Constrained Clustering Problem: Heuristic and Exact Approaches for Optimizing Centroid

Project Report

Università Di PISA

Department of Computer Science

Master's in data science & business informatics

(P) is the Constrained Clustering Problem in the L1 norm: find k centroids for the clusters and assign each of the n inputs to exactly one centroid so as to minimise the sum of the L1 distances between each point and the assigned centroid.

(A1) is a heuristic approach obtained by suitably modifying the k-medians one.

(A2) Is the exact approach of applying a general-purpose solve to an appropriate formulation of the problem.

Submitted To:

Prof: Antonio Frangioni

Submitted By:

Amir Hassan (683185)

Attia Maqbool (682009)

Group 64

Academic Year 2023/24

Abstract:

In this report we addressed the constrained Clustering problem using L1 norm, also known as Manhattan distance. The objective is to find K-centroids and assign n data points to these centroidsin such a way that the sum of the L1 distance between each point and its assigned centroids is minimized. We explore two primaries approached in our project; A heuristic method based on a modified K-median algorithm, that offer practical and efficient solution and second is exact methodformulated as an Integer Linear Program (ILP), which make sure to find optimal clustering but can be computationally intensive.

The heuristic approach involves initializing random cluster centers, iteratively assigning data points to the nearest centers and updating the center of the median of assigned points. This method is fast and suitable for large datasets but may not always find the optimal solution. The exact approach, formulated as an MIP, precisely minimizes the total L1 distance by defining binary assignment variable and solving for optimal cluster centers. While computationally demanding, this approach ensures the best possible clustering results. By understanding and comparing these methods, we can make informed decisions on their application to various clustering tasks.

This report aims to provide a detailed overview of theoretical foundations of these methods, their applications, and the scenarios where each approach is most suitable.

Contents

Chapter: 1

Introduction

In the data analysis, clustering plays a vital role. It involves grouping a set of objects that have the same group (or cluster) and are similar to each other than to those in other groups. We focused on solving specific clustering problems using L1 norm, also called the Manhattan distance. We used this to measure the distance between two points in a grid-based path by summing the absolute differences of their coordinates. For example, L1 distance between two points $(x1, y1)$ and $(x2, y2)$ in a two-dimensional space calculated as;

$$| x1 - x2| + |y1 – y2|$$

This contrasts with L2 norm known as Euclidean distance, which measure the distance between two straight lines points. The L1 norm preferred to use in high dimensional or space data scenarios because it is more robust to outliers.

Our goal is to determine the best way to cluster data points to minimize the total L1 distance from each point to its assigned cluster center. Mathematically we aim to minimize.

$\sum_{i=1}^{n} \min_{j=1}^{k} \|x_i - c_j\|_1$ where x_i are the data points and c_j are the cluster centers.

We will explore both a heuristic approach (a practical good enough method) and exact approach (a mathematically optimal method).

Regarding the second point we choose to investigate the usage of Mixed Integer Programming (MIP) and Integer Linear Programming (ILP).[1]

1.1. Integer Linear Programming (ILP)

Integer Linear Programming (ILP) is a mathematical optimization technique where all the decision variables are constrained to take to integer values. The objective function and constraints are linear, making ILP a powerful tool for specifically discrete optimization problems.

Characteristics:

Variables: All variables in ILP should be Integers Objective
Function: Linear, involving integer variables Constraints:
Linear constraints involving integer variable

Use cases:

ILP is typically used in scenarios where decision variables would be representing discrete items, such as scheduling, allocation and routing problems. For example, in a scheduling problem, variables might represent whether a task is assigned to a time slot, inquiring binary (0 or 1) values.

Pros and Cons

Pros:

1- Suitable for problems requiring the integer solutions
2- Can model complex combinatorial problems.
Cons:

1- Computationally expensive for large scale problems.
2- Limited flexibility due to only integer constraints.

1.2 Mixed Integer Programming (MIP)

Mixed Integer Programming (MIP) extends ILP by allowing both integer and continuous variables. This flexibility makes us confident to choose this and makes MIP a versatile tool forwider range of optimization problems.

Characteristics

Variables: MIP included both integer and continuous variables.

Objective Function: Linear, involving both integer and continuous variables.
Constraints: Linear constraints involving both types of variables.

Use cases

MIP is commonly used in resource allocation, production planning and network design, where some decisions are discrete like weather to open a facility and other are continuous like the amount of resources to allocate.

Pros and Cons

Pros:

1- Flexible, accommodating both integer and continuous variables.

2- Can model be used for wider range of real-world problems.

Cons:

1- Still computationally intensive, though often more manageable than ILP for certain problems.

2- Required sophisticated solvers for efficient solutions.

1.3 Heuristic Approach: Modified K-Medians Algorithm

A heuristic approach provides a quick, practical solution, even if it is not the perfect one. We canadapt the k-Medians algorithm for our L1 norm clustering problem.

Steps:

Step 1- Initialization: Select K points from the dataset randomly to serve as the initial cluster center.

Step 2- Assignment: For each data point, calculate the L1 distance to each cluster center and assign the point to the closet center.

Step 3- Update: After all the points are assigned to clusters, update each cluster center to be the median of the points in that cluster. the median minimizes the sum of the absolute deviations, whichaligns with the L1 norm.

This can be understood with the following explanation:

For a set of points {x1, x2,....., xn} in one dimension, the median m is defined as the value that minimizes the sum of absolute deviations.

$$m = \arg\min_y \sum_{i=1}^{n} |x_i - y|$$

This is a well-known result in statistics and can be found in many standard text on robust statistics or data analysis. The intuition behind this is that the median splits the data into two halves, such thatthe total distance from all points to the median is minimized compared to any other point.

In higher dimensions, the median is computed for each coordinate separately, making it robust and

well-aligned with L1 norm properties. For further details and proofs, could be found in [1,2,3].

Step 4- Iteration: Repeat steps 2 and 3 until the assignments no longer change significantly (convergence).

Why We Use This Approach?

Speed: This method is fast and can handle large dataset efficiently.

Good Solutions: While it may not always find the optimal solution, it usually finds a very good one.

Limitations:

Local Optimization: The algorithm might get stuck in a local optimum, meaning the solution couldbe good but not the best possible.

Sensitivity to Initial Centers: The initial choice of the centers can influence the final clusteringresult.

1.4 Exact Approach: L1 Clustering as a Mixed Integer Linear Program (MILP)

The L1 clustering problem involves partitioning a set of data points into clusters such that the sum of the L1 norms (Manhattan distances) of points to their assigned cluster centers is minimized. Below is the detailed formulation of the problem as an MILP.

Given: A set of n data points $x1,x2,\ldots,x_n$ in R^d and an integer k representing the number of clusters.

Objective: Partition the data points into k clusters and determine the cluster centers so that the total L1 distance between data points and their respective cluster centers is minimized.

Sets and Indices

• $i \in \{1, 2,\ldots,n\}$: Index for data points.

• $j \in \{1, 2,\ldots,k\}$: Index for clusters.

• $l \in \{1, 2,\ldots,d\}$: Index for dimensions.

Parameters

• $xi \in R^d$: Coordinates of data point i.

• n: Number of data points.

• k: Number of clusters.

• d: Dimensionality of data points.

• Ml: A large constant (Big-M value) for each dimension l, which is computed as:

$$M_l = \max(X[:, l]) - \min(X[:, l])$$

Decision Variables

• *yij* ∈ {0,1}: Binary variable indicating whether data point i is assigned to cluster j.

• *cj=(cjl)* ∈ R^d*:* Continuous variables representing the coordinates of the center of cluster j.

• zi ∈ R: Continuous variables representing the *L1* distance from data point *i* to its assigned cluster center.

• dijl ∈ R^d: Auxiliary variables to represent the absolute differences between coordinates of data points and cluster center.

Objective Function

The objective function minimizes the total L1 distance from each data point to its assigned cluster center. This can be formulated as:

$$\text{Minimize} \quad \sum_{i=1}^{n} z_i \tag{1}$$

where zi is the L1 distance from data point i to its assigned centroid.

Constraints

1. **Assignment Constraints**: Here each data point must be assigned to exactly one cluster. Thisensures that each point is only assigned to one centroid:

   $$\sum_{j=1}^{k} y_{ij} = 1 \quad \forall i \in \{1, 2, \ldots, n\} \tag{2}$$

   Meaning that for each data point i, the sum of binary variables yij across all clusters j must be equal to 1.

2. **Distance Calculation:**
   The constraint ensures that zi, the L1 distance for point i, is correctly calculated only for the assigned cluster:

   $$z_i \geq \sum_{l=1}^{d} d_{ijl} - M \cdot (1 - y_{ij}) \quad \forall i, j \tag{3}$$

Where $M = \sum_{l=1}^{d} M_l$, and M_l is the maximum distance in dimension l.

3. **Absolute Difference Calculation:**
   For each dimension l, calculate the L1 distance directly:

   Positive Direction:

$$d_{ijl} \geq x_{il} - c_{jl} \quad \forall l, i, j \tag{4}$$

   Negative Direction:

$$d_{ijl} \geq c_{jl} - x_{il} \quad \forall l, i, j \tag{5}$$

Where dijl represents the absolute difference in dimension l between point i and cluster center j.

4. **Bounding Box for Centroids:**

The coordinates of the centroids are constrained within a bounding box defined by the minimum and maximum coordinates of the data points:

$$\min_i x_{il} \leq c_{jl} \leq \max_i x_{il} \quad \forall j, l \tag{6}$$

How CBC (Coin-or Branch and Cut) Solves MILP [3]

- CBC solves MILP using a branch-and-bound method, supported by cutting planes to improve the search efficiency.
- It starts by solving the linear relaxation of the MILP, then branches on integer variables, and uses bounding and pruning to narrow the search.
- Cutting planes tighten the problem by eliminating fractional solutions without excluding feasible integer ones.
- Heuristics and preprocessing steps help to find feasible solutions early and reduce problem size, while CBC's modularity allows integration with other solvers for the linear components.

Why We Use this Approach?

Optimality: This method guarantees finding the best possible clustering solution by mathematically minimizing the total L1 distance.

Precision: Ensures that the solution is precise, making it suitable for applications where accuracy is critical.

Limitations:

Computationally Intensive: Solving an MILP can be very time consuming, especially for largedatasets.

Complexity: Requires sophisticated mathematical tools and solvers, making it more complex to implement compared to heuristic methods.

1.5 Conclusion

In the final discussion of chapter one, the constrained clustering algorithm L1 norm can be approached in two main ways: a heuristic method using a K-medians algorithm and exact method using mixed integer programming. The heuristic method, offers speed and practically, making it suitable for large datasets. The exact approach provides guaranteed optimal solutions, ideal for scenarios where precision is mainly focused. We will explore these approaches in the next chapter.

Chapter: 2

Optimizers

In this chapter, we will explore optimization algorithms tailored to solve the Constrained Clustering Problem with the L_1 norm. Our primary goal is to identify k centroids and assign each of the dataset's n inputs to exactly one centroid in a way that minimizes the sum of the L_1 distances between each point and its assigned centroid.

2.1 Introduction to Optimizers

In the realm of data science and machine learning, optimizers play a crucial role in refining models to perform more effectively on given datasets. In the specific context of constrained clustering, optimizers are instrumental in minimizing the sum of L_1 *norm* distances between data points and their designated centroids. This problem involves identifying a set of centroids such that the total distance, defined according to the L_1 *norm*, between each data point and its nearest centroid, is minimized while adhering to constraints like the maximum number of points per cluster or specific distribution requirements across clusters.

2.2 Heuristic Approach - Modified k-medians Method

The heuristic approach for the constrained clustering problem involves adapting the traditional k-medians method to the L_1 *norm*, which is more robust to outliers compared to the Euclidean norm (L2). The L_1 *norm* minimizes the median of the absolute deviations rather than the mean, making it suitable for distributions with high variability or skewed data.

2.2.1 Technical Description

Objective: Minimize the sum of L_1 *norm* distances between data points and their respective centroids, involving the absolute deviations from the median.

Algorithm 1: Basic Modified k-medians

Require:

- Number of centroids k
- Convergence threshold ϵ

Initialize:

- $C \leftarrow$ Randomly choose k centroids from the dataset.

- *labels* ← Initialize an array to store the cluster assignment for each point.

Repeat:

- C_{old} ← C (Store old centroids for convergence check)
- **For each** point x_i in the dataset **do:**
    - d_{min} ← ∞ (Initialize minimum distance to infinity)
    - **For each** centroid c_j in C **do:**
        - Compute L_1 distance $d=\sum|x_i - c_j|$
        - **If** $d < d_{min}$ **then**:
            - d_{min} ← d (Update the minimum distance)
            - $labels[_i]$ ← j (Assign point to the closest centroid)
- **For each** centroid c_j in C **do:**
    - Update c_j by calculating the median of all points x_i where $labels[i]= j$

Until $max_j \|c_j - c_{old,\ j}\| < \epsilon$ (Check convergence)

Output:

- Cluster centroids C
- Cluster assignments *labels*

Explanation of the Basic Modified *k-medians* Algorithm

The Basic Modified *k-medians* Algorithm is a clustering technique that partitions a dataset into k distinct clusters by minimizing the L_1 *norm* (also known as Manhattan distance) between data points and their assigned centroids. The L_1 *norm* is particularly useful in scenarios where outliers might heavily influence the results, as it is less sensitive to extreme values compared to the L_2 *norm* (Euclidean distance). The algorithm consists of several key steps:

Initialization

- **Centroid Selection**: The algorithm begins by initializing the cluster centroids. This is done by randomly selecting k data points from the dataset to serve as the initial centroids. This random selection serves as the starting point for the iterative process.

- **Label Array**: An array called *labels* is also initialized, which will store the cluster assignment for each data point in the dataset. Initially, this array can be empty or set with default values indicating no assignment.

Iterative Process

- **Storing Old Centroids**: At the beginning of each iteration, the current centroids are stored in an array called *C_{old}*. This is crucial for the convergence check at the end of each iteration.
- **Assignment**: Each data point in the dataset is then considered in turn. For each point x_i , the algorithm calculates the L_1 distance to each centroid c_j. The L_1 distance is computed as the sum of the absolute differences between the coordinates of the point and those of the centroid:

$$d = \sum | x_i - c_j |$$

  The centroid that has the smallest distance to the point is identified, and the point is assigned to this centroid's cluster. This assignment is recorded in the labels array.

- **Update**: After all points have been assigned to clusters, the next step is to update the centroids. For each cluster, the new centroid is calculated as the median of all points assigned to that cluster. The median is chosen because, in one dimension, it minimizes the sum of absolute deviations from the central value, making it an appropriate choice for the L_1 *norm*.

Convergence Check

- The iterative process continues until the change in centroids is less than a specified threshold ϵ. This is determined by comparing the stored centroids (C_{*old*}) with the updated centroids from the current iteration. The comparison involves calculating the maximum change in centroids across all clusters:

  $$\max_j \| c_j - c_{old,\,j} \| < \epsilon$$

  If this condition is met, the algorithm has converged, and the current centroids and cluster assignments are considered the final output.

Output

- **Cluster Centroids (*C*)**: The final positions of the cluster centroids after convergence.
- **Cluster Assignments (*labels*)**: The assignment of each data point in the dataset to a cluster.

This algorithm effectively groups the data points into clusters such that the total intra-cluster L_1 distance is minimized. The choice of L_1 *norm* and the median for updating centroids make this

algorithm particularly robust against outliers, providing reliable clustering even in the presence of noisy data.

Why Use This Approach?

Speed: This method is fast and can handle large datasets efficiently.

Good Solutions: While it may not always find the optimal solution, it usually finds a very good one.

Limitations:

Local Optimization: The algorithm might get stuck in a local optimum, meaning the solution could be good but not the best possible.

Sensitivity to Initial Centers: The initial choice of the centers can influence the final clustering result.

2.3 Exact Approach: Mixed Integer Programming (MIP)

Mixed Integer Programming (MIP) is crucial in our clustering project because it allows for precise modelling of both discrete and continuous variables in one framework. This capability is essential for accurately assigning data points to clusters (a discrete choice) while optimizing the positions of cluster centroids (a continuous variable). MIP's ability to integrate complex constraints ensures that our solution is not only optimal but also tailored to handle specific requirements of the L_1 *norm*-based clustering efficiently.

Algorithm 1: Branch and Bound Algorithm for MIP

Objective: Minimize the total L_1 distance across all clusters by solving the MIP problem efficiently using a systematic search approach.

Algorithm: Branch and Bound for MIP

Require:

- Set of data points $\{x_1, x_2, \ldots, x_n\}$
- Number of clusters k

Initialization:

1. **Initial Formulation**: Begin with the basic MIP model formulation:
    - **Objective Function**: Minimize the total L_1 distance between data points and centroids.

- **Decision Variables**:
  - $X_{ij} \in \{0,1\}$: Binary variable indicating if data point i is assigned to cluster j.
  - $C_j \in R^d$: Continuous variables representing the coordinates of the cluster centroids.
- **Constraints**:
  - Each data point must be assigned to exactly one cluster.
  - Other problem-specific constraints (e.g., capacity limits, balancing constraints).

2. **Node Initialization**:
   - Start with the root node representing the original MIP problem.
   - Calculate the linear relaxation of the MIP problem (i.e., solve the problem by ignoring the integer constraints on the variables).

Procedure:

3. **Branching**:
   - Identify a decision variable x_{ij} that is fractional in the current relaxed solution (i.e. x_{ij} is not an integer).
   - Create two new subproblems (branches) by imposing additional constraints on x_{ij}:
     - **Branch 1**: Add the constraint $x_{ij} = 0$.
     - **Branch 2**: Add the constraint $xij = 1$.
4. **Bounding**:
   - Solve the linear relaxation for each branch. This gives a lower bound on the objective function value for the subproblem.
   - If a branch yields an integer solution, this becomes a candidate for the optimal solution.
   - If the lower bound of a branch is worse than the current best integer solution (i.e., objective value), prune that branch (discard it).

5. **Pruning**:
    - **Node Pruning**: If the solution to a branch's linear relaxation is worse than the best-known integer solution or is infeasible, discard the branch.
    - **Optimality Check**: If a node results in an integer solution that is better than the current best solution, update the best solution.
6. **Selection**:
    - Continue selecting the most promising branch (often using a strategy like depth-first or best-first search) and repeat the branching, bounding, and pruning steps until all branches are either pruned or solved.
7. **Convergence**:
    - The algorithm terminates when all branches have been explored or pruned. The best integer solution found during the process is the optimal solution to the MIP.

Output:

- **Optimal Cluster Centroids C**: The coordinates of the centroids after solving the MIP.
- **Optimal Cluster Assignments labels**: The assignment of each data point to a cluster, as determined by the optimal MIP solution.

Why Use Branch and Bound?

- **Optimality**: This method guarantees finding the best possible clustering solution by mathematically minimizing the total L1 distance.
- **Efficiency**: By systematically pruning non-promising branches, the algorithm significantly reduces the number of potential solutions that need to be explored, making it more efficient than brute-force methods.

Limitations:

- **Computationally Intensive**: Despite its advantages, Branch and Bound can become computationally expensive for very large problems or highly complex MIPs. The performance heavily depends on the problem structure and the quality of the relaxation bounds.

Chapter: 3

Experiments

In this chapter we will describe the data we'll use for our experiments, the experimental setup, any configuration and in the end the results.

3.1. Dataset

The Iris dataset is one of the most well-known and frequently used datasets in data science and machine learning. It consists of 150 observations of iris flowers and is often employed for demonstrating various classification, clustering, and visualization techniques. The dataset includes 5 columns, of which four are continuous numerical attributes and one is a categorical target label. The attributes describe the physical characteristics of the flowers, and the dataset is structured as follows:

- Sepal Length: The length of the sepal in centimeters.
- Sepal Width: The width of the sepal in centimeters.
- Petal Length: The length of the petal in centimeters.
- Petal Width: The width of the petal in centimeters.
- Variety: The species of the iris flower, with three possible values: *Setosa*, *Versicolor*, and *Virginica*.

The dataset consists of 50 samples from each of the three species. This balanced composition makes it ideal for classification problems, as it provides an equal number of samples for each class. The four features allow for a variety of clustering and classification approaches, including linear and non-linear models.

3.2 Introduction

This report explores the implementation of a constrained clustering problem, which seeks to identify k centroids and assign each of the n inputs to exactly one centroid to minimize the sum of the L1 distances between each point and its assigned centroid.

3.3 Heuristic Approach to Clustering (A1)

The process of clustering involves grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups. The k-medians algorithm is a popular method used in clustering that seeks to minimize the median distance between points in a

cluster and a given centroid. This report discusses a heuristic approach to implementing the k-medians algorithm, focusing on its application, efficiency, and scalability when applied to large datasets.

3.3.1 Experiments

The experiments were conducted using synthetic datasets to simulate various clustering scenarios. The implementation of the k-medians heuristic was carried out in Python, utilizing libraries such as NumPy for data manipulation and Matplotlib for generating visual results. The experiments were structured to evaluate the performance of the heuristic across different numbers of medians (k = 2, 3, and 4) and varying dataset sizes (15; 20; points).

3.3.2 Experiment and Results

Clustering Quality:

The total L1 distance decreases as the number of clusters *k* increases, indicating improved clustering precision with higher *k*. For n=20 and k=4, the total L1 distance drops significantly to 2.6, suggesting better clustering compactness with more data points and clusters.

Computational Efficiency:

Computation time for the K-medians method remains extremely low across all values of n and *k*, with values generally below 0.005 seconds, indicating high efficiency.

Table: Summary of Results

n	k	method	total_l1_distance	time_taken	random_seed
15	2	K-medians	6.2	0.004007	46
15	3	K-medians	5.2	0.000995	46
15	4	K-medians	5.2	0.000998	46
20	2	K-medians	5.4	0.000999	46
20	3	K-medians	4.8	0.000000	46
20	4	K-medians	2.6	0.002000	46

3.3.3 Graphs This graph (fig- I) illustrates that the K-medians heuristic is both effective and

efficient: higher cluster counts (k) yield lower L1 distances (better clustering), while computation times remain negligible across all configurations. This suggests that the K-medians method can deliver rapid clustering with reasonable quality, making it suitable for large datasets or real-time applications.

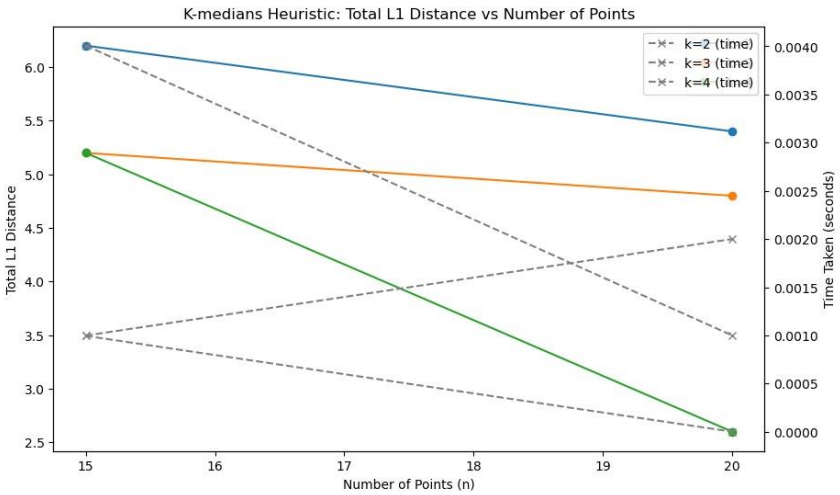

Figure I

3.3.3 Conclusion

The heuristic approach to k-medians clustering demonstrates considerable promise in efficiently handling large datasets, providing a scalable solution that balances between clustering accuracy and computational feasibility. While the increase in the number of medians enhances clustering quality, it also requires greater computational resources. Future improvements could focus on optimizing the algorithm to reduce execution times while maintaining or improving the quality of the clustering. This balance is crucial for practical applications where both speed and precision are valued, such as in real-time data processing and analysis environments.

3.4 Exact Mixed Integer Approach to Constrained Clustering (A2)

The Exact Mixed Integer Linear Programming (MILP) approach (A2) is designed to tackle the constrained clustering problem by optimally solving for k centroids and assigning n inputs to minimize the sum of the L1 distances between each point and its assigned centroid. This method involves formulating the clustering problem as a MILP, utilizing a general-purpose solver to achieve precise and optimal clustering outcomes.

3.4.1 Methodology

The methodology employs the CBC MILP solver, an open-source linear optimization tool, to solve the MILP formulation of the clustering problem. The formulation includes decision variables for each potential cluster assignment and centroid positions, constraints that ensure each data point is assigned to exactly one centroid, and an objective function aimed at minimizing the total L1 distance across all clusters.

3.4.2 Experiments and Results

The CBC MILP approach demonstrates effective clustering potential through decreasing total L1 distances as the cluster count *k* rises. Key findings from the analysis include:

- **Clustering Quality**: As observed in the results, total L1 distance generally decreases with an increasing number of clusters *k*, suggesting that the CBC MILP method provides more refined clustering as *k* increases. For instance, with n=15, the total *L1* distance declines from 2.7 at k=2 to 1.2 at k=4.
- **Impact of Data Points**: When *n* is increased from 15 to 20, the total L1 distance sees a proportional rise for each fixed *k.* This trend points to a correlation between dataset size and clustering compactness, indicating a need for further investigation into how larger data volumes influence clustering at varying *k* values.
- **Scalability Concerns**: The time taken for computation rises dramatically with larger *k* and *n* values, as observed in the 305.96 seconds required for n=20 and k=4. This suggests that the CBC MILP method may struggle with scalability in larger datasets and higher cluster counts, underscoring a need to explore potential algorithmic optimizations or alternative methods for handling high *k* values.

Table: Summary of Result

n	k	method	total_l1_distance	time taken
15	2	CBC MILP	2.7	0.855945
15	3	CBC MILP	1.6	1.967266
15	4	CBC MILP	1.2	7.714633
20	2	CBC MILP	3.9	3.875109
20	3	CBC MILP	2.5	9.017689
20	4	CBC MILP	1.6	305.959156

3.4.3 Graphical Analysis

In the figure II, ***The Time Taken by CBC MILP Solver vs. Number of Data Points*** graph shows that computation time significantly increases as both *n* (number of data points) and *k* (number of clusters) grow. This increase is especially pronounced for k=4, highlighting scalability challenges for the CBC MILP solver with larger datasets and higher cluster counts.

The Total L1 Distance vs. Number of Data Points graph shows that total L1 distance rises with increasing *n* for each fixed *k* value. However, as *k* increases, the total L1 distance decreases, indicating that clustering quality improves with more clusters

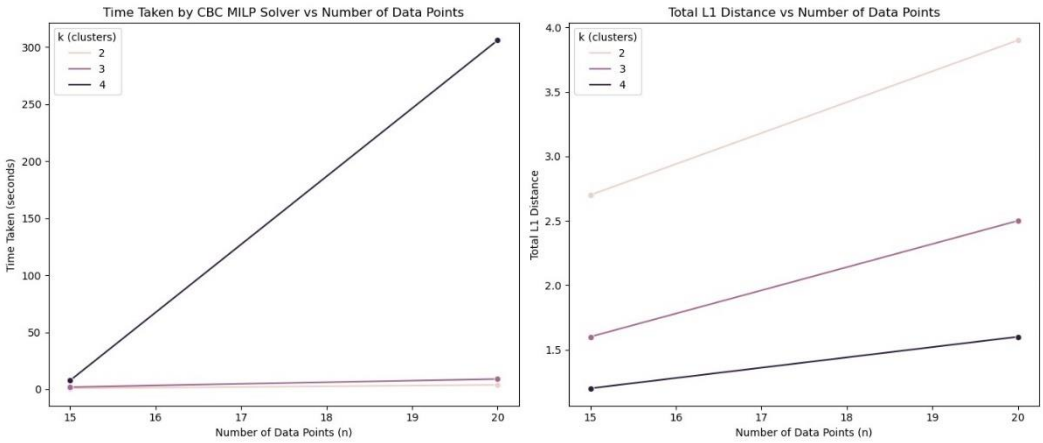

Figure II

3.4.4 Conclusion

The Exact MILP approach (A2) experiments demonstrate that the CBC MILP method effectively minimizes total L1 distance, thereby improving clustering quality with an increased number of clusters (*k*). The reduction in total L1 distance as *k* grows suggests that the method provides more precise clustering results with higher cluster counts. However, the data also reveal a clear scalability limitation: as both the number of data points (*n*) and clusters (*k*) increase, computation time rises sharply, particularly at higher values of *k*, indicating potential inefficiencies in handling larger datasets or cluster counts.

3.5 comparison of Heuristic (A1) and Exact (A2) Approaches

This section compares two methods for solving the constrained clustering problem: the heuristic k-medians approach (A1) and the exact Mixed Integer Linear Programming (MILP) approach (A2). The heuristic approach aims for speed and scalability, providing approximate solutions, while the exact approach focuses on accuracy by finding the optimal solution at a higher computational cost. The comparison examines two key metrics: total L1 distance (quality of clustering) and computation time (efficiency).

3.5.2 GAP Calculation : The GAP in optimization problems, especially in the context of comparing heuristic methods (like K-medians) with exact methods (such as CBC MILP), measures the difference between the solution found by a heuristic method and the optimal solution.

GAP Formula:

GAP (%) = (Heuristic Solution – Optimal Solution (MILP) / Optimal Solution (MILP)) * 100

Heuristic Solution: The total L1 distance (or objective value) obtained using a heuristic approach like K-medians.

Optimal Solution (MILP): The optimal total L1 distance found by solving the problem using an exact method, such as the CBC MILP solver.

GAP (%): The percentage difference between the heuristic and optimal solutions. A lower GAP indicates that the heuristic solution is close to the optimal solution, while a higher GAP suggests that the heuristic is less accurate.

3.5.1 Experiments and Results

This experiment aims to evaluate the performance of the K-medians heuristic approach compared to the CBC MILP exact method in terms of clustering quality (total L1 distance) and computation time across various data points *n* and cluster counts *k*.

- **Clustering Quality:** The K-medians method produces comparable clustering results to the CBC MILP method, with a relatively small gap in total L1 distance. In some cases, the gap percentage is negligible (e.g., 0% at k=3 for both n=15 and n=20), indicating that K-medians achieves similar clustering accuracy as the exact MILP method. For k=4, there is a noticeable increase in the gap, reaching 25% for n=20, showing that the K-medians method may lose some clustering quality at higher values of k.
- **Computational Efficiency:** The K-medians method demonstrates exceptional efficiency, with computation times consistently under 0.02 seconds, regardless of n or k. Conversely, the CBC MILP method requires substantially longer computation times, especially at higher k values. For instance, with n=20 and k=4, the MILP method's computation time reaches 432.31 seconds. This discrepancy highlights the scalability issues of the CBC MILP approach, making it impractical for larger datasets or higher cluster counts without optimization.
- **Practical Implications:** The K-medians heuristic provides a viable alternative for scenarios where computational resources or time are limited, as it produces reasonably accurate clustering results quickly. The CBC MILP method, while delivering slightly better clustering quality in certain cases, faces limitations in handling larger datasets and higher cluster numbers due to its increased computational cost.

Detailed Table of Results

n	k	method	total_l1_distance	time_taken	seed	gap (%)	milp_l1_distance	milp_time
15	2	K-medians	2.9	0.019115	46	7.407407	2.7	0.733132
15	3	K-medians	1.6	0.001385	46	0.000000	1.6	1.628287
15	4	K-medians	1.3	0.000000	46	8.333333	1.2	5.867413
20	2	K-medians	4.4	0.000707	46	12.820513	3.9	2.814519
20	3	K-medians	2.5	0.001146	46	0.000000	2.5	6.810848
20	4	K-medians	2.0	0.001339	46	25.000000	1.6	432.309794

3.5.2 Graph Analysis

Total L1 Distance (fig-III): The CBC MILP method consistently achieves slightly lower L1 distances, indicating more precise clustering. However, K-medians provides comparable results, especially at lower cluster counts, with only a minor increase in L1 distance as n grows.

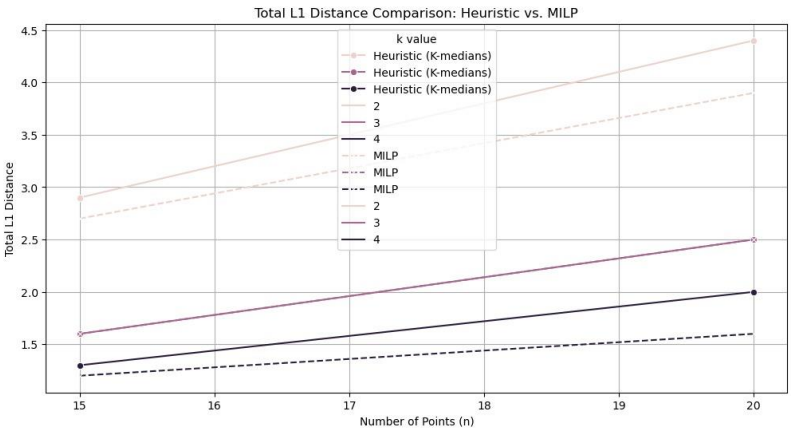

Figure III

Computation Time (fig-IV): The K-medians method shows near-instant computation time, making it highly efficient. In contrast, CBC MILP's computation time rises sharply with larger n and k, especially at k=4 for n=20, where it exceeds 400 seconds, highlighting scalability issues.

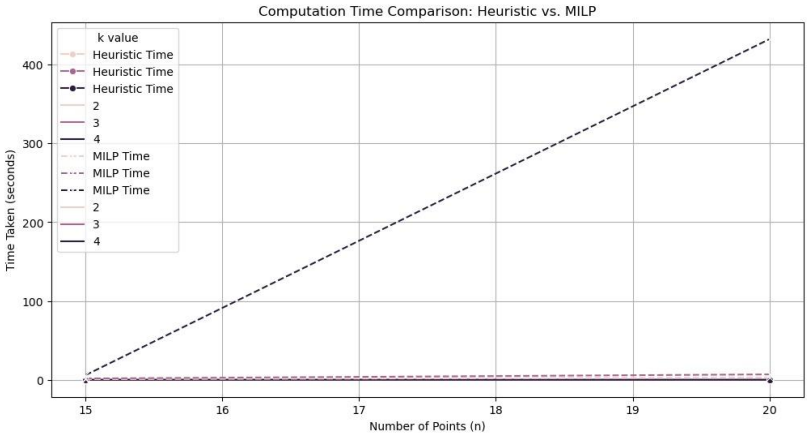

Figure IV

3.5.4 Conclusion

The experiment demonstrates that while the CBC MILP method offers slight improvements in clustering quality, the K-medians heuristic approach achieves comparable results with minimal computation time. For applications where quick clustering is essential, the K-medians method proves to be a practical choice. However, the CBC MILP approach may still be preferred in cases where exact clustering accuracy is prioritized, and computational resources are not a constraint.

References:

[1] Data Clustering: Algorithms and Applications by Charu C. Aggarwal and Chandan K. Reddy

[2] Robust Statistics: The Approach Based on Influence Functions" by Frank Hampel et al.

[3] The Elements of Statistical Learning" by Trevor Hastie, Robert Tibshirani, and JeromeFriedman.

[4] CBC User Guide John Forrest IBM Research Robin Lougee-Heimer (www.coin-or.org)

[5] Data Mining: Concepts and Techniques" by Jiawei Han, Micheline Kamber, and Jian