Global Optimal K-Medoids Clustering of One Million Samples

Jiayang Ren[†*], **Kaixun Hua**[†*], **Yankai Cao**[†‡]
[†]University of British Columbia
[‡]yankai.cao@ubc.ca

Abstract

We study the deterministic global optimization of the K-Medoids clustering problem. This work proposes a branch and bound (BB) scheme, in which a tailored Lagrangian relaxation method proposed in the 1970s is used to provide a lower bound at each BB node. The lower bounding method already guarantees the maximum gap at the root node. A closed-form solution to the lower bound can be derived analytically without explicitly solving any optimization problems, and its computation can be easily parallelized. Moreover, with this lower bounding method, finite convergence to the global optimal solution can be guaranteed by branching only on the regions of medoids. We also present several tailored bound tightening techniques to reduce the search space and computational cost. Extensive computational studies on 28 machine learning datasets demonstrate that our algorithm can provide a provable global optimal solution with an optimality gap of 0.1% within 4 hours on datasets with up to one million samples. Besides, our algorithm can obtain better or equal objective values than the heuristic method. A theoretical proof of global convergence for our algorithm is also presented.

1 Introduction

In this paper, we concentrate on the K-Medoids clustering problem: given a dataset, K-Medoids aims to select K samples from the dataset as cluster medoids that minimize the sum of dissimilarities from other samples to the closet medoids. Here, selecting from existing samples as medoids is called the "medoids on samples" constraint. K-Medoids is similar to the K-Means problem except for the arbitrary dissimilarity measurements and the "medoids on samples" constraint. Hence, K-Medoids generally has better interpretability than K-Means [1]. K-Medoids problems are generally NP-hard to solve exactly. So there are many heuristic methods, such as PAM [2], CLARA [3], K-means-like method [4], and Fast-PAM [5]. However, none of them can guarantee reaching the global minimum.

To solve the K-Medoids problem deterministically, back in the 1970s, Cornuejols et al. [6] proposed a Lagrangian relaxation method for obtaining a lower bound of the optimal value, which can guarantee a relative optimal gap smaller than $1/e$. Interestingly, to the best of our knowledge, this is the only application in which the Lagrangian duality gap can be guaranteed within a certain threshold. Several improvements to this method are proposed in the literature. One common direction is improving the heuristic process of Lagrangian dual problems. [7] constructed a semi-Lagrangian relaxation, then accelerated the heuristic process by a cutting plane method. It was further improved in [8] by fixing variables in the heuristic process. Their works can deal with medium-scale datasets (e.g., 3000 samples). Another common direction is column generation [9, 10].

One limitation of these Lagrangian relaxation based methods is that there is no guarantee of convergence to the global optimal solution, due to the existence of duality gap. To address the global

[*]Jiayang Ren (rjy12307@mail.ubc.ca) and Kaixun Hua (kaixun.hua@ubc.ca) contributed equally.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

optimum guarantee, a few works have utilized the branch and bound (BB) scheme in K-Medoids [11, 12, 13]. Christofides and Beasley [11] proposed a prototype by branching on binary variables and was capable of small-scale datasets (200 samples). This scheme was further improved by [12] through branching on semi-assignment constraints, which obtained solutions faster on small-scale datasets. [13] presented a tighter form of the K-Medoids problem and was capable of medium-scale datasets (900 samples). Besides these specific works, several off-the-shelf solvers also contain well-implemented BB schemes, such as CPLEX [14] and Gurobi [15]. These solvers can typically address the K-Mediods problem with medium-scale datasets (e.g., 2000 samples).

The BB schemes used in the aforementioned works can only address medium-scale datasets because they need to branch on all the binary variables to guarantee convergence, which leads to poor scalability because the number of binary variables increases linearly as the number of samples. Therefore, we focus on the reduced-space BB scheme proposed in the stochastic programming community, which only needs to branch on the space of coupling variables (e.g., medoids). This idea was first reported for stochastic mixed-integer linear programs in [16] and then extended to stochastic mixed-integer nonlinear programs in [17]. The convergence for stochastic nonlinear programs was proved in [18]. Recently, the authors in [19] adopted this method and reported the capacity of solving the K-means clustering problem (a similar clustering task without the "medoids on samples" constraint) with 210,000 samples in parallel to an optimal gap of 2.5%. However, the convergence of the reduced-space BB scheme can only be guaranteed when it is combined with suitable lower bounding methods. In previous works, lower bounds are constructed by the Lagrangian relaxation [16] [17] [19] or removal [18] of the so-called non-anticipativity constraints (an implicit constraint that all the samples share the same coupling variables, that is Constraint 10 in this paper). However, these methods are unsuitable for the K-Medoids problem, since they cannot address the "medoids on samples" constraint (a constraint that has a large dimension and connects all samples) and thus cannot ensure the convergence of the reduced-space BB scheme. Moreover, these lower bounding methods require the solution of many expensive mixed-integer nonlinear programs to global optimality at each node. In contrast, the work of Cornuejols et al. [6] generates lower bounds for the K-Medoids problem by the Lagrangian relaxation of the semi-assignment constraint (Constraint 2b in this paper), and the formulation directly takes the "medoids on samples" constraint into consideration. This method has a closed-form solution, and its computation can be easily parallelized.

Our Contributions: Firstly, we combine the method of reduced-space BB scheme and the work of Cornuejols et al. [6], and adapt these methods for the K-Medoids problem. The Lagrangian relaxation in [6] is used to generate a tight lower bound, which already guarantees a maximum gap at the root node. With this lower bound, we can prove the finite convergence (detailed proofs presented in the Appendix A) by branching only on the regions of medoids (the dimension of which is independent of the number of samples). Secondly, several tailored bound tightening techniques, including probing and feasibility based methods, are proposed to significantly reduce the search space and computational cost of each node. Lastly, we propose an upper bounding method by heuristic updates of the candidate solutions obtained in the lower bounding procedure. These contributions enable our algorithm to be extremely scalable.

Capability For Million Scale Problems: we provide an open-source project in Julia and present numerical experiments on 28 datasets. Specifically, for two datasets with one million samples (which is **400 times larger** than the state-of-art BB method), our algorithm can reach a small gap (0.1%) within one hour. To the best of our knowledge, no other article has ever reported results on datasets of this scale for K-Medoids problems with a global convergence guarantee.

2 Preliminaries

K-Medoids problem: we denote a dataset with S samples and A attributes as $X = \{x_1, ..., x_s, ..., x_S\} \in \mathbb{R}^{S\times A}$, in which $x_s = [x_{s,1}, ..., x_{s,A}] \in \mathbb{R}^A$ is the sth sample, and $x_{s,a}$ is the ath attribute of sth sample. The K-Medoids problem aims to select a subset from the dataset as the cluster medoids with the following objective:

$$\min_{\mu \in \overset{\circ}{M} \cap X} \sum_{s\in\mathcal{S}} \min_{k\in\mathcal{K}} ||x_s - \mu^k||_2^2, \tag{1}$$

where $\mu^k := [\mu_1^k, ..., \mu_A^k] \in \mathbb{R}^A$ is the medoid of kth cluster, $\mu := \{\mu^1, ..., \mu^K\}$ is the medoids set, $\mathcal{S} := \{1, ..., S\}$ is the index set of samples in the dataset, $\mathcal{K} := \{1, ..., K\}$ is the index set of clusters,

$\mathcal{A} := \{1, ..., A\}$ is the index set of attributes. We use $\mu \in X$ to represent the "medoids on samples" constraint in the K-Medoids problem. To facilitate the discussion of the BB scheme, we introduce $\overset{\circ}{M} := \{\mu | \overset{\circ}{\underline{\mu}} \leq \mu \leq \overset{\circ}{\bar{\mu}}\}$ as the initial regions of medoids. Here, $\overset{\circ}{\underline{\mu}}$ and $\overset{\circ}{\bar{\mu}}$ are the lower and upper bounds of medoids referred from data. Specifically, $\overset{\circ}{\underline{\mu}}_a = \min_{s \in S} x_{s,a}$ and $\overset{\circ}{\bar{\mu}}_a = \max_{s \in S} x_{s,a}$ are the bounds of the ath attribute. Note that the introduction of $\overset{\circ}{M}$ does not affect the optimal solution.

Equivalently, the K-Medoids problem can also be expressed in an extensive form (EF) [20]:

$$\text{(EF)} \quad z(\overset{\circ}{M}) = \min_{b,y} \sum_{s \in \mathcal{S}} \sum_{j \in \mathcal{S}} d_{s,j} b_{s,j} \tag{2a}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{S}} b_{s,j} = 1 \tag{2b}$$

$$\sum_{j \in \mathcal{S}} y_j = K \tag{2c}$$

$$b_{s,j} \leq y_j \tag{2d}$$

$$b_{s,j}, y_j \in \{0, 1\} \tag{2e}$$

$$s, j \in \mathcal{S},\ k \in \mathcal{K}, \tag{2f}$$

where $d_{s,j}$=$||x_s - x_j||_2^2$ is the distance between samples x_s and x_j; y_j is equal to 1 if sample x_j is a medoid, and otherwise 0; $b_{s,j}$ is equal to 1 if sample x_s belongs to the cluster whose medoid is x_j, and otherwise 0. Note here $d_{s,j}$ is computed offline, so arbitrary dissimilarity measurements can be used without affecting the solution process. Constraint 2b ensures that each sample x_s must belong to one cluster. Constraint 2c guarantees that a total of K samples are selected as medoids. Constraint 2d expresses that sample x_s can be assigned to sample x_j only if x_j is the medoid of a cluster.

Lagrangian Relaxation of the K-Medoids problem: by the Lagrangian relaxation of the Constraint 2b, we can have a relaxed problem of the Extensive Form 2:

$$\begin{aligned} \beta_{LD}(\overset{\circ}{M}, \lambda) = \min_{b,y} \{ \sum_{s \in S} [\sum_{j \in S} (d_{s,j} - \lambda_s) b_{s,j} + \lambda_s] \} \\ \text{s.t.} \quad \text{Constraints in the Extensive Form 2 except 2b,} \end{aligned} \tag{3}$$

where λ_s is the Lagrangian multiplier for sth sample. We define the multiplier set as $\lambda := \{\lambda_1, ..., \lambda_S\}$. For a given λ, $\beta_{LD}(\overset{\circ}{M}, \lambda)$ can be solved analytically. Considering the objective function and constraints of the Relaxed Problem 3, we note that the optimal values for $b_{s,j}$ are given by

$$b_{s,j} = \begin{cases} y_j, & \text{if } d_{s,j} - \lambda_s \leq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

We define the contribution of jth sample to the objective function as:

$$\rho_j(\lambda) := \sum_{s \in S} \min(0, d_{s,j} - \lambda_s). \tag{5}$$

The Lagrangian problem can be reduced to

$$\begin{aligned} \beta_{LD}(\overset{\circ}{M}, \lambda) = \min_{y_j \in \{0,1\}, j \in \mathcal{S}} \{ \sum_{j \in S} \rho_j(\lambda) y_j + \sum_{s \in S} \lambda_s \} \\ \text{s.t.} \quad \sum_{j \in \mathcal{S}} y_j = K \end{aligned} \tag{6}$$

Its closed-form solution can be easily derived by selecting y_j corresponding to the K smallest ρ_j.
Lagrangian dual: a proper selection of λ may generate a tighter lower bound. Hence, we need to solve the Lagrangian dual problem to get the tightest lower bound:

$$\beta_{LD}(\overset{\circ}{M}) = \max_{\lambda} \beta_{LD}(\overset{\circ}{M}, \lambda). \tag{7}$$

There are several off-the-shelf heuristic methods to update λ, such as Sub-gradient method [11], Volume method [21] and Cutting-Plane method [22]. The solution to the Dual Problem 7 provides a lower bound to the original problem, with $\beta_{LD}(\overset{\circ}{M}) \leq z(\overset{\circ}{M})$. Particularly, for this problem, there is a unique guarantee of maximum gaps between the lower bounds and the global optimal value, which is $\frac{z(M) - \beta_{LD}(\overset{\circ}{M})}{z_r - \beta_{LD}(\overset{\circ}{M})} < \frac{1}{e}$, $z_r = \sum_{s \in \mathcal{S}} \max_{j \in \mathcal{S}} d_{s,j}$. This guarantee is theoretically proved in [6, 23].

3 Lagrangian Based Lower Bound

The aforementioned Lagrangian method cannot close the duality gap. To obtain a global optimal solution, we propose to combine it with the spatial branch and bound (BB) scheme, in which Lagrangian relaxation provides a lower bound for each BB node.

Tailored Lagrangian Relaxation in BB nodes: Since our BB scheme branches on the regions of the medoids μ, at each BB node, we solve the problem with the partition set $M \subseteq \overset{\circ}{M}$. That is, a constraint $\mu \in M$ is added at each BB node. We further define M^k:=$\{\mu^k | \underline{\mu}^k \leq \mu^k \leq \bar{\mu}^k\}$ as the medoid region of the kth cluster, where $\underline{\mu}^k$ and $\bar{\mu}^k$ are the lower and upper bounds of the kth medoids. Since the possible region of each cluster's medoid can be different, we need to know the assignment of medoids to each cluster. Hence, we replace y_j with y_j^k in EF 2: y_j^k is equal to 1 if sample x_j is selected as the medoid of the kth cluster, and otherwise 0. The Constraint 2d is replaced by

$$\sum_{j \in \mathcal{S}} y_j = K \quad \text{(2d)} \quad \Longrightarrow \quad \sum_{j \in \mathcal{S}} y_j^k = 1 \quad \text{(2d.1)}, \quad \sum_{k \in \mathcal{K}} y_j^k \leq 1 \quad \text{(2d.2)}.$$

Notice that μ^k is not directly modelled in EF, we need to transform the medoid regions $\mu^k \in M_k$ to the constraints of y_j^k. The regions of medoids implicitly control the selection of y_j^k because we have $y_j^k = 0$, if $x_j \notin M^k$. Hence, we define the feasible index set of kth cluster's medoid as $\mathcal{S}^{k+}(M) := \{j \in \mathcal{S} \mid \underline{\mu}^k \leq x_j \leq \bar{\mu}^k\}$. In this way, we can reformulate the reduced Lagrangian Problem 6 at the BB node with M as follow:

$$\begin{aligned} \beta_{LD}(M, \lambda) = & \min_{y_j^k \in \{0,1\}, k \in \mathcal{K}, j \in \mathcal{S}^{k+}(M)} \{\sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{S}^{k+}(M)} \rho_j(\lambda) y_j^k + \sum_{s \in \mathcal{S}} \lambda_s\} \\ & \text{s.t.} \quad \text{Constraint (2d.1), (2d.2).} \end{aligned} \tag{8}$$

with the Lagrangian dual problem defined as $\beta_{LD}(M) = \max_{\lambda} \beta_{LD}(M, \lambda)$. The optimal value of y_j^k in Problem 8 can be obtained by the following procedure: (1) initialize $k = 0$ and $J^* = \emptyset$ (2) let j^{k*} be the index of smallest $\rho_j(\lambda)$ subject to $j \in \mathcal{S}^{k+}(M)$ and $j \notin J^*$; (3) set $y_j^k = 1$ if $j = j^{k*}$, otherwise, 0; (4) add j^{k*} to J^*; (5) update k=$k+1$ and return to (2) until $k = K$.

Parallel implementation: we implement a parallel version of the tailored Lagrangian relaxation, in which the computations of contributions, $\rho_j(\lambda)$, are performed in each process. At the beginning of the BB procedure, given P processes, the index set $\mathcal{S}$ of the whole dataset is evenly divided into P subsets, $\mathcal{S}_p, p = 1, .., P$. Then, each process computes and stores the distance assigned to it, which is $d_{s,j}, s \in \mathcal{S}, j \in \mathcal{S}_p$. At each BB node with M, denote the feasible index set of all the medoids as

$$\mathcal{S}^+(M) = \mathcal{S}^{1+}(M) \cup \mathcal{S}^{2+}(M) \cup ... \cup \mathcal{S}^{K+}(M). \tag{9}$$

Then, each process computes the assigned contributions using the stored $d_{s,j}$, which is $\rho_j(\lambda), \forall j \in \mathcal{S}_p \cap \mathcal{S}^+(M)$. The remaining parts of the parallel implementation (e.g., update of λ) are identical to the serial version.

4 Tailored Reduced-space Branch and Bound Scheme

This section introduces the tailored reduced-space branch and bound algorithm for the K-Medoids problem, which is detailed in Algorithm 1. Specifically, starting from the root node with $\overset{\circ}{M}$, the BB scheme recursively partitions the medoid regions. At each iteration, the algorithm first selects the node with the lowest lower bound and denotes it as M. Then, bound tightening techniques are applied to get a tightened medoid region $\hat{M}$. The lower and upper bounds of the tightened node are computed and updated sequentially. If the gap is larger than the tolerance, we branch on the medoid variable μ_a^k with the maximum range to get two sub-nodes M_1 and M_2 with $relint(M_1) \cap relint(M_2) = \emptyset$, where $relint(M)$ is the relative interior of M. On the convergence of the algorithm, we have the following theorem with detailed proofs in Appendix A:

Theorem 1. *Algorithm 1 is convergent to the global optimal solution after a finite step L, with $\beta_L = z = \alpha_L$, by only branching on the space of μ.*

Algorithm 1 Tailored Reduced-space Branch and Bound Scheme for K-Medoids clustering

Initialization
Load and store dataset as X.
Initialize the iteration index $l = 0$, the node set $\mathbb{M} \leftarrow \overset{\circ}{M}$, and tolerance $\epsilon \geq 0$.
Compute initial lower, upper bounds: $\beta_l = \beta(\overset{\circ}{M})$, $\alpha_l = \alpha(\overset{\circ}{M})$.
while $\mathbb{M} \neq \emptyset$ **and** $|\beta_l - \alpha_l| \geq \epsilon$ **do**
  Node Selection
  Select and delete from $\mathbb{M}$ a Node $M \in \mathbb{M}$ with its parent node's lower bound $\beta(M_P) = \beta_l$ (for the root node, select and delete itself).
  Update $l \leftarrow l + 1$.
  Bound Tightening
  Probing // Alg. 2.
  Cluster Assignment // Alg. 3.
  Feasibility Based Bound Tightening // Alg. 4.
  Tighten according to the "medoids on samples" constraint.
  Obtain the tightened node $\hat{M}$.
  Bounding
  Compute lower, upper bounds: $\beta(\hat{M})$, $\alpha(\hat{M})$.
  Update $\beta_l \leftarrow min\{\beta(M') \mid M' \in \mathbb{M}\}$.
  Update $\alpha_l \leftarrow min\{\alpha_{l-1}, \alpha(\hat{M})\}$.
  Delete all M' from $\mathbb{M}$ if $\beta(M') > \alpha_l$.
  Branching
  if $|\hat{M} \cap X| > 1$ **and** $|\beta_l - \alpha_l| \geq \epsilon$ **then**
    Partition $\hat{M}$ into subsets M_1 and M_2 with $relint(M_1) \cap relint(M_2) = \emptyset$.
    Add M_i to $\mathbb{M}$ if $M_i^k \cap X \neq \emptyset, \forall k \in \mathcal{K}, i \in \{1, 2\}$.
  end if
end while

Algorithm 2 Probing

Select the region of the branched variable M_a^k.
Divide M_a^k into two sub-regions M_{sub}.
Compute basic LB for each sub-region.
Delete the sub-region if $\beta_{basic}(M_{sub}) > \alpha_l$.

Algorithm 3 Cluster Assignment

for Sample $x_s \in X \cap M$, **do**
  if x_s is not assigned in the parent node **then**
    if $d_{s,max}^{k^+} < d_{s,min}^{k^-}, \forall k^- \in \mathcal{K} \setminus k^+$ **then**
      Assign x_s to k^+th cluster.
    end if
  end if
end for

Algorithm 4 Feasibility Based Bound Tightening

for Cluster $k \in \mathcal{K}$ **do**
  Obtain the assigned set $\mathcal{S}_A^k$ according to Alg.3.
  Compute $RHS = \alpha_l - \beta(M)^-$.
  for Attribute $a \in \mathcal{A}$ **do**
    Obtain the tightened node, $\hat{M}_a^k = \{\mu_a^k | \hat{\underline{\mu}}_a^k \leq \mu_a^k \leq \hat{\bar{\mu}}_a^k\}$, by solving the inequation 13.
    Update $\hat{M}_a^k \leftarrow \hat{M}_a^k \cap M_a^k$
  end for
end for

Algorithm 5 Lagrangian Based Lower Bounding

Compute basic LB, $\beta_{basic}(\hat{M})$.
if $|\beta_{basic}(\hat{M}) - \alpha_l| \geq \epsilon$ **then**
  Compute Lagrange Dual LB, $\beta_{LD}(\hat{M})$.
  $\beta(\hat{M}) = max\{\beta_{basic}(\hat{M}), \beta_{LD}(\hat{M})\}$.
else
  $\beta(\hat{M}) = \beta_{basic}(\hat{M})$.
end if

4.1 Lower Bounds

Basic lower bound: Besides the Lagrangian based lower bound, we also introduce a basic lower bound β_{basic}. Although it is not as tight as the Lagrangian based lower bound β_{LD}, its computational cost is significantly lower. Thus it is useful when the BB tree is deep with small medoid regions. For K-Medoids 1, there is an implicit constraint that all the samples share the same medoid set:

$$\mu_s^k = \mu_{s+1}^k, \forall s \in \{1, ..., S-1\}, k \in \mathcal{K}, \tag{10}$$

in which μ_s^k is the kth medoid for the sth sample. By removing the "medoids on samples" constraint $\mu \in X$ and Constraint 10, we have the basic lower bound $\beta_{basic}(M)$ with $\beta_{basic}(M) \leq z(M)$:

$$\beta_{basic}(M) = \min_{\mu_s \in M} \sum_{s \in S} \min_{k \in \mathcal{K}} ||x_s - \mu_s^k||_2^2 = \sum_{s \in S} \min_{k \in \mathcal{K}} \min_{\mu_s^k \in M^k} ||x_s - \mu_s^k||_2^2. \tag{11}$$

We define $d_{s,min}^k(M) := \min_{\mu^k \in M^k} ||x_s - \mu^k||_2^2$ and obviously $\beta_{basic}(M) = \sum_{s \in S} \min_{k \in \mathcal{K}} d_{s,min}^k(M)$. The analytic solution to $d_{s,min}^k(M)$ can be easily derived: $\mu_{s,a}^k = \text{mid}\{\underline{\mu}_a^k,\ x_{s,a},\ \bar{\mu}_a^k\}, a \in \mathcal{A}$, in which $\underline{\mu}_a^k$ and $\bar{\mu}_a^k$ are the bounds of the ath attribute in kth medoid region.

Lagrangian based lower bound: as shown in Algorithm 5, by combining the basic and LD lower bound methods, we give the final form of lower bounds: $\beta(M) = max\{\beta_{basic}(M), \beta_{LD}(M)\}$. We note that because $\beta_{basic}(M) \leq z(M)$ and $\beta_{LD}(M) \leq z(M)$, we have $\beta(M) \leq z(M)$. Besides, according to the guarantee of maximum gap between the Lagrangian based lower bound and the global optimal value, we can also derive a guarantee of maximum gaps for $\beta(M)$:

Proposition 2. $\frac{z(M)-\beta(M)}{z_r-\beta(M)} < \frac{1}{e}$, *where* $z_r = \sum_{s \in \mathcal{S}} \max_{j \in \mathcal{S}} d_{s,j}$.

4.2 Upper Bounds

Upper bounds of the K-Medoids problem can be obtained by selecting a feasible set of medoids $\tilde{\mu} \in M \cap X$. Given a feasible solution, we can obtain the assignments of samples and object value through Equation 1, without solving any optimization problem explicitly. However, the selection of $\tilde{\mu}$ will critically influence the efficiency of convergence. Hence, in our implementation, we introduce two efficient upper bounding methods:

Root node: we use the K-Means-like method [4] as a backbone. Because the K-Means-like method is sensitive to the initial seeds and apt to the local minimum [5], we run this method several times with random seeds to get a candidate solution set. Then, we improve the quality of the upper bound by feeding the candidate solution set as initial guesses to Evolutionary Centers Algorithm [24].

Child nodes: we utilize the candidate solution obtained in the lower bounding method to update the upper bounds. Specifically, we first compute the corresponding assignments, then feed these assignments as the initial solution to the K-Means-like method. It should be noted that this K-Means-like method [4] gives a solution full-filling the "medoids on samples" constraint in K-Medoids.

5 Bound Tightening Techniques

Here, we introduce some bound tightening techniques tailored for the K-Medoids problem to reduce the search regions M at each BB node while guaranteeing the optimal solution is not excluded. These bound tightening techniques are deployed before lower and upper bounds calculation.

5.1 Probing

Probing is a technique to reduce bound regions by exploiting the inequalities in a mixed integer programming problem [25]. This paper exploits the implicit inequality that the lower bound of a region containing the optimal solution is smaller than the current best upper bound. As shown in Algorithm 2, it tentatively restricts the bounds of μ to a subinterval and then computes the corresponding lower bound. If the lower bound is larger than the best upper bound, we can conclude that no optimal solution exists within the subinterval, and tighten the bound on the variable accordingly. The computation costs are trivial since we use the basic lower bounding method.

5.2 Feasibility Based Bound Tightening

This subsection introduces the Feasibility Based Bound Tightening (FBBT) technique. It is based on the observation that the assignment of some samples to clusters can be predetermined.

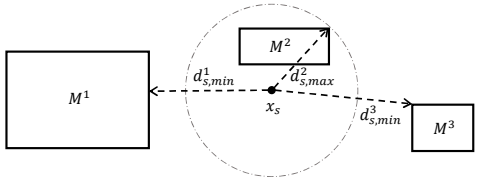

Figure 1: Cluster assignment with 3 clusters. In this example, we have $d^2_{s,max} < d^1_{s,min}$ and $d^2_{s,max} < d^3_{s,min}$. Therefore, we assign x_s to the second cluster.

Cluster assignment: if we have $||x_s - \mu^{k^+}||_2^2 < ||x_s - \mu^{k^-}||_2^2, \forall k^- \in \mathcal{K} \setminus k^+$, then obviously sample x_s is in the k^+th cluster. However, the value of μ here is not known before solving the overall problem. Recall that we have $d^k_{s,min}(M) := \min_{\mu^k \in M^k} ||x_s - \mu^k||_2^2$. Similarly, we also define $d^k_{s,max}(M) := \max_{\mu^k \in M^k} ||x_s - \mu^k||_2^2$ as the maximum distance between x_s and μ_k with $\mu^k \in M^k$. At a BB node with partition M, if we have $d^{k^+}_{s,max}(M) < d^{k^-}_{s,min}(M), \forall k^- \in \mathcal{K} \setminus k^+$, then sample x_s is guaranteed to be assigned to the k^+th cluster. Based on this observation, we can check whether the cluster of some samples can be determined in a BB node as shown in Algorithm 3. Recall that $d^k_{s,min}(M)$ is already computed as a byproduct of the basic lower bound in Section 3. Similarly, since the cluster regions for each node are rectangles, $d^k_{s,max}(M)$ can also be analytically computed. Besides, since child nodes are subsets of the parent node, once a sample is assigned to a cluster in

a parent node, it is also assigned to the same cluster in its child nodes. Figure 1 shows a simple illustration of this cluster assignment scheme.

FBBT: In a node with M, we denote the index sets of samples assigned to cluster k as $\mathcal{S}_A^k$. FBBT infers the regions of μ^k from the following inequation:

$$\beta(M)^- + \sum_{s \in \mathcal{S}_A^k} ||x_s - \mu^k||_2^2 \leq \alpha_l, \forall k \in \mathcal{K} \tag{12}$$

where α_l is the current best upper bound, $\beta(M)^-$ represents the lower bound contributions of samples unassigned to cluster k (either samples assigned to other clusters or samples whose assignments are undetermined). Similar to the calculation of the basic lower bounding method, we have $\beta(M)^- = \sum_{s \in \mathcal{S} \setminus \mathcal{S}_A^k} \min_{k \in \mathcal{K}} d_{s,min}^k(M)$.

As shown in Algorithm 4, FBBT is applied for each cluster. Specifically, for the kth medoid region, 1) obtain $\mathcal{S}_A^k$ according to the results of Algorithm 3; 2) compute the right-hand-side, $RHS = \alpha_l - \beta(M)^-$; 3) for each attribute $a \in \mathcal{A}$, obtain the tightened region, $\hat{M}_a^k = \{\mu_a^k | \hat{\underline{\mu}}_a^k \leq \mu_a^k \leq \hat{\bar{\mu}}_a^k\}$, by solving the following inequation:

$$\sum_{s \in \mathcal{S}_A^k} (x_{s,a} - \mu_a^k)^2 + \sum_{s \in \mathcal{S}_A^k} \sum_{i \in \mathcal{A} \setminus a} \min_{\mu_i^k \in M^k} (x_{s,i} - \mu_i^k)^2 \leq RHS, \quad \forall a \in \mathcal{A}, \forall k \in \mathcal{K}. \tag{13}$$

Since 13 is a simple quadratic inequation, we can analytically solve the problem to obtain the lower and upper bound of μ_a^k as $\hat{\underline{\mu}}_a^k$ and $\hat{\bar{\mu}}_a^k$. Finally, update $\hat{M}_a^k$ by $\hat{M}_a^k \leftarrow \hat{M}_a^k \cap M_a^k$.

5.3 Miscellaneous

Symmetric-breaking: we enforce a symmetric-breaking constraint on the first attribute, i.e., $\mu_1^k \leq \mu_1^{k+1}, \forall k = 1, ..., K-1$. It is used to tighten the regions of M at each BB node.

Bound tightening according to the "medoids on samples" constraint: since the K-Medoids problem requires medoids to be selected from samples in the dataset, we can further tighten the regions of M by $\hat{M} = \{\mu | \hat{\underline{\mu}} \leq \mu \leq \hat{\bar{\mu}}\}$, $\hat{\underline{\mu}}_a^k = \min_{s \in M \cap X} x_{s,a}$, and $\hat{\bar{\mu}}_a^k = \max_{s \in M \cap X} x_{s,a}$.

5.4 Effects on Computation

With a reduced search space, these bound tightening techniques can significantly reduce the number of BB nodes to be explored. Moreover, they can also accelerate the calculation of bounds. For example, for the Lagrangian Problem 8, we need to calculate the contributions, ρ_j, of samples belonging to the overall feasible set of medoids, $\hat{S}^+ = \hat{S}^{1+} \cup \hat{S}^{2+} \cup ... \cup \hat{S}^{K+}$. A reduced search space $\hat{M}$ means a reduced overall feasible selection set, which further leads to fewer calculations and faster speeds.

6 Experiments and Discussions

To evaluate the performance, we use 25 datasets from the UCI Machine Learning Repository [26], one dataset called PR2392 from [27], one dataset called HEMI[28] and one synthetic dataset from [19]. The datasets from UCI Repository are in the clustering category with multivariate numerical attributes and range from 100 to 2,458,285 instances. We implement our branch and bound K-Medoids algorithm with Lagrangian based lower bound method (BB+LD) in Julia 1.6.1. We also implement four other methods to compare the performance, including the extensive form of K-Medoids problem solved by the state-of-art global optimizer CPLEX 20.1.0 [14] (CPLEX), the heuristic method in the root node (Heuristic), the pure Lagrangian relaxation method without branch and bound (LD), and the state-of-art BB method for K-Means in [19] adapting to K-Medoids by adding "medoids on samples" constraint (BB+Basic). We executed all the experiments on a high-performance computing cluster, of which each node contains 40 Intel cores at 2.4 GHz and 202 GB RAM. We present the results of cluster number $K = 3$ in the main body with additional results $K = 5$ and $K = 10$ in Appendix C. The complete code files can be found in `https://github.com/YankaiGroup/global_kmedoids_clustering`

The numerical performances are compared using three indicators, including upper bound value (UB), relative optimal gap, and the number of solved nodes. Here, UB is the best upper bound in all the iterations. The relative optimal gap is the relative difference between the best lower and upper bounds, which is calculated by $Gap(\%) := \frac{\alpha_l - \beta_l}{\min\{\alpha_l, \beta_l\}} \times 100\%$. This relative gap represents the worst gap of the found solution from the global optimal solution. Notably, this gap is a particular property of deterministic global optimization algorithms. Heuristic methods can not quantify the worst gap from the global optimal solution. The number of solved nodes represents the total number of BB nodes.

All the deterministic methods CPLEX, LD, BB+Basics, and BB+LD, share the same terminal criteria, including (1) $Gap \leq 0.1\%$, (2) the solving time reaches 4 hours, (3) the number of solved nodes reaches 5 million. Both LD and BB+LD use the same sub-gradient method called the Volume method [21]. Particularly, for BB+LD, the maximum iteration of Lagrangian multiplier updates in each node is 10. For LD, the maximum iteration is 10,000. In the heuristic method, we execute the K-Means-like method 20 times with a fixed seed set including 20 different seeds, then feed the solution set as initial guesses to ECA to get the best solutions.

Table 1: Serial numerical results ($K = 3$)

Data-set	Sam-ple	Dim-ension	Method	UB	Nodes	Gap (%)	Time (s)	Data-set	Sam-ple	Dim-ension	Method	UB	Nodes	Gap (%)	Time (s)
			Heuristic	84.63	-	-	-				Heuristic	9.91E+06	-	-	-
			CPLEX	83.91	1	≤0.10	41				CPLEX	9.91E+06	1	≤0.10	2044
iris	150	4	**LD**	**83.91**	**1**	**≤0.10**	**19**	hemi	1955	7	LD	9.91E+06	1	12.45	3176
			BB+Basic	83.91	2.0E+5	≤0.10	124				BB+Basic	9.91E+06	1.9E+6	4.32	4h
			BB+LD	83.91	25	≤0.10	93				**BB+LD**	**9.91E+06**	**63**	**≤0.10**	**97**
			Heuristic	598.29	-	-	-				Heuristic	2.13E+10	-	-	-
			CPLEX	598.29	1	≤0.10	60				CPLEX	2.13E+10	1	≤0.10	5339
seeds	210	7	**LD**	**598.29**	**1**	**≤0.10**	**19**	pr2392	2392	2	LD	2.13E+10	1	≤0.10	35
			BB+Basic	598.29	2.4E+5	≤0.10	273				BB+Basic	2.13E+10	3.0E+5	4.38	4h
			BB+LD	598.29	9	≤0.10	84				**BB+LD**	**2.13E+10**	**37**	**≤0.10**	**123**
			Heuristic	629.02	-	-	-				Heuristic	1.96E+05	-	-	-
			CPLEX	629.02	1	≤0.10	60				CPLEX	-	-	-	-
glass	214	9	**LD**	**629.02**	**1**	**≤0.10**	**19**	TRR	5456	24	LD	1.96E+05	1	0.41	4h
			BB+Basic	629.02	2.2E+6	≤0.10	2871				BB+Basic	1.96E+05	7.4E+4	1644.88	4h
			BB+LD	629.02	32	≤0.10	107				**BB+LD**	**1.96E+05**	**553**	**≤0.10**	**325**
			Heuristic	8.65E+05	-	-	-				Heuristic	2211.19	-	-	-
			CPLEX	8.63E+05	1	≤0.10	64				CPLEX	-	-	-	-
BM	249	6	**LD**	**8.63E+05**	**1**	**≤0.10**	**19**	AC	7195	22	LD	2199.12	1	≤0.10	287
			BB+Basic	8.63E+05	5.0E+6	25.74	7698				BB+Basic	2200.04	1.3E+5	573.92	4h
			BB+LD	8.63E+05	53	≤0.10	86				**BB+LD**	**2199.10**	**67**	**≤0.10**	**222**
			Heuristic	50.77	-	-	-				Heuristic	1.49E+07	-	-	-
			CPLEX	**50.77**	**1**	**≤0.10**	**74**				CPLEX	-	-	-	-
UK	258	5	LD	50.77	1	0.96	176	rds_cnt	10000	4	LD	1.49E+07	1	14.89	4h
			BB+Basic	50.77	9.5E+5	3.93	4h				BB+Basic	1.49E+07	2.6E+5	≤0.10	5883
			BB+LD	50.77	119	≤0.10	89				**BB+LD**	**1.49E+07**	**31**	**≤0.10**	**203**
			Heuristic	7.83E+11	-	-	-				Heuristic	8.21E+07	-	-	-
			CPLEX	**7.83E+11**	**1**	**≤0.10**	**74**				CPLEX	-	-	-	-
HF	299	12	LD	7.83E+11	1	19.73	28	HTRU2	17898	8	LD	8.21E+07	1	22.97	4h
			BB+Basic	7.83E+11	9.6E+4	≤0.10	173				BB+Basic	8.21E+07	2.3E+5	25.48	4h
			BB+LD	7.83E+11	99	≤0.10	107				**BB+LD**	**8.21E+07**	**465**	**≤0.10**	**1555**
			Heuristic	8.34E+10	-	-	-				Heuristic	1.95E+07	-	-	-
			CPLEX	**8.33E+10**	**1**	**≤0.10**	**124**				CPLEX	-	-	-	-
Who	440	8	LD	8.33E+10	1	11.10	37	GT	36733	11	LD	1.95E+07	1	1.43	4h
			BB+Basic	8.33E+10	5.0E+6	2.42	11943				BB+Basic	1.95E+07	4.3E+4	298.13	4h
			BB+LD	8.33E+10	193	≤0.10	117				**BB+LD**	**1.95E+07**	**101**	**≤0.10**	**1936**
			Heuristic	2.75E+06	-	-	-				Heuristic	486.75	-	-	-
			CPLEX	**2.75E+06**	**1**	**≤0.10**	**121**				CPLEX	-	-	-	-
HCV	572	12	LD	2.75E+06	1	0.34	536	rds	50000	3	LD	486.75	1	4.34	4h
			BB+Basic	2.75E+06	1.9E+6	≤0.10	5587				BB+Basic	476.79	2.0E+5	35.72	4h
			BB+LD	2.75E+06	4818	≤0.10	215				**BB+LD**	**476.79**	**23**	**≤0.10**	**811**
			Heuristic	2.62E+06	-	-	-				Heuristic	4.94E+08	-	-	-
			CPLEX	2.62E+06	23	0.23	285				CPLEX	-	-	-	-
Abs	740	21	LD	2.62E+06	1	0.86	726	KEGG	53413	23	LD	4.94E+08	1	151.02	4h
			BB+Basic	2.62E+06	2.4E+6	57.45	4h				BB+Basic	4.94E+08	1.8E+4	11.72	4h
			BB+LD	**2.62E+06**	**125**	**≤0.10**	**119**				**BB+LD**	**4.94E+08**	**177**	**≤0.10**	**3901**
			Heuristic	1136.93	-	-	-				Heuristic	1.26E+05	-	-	-
			CPLEX	1134.45	3	≤0.10	855				CPLEX	-	-	-	-
TR	980	10	LD	1136.93	1	0.80	130	urban GB_10	100000	2	LD	1.26E+05	1	26.5704	4h
			BB+Basic	1136.93	2.5E+6	209.23	4h				BB+Basic	1.15E+05	7.6E+4	17.86802	4h
			BB+LD	**1134.45**	**191**	**≤0.10**	**126**				**BB+LD**	**1.15E+05**	**49**	**≤0.10**	**6834**
			Heuristic	1.28E+09	-	-	-								
			CPLEX	1.28E+09	1	≤0.10	347								
SGC	1000	21	LD	1.28E+09	1	21.85	1248								
			BB+Basic	1.28E+09	3.8E+5	≤0.10	2739								
			BB+LD	**1.28E+09**	**189**	**≤0.10**	**140**								

* Out-of-memory in the root node with no solution.

6.1 Serial Results

Table 1 shows the serial numerical results for the datasets with less than 100,000 samples. In terms of best found upper bound (UB), the deterministic global optimization methods (CPLEX, BB+Basic, and BB+LD) can obtain better UB than the heuristic method in several datasets (e.g., IRIS, BM, RDS). While we try to obtain the best heuristic results by multiple initializations, there is still no guarantee for the heuristic method to reach the global optimum. Moreover, the heuristic method can not provide an optimal gap to estimate the solution quality. Notably, our method and CPLEX provide the same upper bounds and gaps for the datasets smaller than PR2392 (2392 samples).

As for the gaps and running times, CPLEX and LD can obtain a small gap ($\leq$0.1%) with less time when datasets are smaller than HCV (572 samples). However, when datasets are larger than HCV, our method, BB+LD, can reach the small gap with much less time than CPLEX and LD. Remarkably, our method can reach the small gap in all the datasets up to 100,000 samples within 4 hours. CPLEX fails to give a solution because of the out-of-memory error in the root node when datasets are larger than PR2392 (2392 samples). LD and BB+Basic methods may not reach the small gap within 4 hours or maximum iteration limits when datasets are larger than BM (249 samples). Recall that LD has no guarantee of reaching the global optimal solution because of the duality gap. All these results show that our method is extremely scalable for K-Medoids problems even under serial mode.

Notably, we followed the tuning guide in the CPLEX documentation to improve its performance. For example, we initiated the root node with the Heuristic solution and specified the node file and memory parameters. Moreover, we tried different MIP strategies, including start, pre-solve, and heuristic strategies. In Table 1, we present the best results obtained by CPLEX.

6.2 Parallel Results

For BB+LD, we executed parallel experiments utilizing MPI interfaces in the large and huge scale datasets with $K = 3$. The results of large-scale datasets ranging from 10,000 to 100,000 samples are presented in Appendix B. Table 2 shows the parallel numerical results for huge datasets with 100,000 to 2,458,285 samples. From this table, we can see that even for the dataset with 1,046,910 samples, BB+LD can still converge to the global optimal solution ($\leq$0.10 gaps) within one hour, which is **400 times larger** than the size of problem that the state-of-art global solver CPLEX can address. For USC1990 with 2,458,285 samples and 68 features, BB+LD can obtain a 6.33% gap within 4 hours. According to Theorem 1, we can obtain a global ϵ-optimal solution for USC1990 given more calculation time.

Table 2: Parallel results of huge scale datasets (BB+LD, $K = 3$)

DATASET	SAMPLE	DIMENSION	CORES	UB	NODES	GAP(%)	TIME(S)
RNG_AGR	199,843	7	1600	8.23E+14	99	$\leq$0.10	341.0
URBANGB	360,177	2	1600	4.14E+05	57	$\leq$0.10	327.0
SPNET3D	434,874	3	1600	2.28E+07	115	$\leq$0.10	865.0
RETAIL	541,909	2	1600	6.80E+09	1	$\leq$0.10	80.0
SYNTHETIC	1,000,000	2	6000	9.44E+06	3	$\leq$0.10	171.0
RETAIL-II	1,046,910	2	6000	2.31E+10	214	$\leq$0.10	2515.0
USC1990*	2,458,285	68	3000	6.91E+08	25	6.33	4H

* $d_{s,j}$ WAS COMPUTED ON-THE-FLY, NOT PRECOMPUTED AND STORED.

7 Complexity Analysis

7.1 Time Complexity

Denoting the number of samples as S, attributes as A, and clusters as K, our algorithm branches on the medoid regions to guarantee convergence, which contains $A \times K$ variables and is independent of the number of samples S. However, the number of BB nodes to converge is hard to predict since the efficiency of bounding methods depends on datasets. Also, the number of Lagrangian Dual (LD) iterations in each BB node is different because we set dynamic stopping criteria in case there is no update of LB in several continuous LD iterations. The time complexity of one LD iteration is $O(S^2)$ when $d_{s,j}$ are precomputed. If $d_{s,j}$ are not precomputed, the time complexity is $O(AS^2)$.

Table 3 compares computing time and nodes on synthetic datasets with various dimensions and samples. In this table, the average run-time of one LD iteration remains almost the same as the

dimension changes and increases almost quadratically as the number of samples increases. As for the BB nodes, although the number of branching variables is $A \times K$, the actual number of explored BB nodes does not strictly follow the growth of dimensions. These results are consistent with the complexity analysis mentioned above.

Table 3: Comparison of time and nodes on synthetic datasets with varies dimensions and samples

SAMPLES	DIMENSION	2	5	10	20	30	40	50	60	70	80	90	100
1,000	NODES*	10	3	3	31	19	3	3	39	7	3	69	3
	TIME (MS)†	4.28	4.28	4.06	3.57	3.62	4.02	4.11	3.78	4.32	3.8	3.21	3.79
10,000	NODES*	3	17	7	17	17	15	41	13	7	3	201	17
	TIME (MS)†	423	363	387	357	390	361	373	366	369	363	211	358
100,000	NODES*	4	7	27	31	3	15	27	13	25	7	25	21
	TIME (S)†	40.6	39.2	34.3	33.4	35.2	28.7	35.6	37.1	34.6	36.2	35.2	34.3

* THE NUMBER OF SOLVED NODES IN THE BB PROCEDURE UNTIL CONVERGENCE.
† TIME CONSUMED IN EACH LAGRANGIAN DUAL ITERATION.

7.2 Space Complexity of Distance

When distances $d_{s,j}$ are precomputed and stored, the space complexity of $d_{s,j}$ matrix is $O(S^2)$. In our implementation, the datasets with no more than 100,000 samples were computed on one compute node with 40 cores and 202GB RAM. For more than 100,000 samples, we executed the experiments on multiple compute nodes. Each core precomputes and stores part of the $d_{s,j}$ matrix, as described in Section 3. For example, we executed the one-million dataset RETAIL-II on 150 nodes with 6,000 cores. Each core only needs to store 1.25GB of $d_{s,j}$ matrix. For more than 1,000,000 samples, we calculated $d_{s,j}$ on the fly, without precomputing and storing. In this case, the time complexity of one LD iteration increases from $O(S^2)$ to $O(AS^2)$. Hence, we expect an acceptable slowdown when the dimension of datasets is small. For example, in Table 2, the dataset USC1990 with two-million samples and 68 features was executed without precomputing $d_{s,j}$, and the result is acceptable. Table 4 compares the computing time of precomputed distance and on-the-fly computing. We can conclude that although calculating $d_{s,j}$ on the fly is slower than precomputing $d_{s,j}$, the slowdown is acceptable.

Table 4: Comparison of precomputed distance and on the fly computing

DATASET	DIMENSION	TOTAL RUN TIME (S)		TIME PER LD ITERATION (MS)	
		ON-THE-FLY	PRECOMPUTED	ON-THE-FLY	PRECOMPUTED
ABS	21	34	12	1.42	0.70
HEMI	7	17	13	9.91	2.02
RDS_CNT	4	47	30	175	37.80
TR	10	58	15	2.54	0.73
TRR	24	365	74	110	10.80

8 Conclusion

We presented a scalable global optimization algorithm of the K-Medoids problem by applying a tailored reduced-space spatial branch and bound scheme. This algorithm includes a Lagrangian based lower bounding method and a basic lower bounding method. Bound tightening techniques are also proposed to accelerate the solution process. We demonstrate our algorithm's scalability by extensive numerical experiments and prove the convergence by theoretical analysis. Besides, one interesting finding is that our algorithm can reduce the objective values for some datasets while remaining the same as the heuristic method for most datasets. We also compared other clustering evaluation metrics in Appendix D, such as Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI). A similar conclusion with objective values holds for these metrics. However, it should be noted that only with the global optimal solution obtained in our work can we give a fair comparison of the heuristic methods and, in turn, contribute to developing a more efficient heuristic method.

Acknowledgement

The authors gratefully acknowledge the computing resources provided by SciNet (www.scinethpc.ca) and Digital Research Alliance of Canada (www.alliancecan.ca). Jiayang Ren acknowledges the financial support from the China Scholarship Council (Grant No. 202106320032).

References

[1] Adil Fahad, Najlaa Alshatri, Zahir Tari, Abdullah Alamri, Ibrahim Khalil, Albert Y. Zomaya, Sebti Foufou, and Abdelaziz Bouras. A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis. *IEEE Transactions on Emerging Topics in Computing*, 2(3):267–279, 2014.

[2] Leonard Kaufman and Peter J. Rousseeuw. Partitioning Around Medoids (Program PAM). In *Finding Groups in Data*, pages 68–125. John Wiley & Sons, Ltd, 1990.

[3] Leonard Kaufman and Peter J. Rousseeuw. Clustering Large Applications (Program CLARA). In *Finding Groups in Data*, pages 126–163. John Wiley & Sons, Ltd, 1990.

[4] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for K-medoids clustering. *Expert Systems with Applications*, 36(2):3336–3341, 2009.

[5] Erich Schubert and Peter J. Rousseeuw. Fast and eager k-medoids clustering: O(k) runtime improvement of the PAM, CLARA, and CLARANS algorithms. *Information Systems*, 101:101804, 2021.

[6] Gerard Cornuejols, Marshall L. Fisher, and George L. Nemhauser. Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms. *Management Science*, 23(8):789–810, 1977.

[7] C. Beltran, C. Tadonki, and J. Ph. Vial. Solving the p-Median Problem with a Semi-Lagrangian Relaxation. *Computational Optimization and Applications*, 35(2):239–260, 2006.

[8] Andréa Cynthia Santos. Solving large p-median problems using a Lagrangean heuristic. Technical report, Université Blaise Pascal (Clermont Ferrand 2), 2009.

[9] Sergio García, Martine Labbé, and Alfredo Marín. Solving Large p-Median Problems with a Radius Formulation. *INFORMS Journal on Computing*, 23(4):546–556, November 2011.

[10] Anton V. Ushakov and Igor Vasilyev. Near-optimal large-scale k-medoids clustering. *Information Sciences*, 545:344–362, 2021.

[11] N. Christofides and J.E. Beasley. A tree search algorithm for the p-median problem. *European Journal of Operational Research*, 10(2):196–204, 1982.

[12] Alberto Ceselli and Giovanni Righini. A branch-and-price algorithm for the capacitated p-median problem. *Networks*, 45(3):125–142, 2005.

[13] Sourour Elloumi. A tighter formulation of the p-median problem. *Journal of Combinatorial Optimization*, 19(1):69–83, 2010.

[14] IBM ILOG Cplex. V20.1.0: User's Manual for CPLEX. *International Business Machines Corporation*, 2020.

[15] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022.

[16] Claus C Carøe and Rüdiger Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1-2):37–45, 1999.

[17] Ramkumar Karuppiah and Ignacio 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.

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

[19] Kaixun Hua, Mingfei Shi, and Yankai Cao. A Scalable Deterministic Global Optimization Algorithm for Clustering Problems. In *International Conference on Machine Learning*, pages 4391–4401. PMLR, 2021.

[20] Hrishikesh D Vinod. Integer programming and the theory of grouping. *Journal of the American Statistical association*, 64(326):506–519, 1969.

[21] Laureano F. Escudero, M. Araceli Garín, Gloria Pérez, and Aitziber Unzueta. Scenario Cluster Decomposition of the Lagrangian dual in two-stage stochastic mixed 0–1 optimization. *Computers & Operations Research*, 40(1):362–377, 2013. 00049.

[22] Kibaek Kim and Victor M. Zavala. Algorithmic innovations and software for the dual decomposition method applied to stochastic mixed-integer programs. *Mathematical Programming Computation*, 10(2):225–266, 2018.

[23] Marshall L. Fisher. The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science*, 50:1861–1871, 2004.

[24] Jesús-Adolfo Mejía-de Dios and Efrén Mezura-Montes. A New Evolutionary Optimization Method Based on Center of Mass. In Kusum Deep, Madhu Jain, and Said Salhi, editors, *Decision Science in Action: Theory and Applications of Modern Decision Analytic Optimisation*, Asset Analytics, pages 65–74. Springer, Singapore, 2019.

[25] M. W. P. Savelsbergh. Preprocessing and Probing Techniques for Mixed Integer Programming Problems. *ORSA Journal on Computing*, (4):445–454, November 1994.

[26] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

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

[28] Edward Wang, Genpei Cai, Riley Ballachay, Yankai Cao, and Heather L. Trajano. Predicting Xylose Yield in Prehydrolysis of Hardwoods: A Machine Learning Approach. *Frontiers in Chemical Engineering*, page 84, 2022.

[29] Reiner Horst and Hoang Tuy. *Global optimization: Deterministic approaches*. Springer Science & Business Media, 2013.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section **??**
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] Our work is about a global solver of the K-Medoids problem. The problem itself may have societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 6 and the supplemental material
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] Our method is a deterministic method.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 6
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We compare our method with some existing methods and we cite the creators in the main body.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include our project codes in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We use the datasets from UCI Machine Learning Repository and three open-source datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] The meaning of datasets are described in UCI Machine Learning Repository.
5. If you used crowdsourcing or conducted research with human subjects... [No] Our work does not use crowdsourcing or conducted research with human subjects
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Convergence Analysis

As stated in Algorithm 1, the branch and bound scheme is a rooted tree, where the search space of the level 0 root node is $\overset{\circ}{M}$. We denote M_{l_q} as a sub-node at iteration l_q and level q. Its child node is denoted as $M_{l_{q+1}}$ satisfying $M_{l_{q+1}} \subset M_{l_q}$. A decreasing sequence from the root node $\overset{\circ}{M}$ to the node M_{l_q} is denoted as $\{M_{l_q}\}$. It is obvious that the sequences $\{\alpha_l\}$ and $\{\beta_l\}$ are monotonically non-increasing and non-decreasing correspondingly. In the following convergence analysis, we adapt the fundamental results from [29] to our work. Different from the BB procedure in [19], which is infinite, our algorithm has a finite BB procedure. Hence, we prove the convergence in a finite manner.

Definition 3. (Definition IV.3 [29]) A bounding operation is called **finitely consistent** if, at every step, any unfathomed partition element can be further refined, and if any decreasing sequence $\{M_{l_q}\}$ successively refined partition elements is finite.

Lemma 4. *The bounding operation in Algorithm 1 is finitely consistent.*

Proof. We first prove any unfathomed partition element M_{l_q} can be further refined. Any unfathomed M_{l_q} satisfies $\exists |M_{l_q} \cap X| > 1, k \in \mathcal{K}$ and $\alpha_l > \beta(M_{l_q}) + \epsilon, \epsilon > 0$. It is obvious that there exists at least one partition to be further refined.

We then prove any decreasing sequence $\{M_{l_q}\}$ successively refined partition elements is finite. Assuming by contradiction that there exists a sequence $\{M_{l_q}\}$ that is infinite. In our algorithm, since we branch on the first-stage variable μ^k corresponding to the diameter of M, this subdivision is exhaustive, we have $\lim_{q\to\infty} \delta(M_{l_q}) = 0$ and $\{M_{l_q}\}$ converge to one point $\bar{\mu}$. If $\bar{\mu} \in X$, there is a ball around $\bar{\mu}$, denoted as $B_r(\bar{\mu}) = \{\mu \mid ||\mu - \bar{\mu}|| \leq r\}$, satisfying $|B_r(\bar{\mu}) \cap X| = 1$. There exists a q_0 that $M_{l_q} \subset B_r(\bar{\mu}), \forall q \leq q_0$. At this q_0 iteration, $M_{l_{q_0}}$ will not be branched anymore. Because X is finite, we have the sequence $\{M_{l_q}\}$ is finite in this case. If $\bar{\mu} \not\subset X$, there is a ball around $\bar{\mu}$, denoted as $B_r(\bar{\mu}) = \{\mu \mid ||\mu - \bar{\mu}|| \leq r\}$, satisfying $|B_r(\bar{\mu}) \cap X| = 0$. There exists a q_0 that $M_{l_q} \subset B_r(\bar{\mu}), \forall q \leq q_0$. At this q_0 iteration, $M_{l_{q_0}}$ will be deleted. Consequently, in this case, the sequence $\{M_{l_q}\}$ is finite. Hence, it is impossible to exists a sequence $\{M_{l_q}\}$ that is infinite. □

Theorem 5. *(Theorem IV.1 [29]) In a BB procedure, suppose that the bounding operation is finitely consistent. Then the procedure terminates after finitely many steps.*

Lemma 6. *Algorithm 1 terminates after finitely many steps*

Proof. From Lemma 4, we have the bounding operation in Algorithm 1 is finitely consistent. From Theorem 5, Algorithm 1 terminates after finitely many steps □

Finally, we prove that the BB scheme is convergent, as shown in Theorem 1:

Theorem 1. *Algorithm 1 is convergent to the global optimal solution after a finite step L, with $\beta_L = z = \alpha_L$, by only branching on the space of μ.*

Proof. From Lemma 6, we have Algorithm 1 terminates after finite steps. The algorithm terminates with two situations. The first situations is $|\beta_l - \alpha_l| \leq \epsilon$. When $\epsilon = 0$, we have $\beta_l = z = \alpha_l$.

The second situations is $\mathbb{M} = \emptyset$. A node M is deleted from $\mathbb{M}$ and not further partitioned either because $\beta(M) > \alpha_l$ or $|M^k \cap X| = 1, \forall k \in \mathcal{K}$. The first case obviously does not contain the global optimal solution μ^*. Therefore, the node M' containing μ^*, is not further partitioned because $|M'^k \cap X| = 1, \forall k \in \mathcal{K}$. After bound tightening according to the "medoids on samples" constraint, the tightened node $M' = \{\mu^*\}$. Obviously for this node, we have $\beta_l = \beta(M') = z = \alpha(M') = \alpha_l$. Consequently, we have proved Theorem 1.

□

B Parallel results of large scale datasets

Table 5 shows the parallel numerical results for large scale datasets with 10,000 to 2,458,285 samples. For the large-scale datasets ranging from 10,000 to 100,000 samples, we obtain an average of 13.80x speedup in time with 40 cores. Here, speedup ratio is defined as $\frac{\bar{T}_1}{\bar{T}_c}$, in which $\bar{T}_c$ is the overall calculation time with c cores. The average efficiency is low since the time ratio of the parallel part is relatively low in the small datasets. Consequently, we can expect a higher speedup in the big datasets. For example, we obtain a 25.89x $Speedup$ for URBANGB_10 with 100,000 samples compared to 7.57x for HTRU2 with 17,898 samples.

Table 5: Parallel results of large scale datasets (BB+LD, $k = 3$)

Dataset	Sample	Dimension	Cores	UB	Nodes	Gap(%)	Time(s)
HTRU2	17,898	8	40	8.21E+07	465	≤0.10	205
GT	36,733	11	40	1.95E+07	101	≤0.10	161
rds	50,000	3	40	476.792	23	≤0.10	71
KEGG	53,413	23	40	4.94E+08	177	≤0.10	322
urbanGB_10	100,000	2	40	1.15E+05	49	≤0.10	264

C Additional results of K-Medoids problems with $K = 5$ and $K = 10$

In this section, we perform several additional experiments with more clusters on the datasets ranging from 100 to 100,000 samples. All these experiments use the same setup in Section 6. Table 6 and 7 are the results of the BB+LD method with cluster number $K = 5$ and $K = 10$ respectively. When the cluster number is bigger, our BB+LD are more likely to obtain an upper bound smaller than the Heuristic UB. However, the difference in UB between BB+LD and Heuristic becomes smaller. Moreover, the search space of medoids increases as the cluster number, making it harder to obtain the global optimum. Nevertheless, our BB+LD method can still obtain a relative gap smaller than 0.1% within 4 hours for most datasets when $K = 5$ and a reasonable relative gap within 4 hours when $K = 10$.

Table 6: Additional results of K-Medoids problems with BB+LD and $K = 5$

Dataset	Sam-ple	Dim-ension	Heuristic UB	Serial Results (core=1)				Parallel Results (core=40)			
				UB	Nodes	Gap (%)	Time (s)	UB	Nodes	Gap (%)	Time (s)
iris	150	4	5.1000E+01	5.0920E+01	88	≤0.10	355	-	-	-	-
seeds	210	7	4.0372E+02	4.0121E+02	43	≤0.10	376	-	-	-	-
glass	214	9	4.3789E+02	4.3773E+02	6,846	≤0.10	592	-	-	-	-
BM	249	6	6.0249E+05	6.0249E+05	281	≤0.10	389	-	-	-	-
UK	258	5	4.0166E+01	4.0166E+01	1,869	≤0.10	457	-	-	-	-
HF	299	12	3.0998E+11	3.0998E+11	43,827	≤0.10	1723	-	-	-	-
Who	440	8	5.5914E+10	5.5914E+10	32,832	≤0.10	1840	-	-	-	-
HCV	572	12	1.9716E+06	1.9716E+06	1,011,564	≤0.10	12768	-	-	-	-
Abs	740	21	1.7476E+06	1.7472E+06	329	≤0.10	410	-	-	-	-
TR	980	10	9.6555E+02	9.5339E+02	3,975	≤0.10	824	-	-	-	-
SGC	1,000	21	4.6969E+08	4.6919E+08	23,827	≤0.10	3290	4.6919E+08	23,827	≤0.10	880
hemi	1,955	7	5.3864E+06	5.3811E+06	1,782	≤0.10	930	5.3811E+06	1,782	≤0.10	66
pr2392	2,392	2	1.1620E+10	1.1619E+10	405	≤0.10	625	1.1619E+10	405	≤0.10	41
TRR	5,456	24	1.6991E+05	1.6870E+05	3,029	≤0.10	3094	1.6870E+05	3,029	≤0.10	499
AC	7,195	22	1.6377E+03	1.6361E+03	1,473	≤0.10	3552	1.6361E+03	1,473	≤0.10	356
rds_cnt	10,000	4	5.3725E+06	5.3725E+06	4,051	≤0.10	7171	5.3725E+06	4,051	≤0.10	450
HTRU2	17,898	8	4.2154E+07	4.2154E+07	3,453	10.85	4h	4.2154E+07	33,682	2.34	4h
GT	36,733	11	1.3358E+07	1.3351E+07	669	0.97	4h	1.3351E+07	4,095	≤0.10	4776
rds	50,000	3	2.8452E+02	2.8265E+02	499	0.94	4h	2.8265E+02	893	≤0.10	1338
KEGG	53,413	23	1.9201E+08	1.9201E+08	503	24.54	4h	1.9200E+08	8,667	1.75	4h
urbanGB_10	100,000	2	5.6232E+04	5.6232E+04	104	3.63	4h	5.6232E+04	543	≤0.10	2266

Table 7: Additional results of K-Medoids problems with BB+LD and $K = 10$

Dataset	Sample	Dimension	Heuristic UB	Serial Results (core=1)				Parallel Results (core=40)			
				UB	Nodes	Gap (%)	Time (s)	UB	Nodes	Gap (%)	Time (s)
iris	150	4	3.0380E+01	2.9790E+01	6,219	≤0.10	735	-	-	-	-
seeds	210	7	2.1849E+02	2.1452E+02	919	≤0.10	448	-	-	-	-
glass	214	9	2.5325E+02	2.5186E+02	31,983	≤0.10	2566	-	-	-	-
BM	249	6	3.8181E+05	3.7597E+05	10,965	≤0.10	1204	-	-	-	-
UK	258	5	2.9785E+01	2.9280E+01	176,103	1.70	4h	-	-	-	-
HF	299	12	6.9604E+10	6.9604E+10	132,742	21.96	4h	-	-	-	-
Who	440	8	3.4614E+10	3.4020E+10	107,430	11.13	4h	-	-	-	-
HCV	572	12	1.1592E+06	1.1315E+06	77,009	7.20	4h	-	-	-	-
Abs	740	21	1.1083E+06	1.0786E+06	56,090	0.19	4h	-	-	-	-
TR	980	10	7.7370E+02	7.7247E+02	41,752	3.17	4h	-	-	-	-
SGC	1,000	21	1.1742E+08	1.1742E+08	33,388	20.70	4h	1.1742E+08	254,583	16.33	4h
hemi	1,955	7	2.7421E+06	2.7421E+06	16,711	9.69	4h	2.7068E+06	351,441	0.15	4h
pr2392	2,392	2	5.3578E+09	5.3578E+09	8,706	4.97	4h	5.3578E+09	195,660	0.52	4h
TRR	5,456	24	1.3933E+05	1.3796E+05	2,487	0.18	4h	1.3796E+05	22,527	≤0.10	4h
AC	7,195	22	1.1817E+03	1.1817E+03	1,341	3.16	4h	1.1637E+03	32,530	0.63	4h
rds_cnt	10,000	4	1.6119E+06	1.6119E+06	1,020	26.13	4h	1.6119E+06	33,615	13.15	4h
HTRU2	17,898	8	1.8273E+07	1.8273E+07	350	24.39	4h	1.8273E+07	16,083	16.86	4h
GT	36,733	11	8.9909E+06	8.9909E+06	65	4.89	4h	8.9909E+06	5,380	1.13	4h
rds	50,000	3	1.3273E+02	1.3273E+02	40	7.89	4h	1.3273E+02	4,756	3.79	4h
KEGG	53,413	23	6.1564E+07	6.1564E+07	51	67.53	4h	6.1564E+07	919	30.11	4h
urbanGB_10	100,000	2	2.5123E+04	2.5123E+04	14	23.54	4h	2.5123E+04	1,427	9.04	4h

D Comparison of heuristic methods for K-Medoids problems

To illustrate the effectiveness of our upper bound method (Heuristic UB), we compares its performance with several popular heuristic methods in the literature, including Kmeans, Kmeans++, and PAM. Here, we run all the methods several times with random seeds and select the best result. The centers of K-Means and K-Means++ are projected to the nearest samples to fulfill the "Medoids on Samples" constraint in the KMedoids problem. This table shows that the Heuristic UB can always obtain the same or better objective value than Kmeans, Kmeans++, and PAM.

Table 8: Additional results of heuristic methods for K-Medoids problems

Dataset	Kmeans	Kmeans++	PAM	Heuristic	BB+LD
iris	84.63	84.63	90.99	84.63	**83.91**
seeds	598.29	598.29	608.72	598.29	598.29
glass	629.02	629.02	652.15	629.02	629.02
BM	8.65E+05	8.65E+05	9.17E+05	8.65E+05	**8.63E+05**
UK	50.77	51.19	51.06	50.77	50.77
HF	7.83E+11	7.83E+11	7.83E+11	7.83E+11	7.83E+11
Who	8.34E+10	8.34E+10	8.44E+10	8.34E+10	**8.33E+10**
HCV	2.85E+06	2.82E+06	2.76E+06	2.75E+06	2.75E+06
Abs	2.62E+06	2.62E+06	2.66E+06	2.62E+06	2.62E+06
TR	1.14E+03	1.14E+03	1.16E+03	1.14E+03	**1.13E+03**
SGC	1.28E+09	1.28E+09	1.49E+09	1.28E+09	1.28E+09
hemi	9.92E+06	9.91E+06	1.18E+07	9.91E+06	9.91E+06
pr2392	2.13E+10	2.13E+10	2.53E+10	2.13E+10	2.13E+10
TRR	1.97E+05	1.96E+05	1.97E+05	1.96E+05	1.96E+05
AC	2.21E+03	2.21E+03	2.34E+03	2.21E+03	**2.20E+03**
rds_cnt	1.49E+07	1.49E+07	1.50E+07	1.49E+07	1.49E+07
HTRU2	8.21E+07	8.21E+07	8.61E+07	8.21E+07	8.21E+07
GT	1.95E+07	1.95E+07	1.96E+07	1.95E+07	1.95E+07
rds	476.88	476.88	486.75	476.88	**476.79**
KEGG	4.94E+08	4.94E+08	4.95E+08	4.94E+08	4.94E+08
urbanGB_10	1.15E+05	1.15E+05	1.26E+05	1.15E+05	1.15E+05
rng_agr	8.23E+14	8.23E+14	-*	8.23E+14	8.23E+14
urbanGB	4.14E+05	4.14E+05	-*	4.14E+05	4.14E+05
spnet3D	2.28E+07	2.28E+07	-*	2.28E+07	2.28E+07
retail	6.80E+09	6.80E+09	-*	6.80E+09	6.80E+09
synthetic	9.44E+06	9.44E+06	-*	9.44E+06	9.44E+06
retail-II	2.90E+10	2.31E+10	-*	2.31E+10	2.31E+10
USC1990	6.91E+08	6.91E+08	-*	6.91E+08	6.91E+08

* Out-of-memory on one computer node

We also compared other clustering evaluation metrics, such as Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI). Since these clustering evaluation metrics need to compare the predicted cluster with ground truth, we can only compare the datasets with ground truth, such as IRIS, HCV, HF, HRTU2 and UK. Comparing Table 8 and Table 9, one interesting finding is that our algorithm can reduce the objective values for some datasets while remaining the same as the heuristic method for most datasets. A similar conclusion with objective values also holds for ARI and NMI. However, it should be noted that only by comparing the heuristic solution with the global optimal solution as we did in the table, we can confidently claim that the heuristic method can do a fairly nice job in finding near-optimal solutions.

Table 9: ARI and NMI of heuristic methods for K-Medoids problems

Dataset	Sample	Dimension	Cluster Number (Ground Truth)	Method	UB (Object Value)	ARI	NMI
iris	150	4	3	Kmeans	8.4680E+01	0.7302	0.7582
				Kmeans++	8.4680E+01	0.7302	0.7582
				PAM	9.1040E+01	0.7060	0.7561
				Heuristic	8.4680E+01	0.7302	0.7582
				BB+LD	**8.3960E+01**	**0.7455**	**0.7980**
HCV	572	12	4	Kmeans	2.2873E+06	0.4417	0.2454
				Kmeans++	2.2873E+06	0.4417	0.2454
				PAM	2.3168E+06	**0.5941**	**0.3363**
				Heuristic	2.2873E+06	0.4417	0.2454
				BB+LD	2.2873E+06	0.4417	0.2454
HF	299	12	2	Kmeans	1.3512E+12	0.0175	0.0025
				Kmeans++	1.3512E+12	0.0175	0.0025
				PAM	1.3917E+12	0.0122	0.0013
				Heuristic	1.3512E+12	0.0175	0.0025
				BB+LD	1.3512E+12	0.0175	0.0025
HRTU2	17898	8	2	Kmeans	1.2536E+08	-0.0780	0.0267
				Kmeans++	1.2536E+08	-0.0780	0.0267
				PAM	1.2800E+08	-0.0738	0.0221
				Heuristic	1.2536E+08	-0.0780	0.0267
				BB+LD	**1.2535E+08**	**-0.0779**	**0.0271**
UK	258	5	4	Kmeans	4.5044E+01	0.2378	0.3253
				Kmeans++	4.5044E+01	0.2378	0.3253
				PAM	4.5420E+01	0.1539	0.2210
				Heuristic	4.5040E+01	0.2378	0.3253
				BB+LD	4.5040E+01	0.2378	0.3253