449.0	-	-	-
			CPLEX+Q	56644.01	35753	100	14400
			BB+CF	13924.00	639	$\leq$0.1	25
			BB+CF+FBBT	**13924.00**	**i**[3]	**$\leq$0.1**	**13**
HTRU2[1]	17898	8	FFT (average)	90363.45	-	-	-
			FFT (best)	71117.75	-	-	-
			CPLEX+Q	No feasible solution			
			BB+CF	52367.35	11247	$\leq$0.1	627
			BB+CF+FBBT	**52367.35**	**69**	**$\leq$0.1**	**12**
GT[2]	36733	11	FFT (average)	6068.98	-	-	-
			FFT (best)	4565.01	-	-	-
			CPLEX+Q	No feasible solution			
			BB+CF	3071.83	135677	38.06	14400
			BB+CF+FBBT	**2976.81**	**20708**	**$\leq$0.1**	**2053**
rds[2]	50000	3	FFT (average)	0.13	-	-	-
			FFT (best)	0.11	-	-	-
			CPLEX+Q	No feasible solution			
			BB+CF	0.08	131874	5.01	14400
			BB+CF+FBBT	**0.08**	**519**	**0.61**	**55**
KEGG[2]	53413	23	FFT (average)	1.09×10^7	-	-	-
			FFT (best)	8.14×10^6	-	-	-
			CPLEX+Q	Out of memory			
			BB+CF	4.98×10^6	87	$\leq$0.1	41
			BB+CF+FBBT	$\mathbf{4.98 \times 10^6}$	**25**	**$\leq$0.1**	**22**
rng_agr[1]	199843	7	FFT (average)	7.01×10^{10}	-	-	-
			FFT (best)	4.78×10^{10}	-	-	-
			CPLEX+Q	Out of memory			
			BB+CF	3.16×10^{10}	36749	5.05	14400
			BB+CF+FBBT	$\mathbf{3.14 \times 10^{10}}$	**1627**	**$\leq$0.1**	**159**
urbanGB[1]	360177	2	FFT (average)	15.11	-	-	-
			FFT (best)	10.75	-	-	-
			CPLEX+Q	Out of memory			
			BB+CF	5.48	16323	$\leq$ 0.1	10713
			BB+CF+FBBT	**5.48**	**117**	**$\leq$0.1**	**40**
spnet3D[1]	434876	3	FFT (average)	1191.77	-	-	-
			FFT (best)	827.39	-	-	-
			CPLEX+Q	Out of memory			
			BB+CF	569.91	21814	0.32	14400
			BB+CF+FBBT	**569.80**	**85**	**$\leq$0.1**	**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] Solved at the root node.

6. Conclusion

This paper propose a global optimization algorithm for k-center problem using the branch and bound scheme. In our reduced-space algorithm, only the centers of clusters need to be branched. We prove that this algorithm converges to a global optimal solution. We also adopt the feasibility-based bounds tightening techniques to tighten bounds, which have accelerated the convergence of branch and bound scheme. We test 32 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 small 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.

Table 4. Computational results on datasets with millions of samples

Dataset	Sample	Dimension	Method	UB	Nodes	Gap (%)	Time (s)
USC1990[1]	2,458,285	68	FFT (average)	4.23×10^{11}	-	-	-
			FFT (best)	2.44×10^{11}	-	-	-
			CPLEX+Q	Can not be solved			
			BB+CF	1.69×10^{11}	916	3.75	14400
			BB+CF+FBBT	$\mathbf{1.68 \times 10^{11}}$	**i**[3]	**$\leq$0.1**	**317**
Gas_methane[1]	4,178,504	18	FFT (average)	2.18×10^8	-	-	-
			FFT (best)	1.60×10^8	-	-	-
			CPLEX+Q	Can not be solved			
			BB+CF	1.04×10^8	1220	45.08	14400
			BB+CF+FBBT	$\mathbf{1.02 \times 10^8}$	**33**	**$\leq$0.1**	**679**
Gas_CO[2]	4,208,261	18	FFT (average)	1.28×10^9	-	-	-
			FFT (best)	8.81×10^8	-	-	-
			CPLEX+Q	Can not be solved			
			BB+CF	5.66×10^8	1095	14.66	14400
			BB+CF+FBBT	$\mathbf{5.46 \times 10^8}$	**62**	**$\leq$0.1**	**878**
kddcup[2]	4,898,431	38	FFT (average)	4.71×10^{17}	-	-	-
			FFT (best)	4.71×10^{17}	-	-	-
			CPLEX+Q	Can not be solved			
			BB+CF	2.25×10^{17}	63	$\leq$0.1	2461
			BB+CF+FBBT	$\mathbf{2.25 \times 10^{17}}$	**37**	**$\leq$ 0.1**	**928**
HIGGS[2]	11,000,000	29	FFT (average)	463.41	-	-	-
			FFT (best)	368.35	-	-	-
			CPLEX+Q	Can not be solved			
			BB+CF	247.03	368	211.88	14400
			BB+CF+FBBT	**237.91**	**290**	**189.90**	**14400**
BigCross[2]	11,620,300	56	FFT (average)	2.27×10^7	-	-	-
			FFT (best)	1.43×10^7	-	-	-
			CPLEX+Q	Can not be solved			
			BB+CF	1.09×10^7	148	48.99	14400
			BB+CF+FBBT	$\mathbf{9.97 \times 10^6}$	**211**	**24.61**	**14400**
Phones-accelerometer[1]	13,062,475	6	FFT (average)	4.48×10^{28}	-	-	-
			FFT (best)	2.04×10^{28}	-	-	-
			CPLEX+Q	Can not be solved			
			BB+CF	1.46×10^{28}	51	$\leq$0.1	2038
			BB+CF+FBBT	$\mathbf{1.46 \times 10^{28}}$	**i**[3]	**$\leq$0.1**	**276**
Phones-gyroscope[1]	13,932,632	6	FFT (average)	4.51×10^{28}	-	-	-
			FFT (best)	2.03×10^{28}	-	-	-
			CPLEX+Q	Can not be solved			
			BB+CF	1.46×10^{28}	51	$\leq$0.1	2195
			BB+CF+FBBT	$\mathbf{1.46 \times 10^{28}}$	**i**[3]	**$\leq$0.1**	**305**
AADP[2]	14,057,567	3	FFT (average)	3900.94	-	-	-
			FFT (best)	3824.00	-	-	-
			CPLEX+Q	Can not be solved			
			BB+CF	2660.10	602	55.91	14400
			BB+CF+FBBT	**2546.92**	**196**	**$\leq$0.1**	**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] Solved at the root node.

Acknowledgement

We gratefully acknowledge the computing resources provided by SciNet (www.scinethpc.ca) and Compute Canada (www.computecanada.ca).

References

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

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

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

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

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

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

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

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

Cplex, I. I. V20.1.0: User's Manual for CPLEX. *International Business Machines Corporation*, 2020.

Daskin, M. S. 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.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Duong, K.-C., Vrain, C., et al. Constrained clustering by constraint programming. *Artificial Intelligence*, 244:70–94, 2017.

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

Garey, M. R. and Johnson, D. S. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

Gonzalez, T. F. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38: 293–306, 1985.

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

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

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

Hua, K., Shi, M., and Cao, Y. A Scalable Deterministic Global Optimization Algorithm for Clustering Problems. In *International Conference on Machine Learning*, pp. 4391–4401. PMLR, 2021.

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

Kleindessner, M., Awasthi, P., and Morgenstern, J. Fair k-center clustering for data summarization. In *International Conference on Machine Learning*, pp. 3448–3457. PMLR, 2019.

Madhulatha, T. S. An overview on clustering methods. *arXiv preprint arXiv:1205.1117*, 2012.

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

Optimization, G. Gurobi optimizer 9.0 reference manual, 2020.

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

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

Wang, E., Ballachay, R., Cai, G., Cao, Y., and Trajano, H. Machine learning for biorefining: Towards a universal kinetic model of wood deconstruction. *Under Review*, 2022.

A. An illustrative example of branch and bound scheme with FBBT

The following is an example of k-center clustering solved by branch and bound scheme in Algorithm 2. This example have 6 samples in 2-dimensional space and we need to divide the samples into 2 clusters. Denote the dataset $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, with x_1=$(-1, 1)$, x_2=$(-1, 0)$, x_3=$(0, 0)$, x_4=$(2, 0)$, x_5=$(3, 0)$, and x_6=$(4, 0)$.

STEP 1: Compute initial upper bound. Use FFT method (Algorithm 1) to find 2 points x_1=$(-1, 1)$ and x_5=$(3, 0)$. With x_1 and x_5 as centers, we can compute the initial upper bound $\alpha = 2$. Note here FFT may return a different set of points depending on the selection of initial points. We normally repeat FFT for several trials and select the one with the best α.

STEP2.1: Assign initial seeds. Since the two points returned by FFT satisfying $||x_1 - x_5||_2^2 = 17 > 4\alpha$, then we can conclude that each point belongs to a distinct cluster. We can arbitrarily assign x_1 to cluster 1 and x_5 to cluster 2.

STEP 2.2: Sample based Assignment. Since $||x_2 - x_5||_2^2 = 16 > 4\alpha$, then x_2 and x_5 are not in the same cluster. Given x_5 was assigned to cluster 2, we have x_2 in cluster 1. Similarly, we have x_3 in cluster 1, x_4 and x_6 in cluster 2. Note here the assignment of all samples are determined at the root node.

STEP 3: Ball-based bounds tightening. Since x_4 and x_6 are in cluster 2, using ball-based bounds tightening as illustrated in Figure 6, $M^2 = \{(3, 0)\}$, i.e., the center of cluster 2 is $(3, 0)$. For cluster 1, we can only attain $M^1 = \{\mu^1 | -1 \leq \mu_1^1 \leq 0, 0 \leq \mu_2^1 \leq 1\}$.

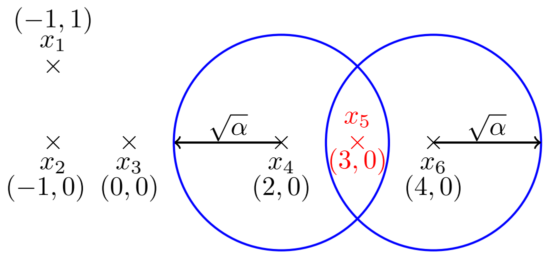

Figure 6. Ball-based bounds tightening for cluster 2.

STEP 4: Compute initial lower bound. Following Equation 7, we can get $\beta = 1$.

STEP 5: Branch and bound.

ITERATION 1: For cluster 1, find two subsets $M_1^1 = \{\mu^1 | -1 \leq \mu_1^1 \leq 0.5, 0 \leq \mu_2^1 \leq 1\}$ and $M_2^1 = \{\mu^1 | -0.5 \leq \mu_1^1 \leq 0, 0 \leq \mu_2^1 \leq 1\}$.For the first child node, $M_1^1 \cap X = \{x_1, x_2\}$. Randomly pick x_2 as the center for cluster 1 and we have $\alpha = 1$.

With $\alpha = \beta = 1$, we can conclude that the optimal value is 1 and the optimal centers are x_2 and x_5.