Evaluating Algorithms for Optimizing McDonald's Outlet Selection Using the P-Median Problem

Lokesh Joshi
22BCS149
Manish Arsiya
22BCS153
Piyush Bihani
22BCS187

Optimization Theory and Application (OE3N37)
April 23, 2025

PDPM IIITDM Jabalpur

Abstract

The p-median problem is a key challenge in location and optimization theory, it seeks the optimal location of p facilities to minimize total distances to demand points. This study aims at applying and comparing various algorithmic strategies, including brute-force, ILP, Greedy heuristics, Partitioning Around Medoids(PAM), Genetic Algorithm (GA), and Simulated Annealing (SA), on a dataset of U.S. McDonald's locations. While exact methods like ILP guarantee optimality, their exponential complexity renders them poor for scalability. Heuristic approaches such as Greedy and PAM improve runtime but trade off solution quality. Metaheuristics, particularly GA and SA, balance scalability and accuracy, with SA slightly outperforming GA as the dataset size or number of medoids (p) increases.

Contents

1 Literature Review **2**

2 Problem Identification and Research Objectives **3**

3 Dataset Description and Visualization **4**
3.1 Data Collection and Preprocessing 4
3.2 Dataset Description 4
3.3 Visualizing Dataset Locations 5

4 Methodology and Solution Techniques **6**
4.1 Distance Computation via Haversine Formula 6
4.2 Mathematical Formulation of the p-Median Problem 6
4.3 Solution Approaches 7
4.3.1 Brute Force Approach 7
4.3.2 Integer Linear Programming (ILP) 8
4.3.3 Greedy Heuristic 8
4.3.4 PAM Heuristic Approach 9
4.3.5 Genetic Algorithm Approach 10
4.3.6 Simulated Annealing Approach 11

5 Results Analysis **13**
5.1 Performance Evaluation and Analysis 13
5.1.1 Effect of Dataset Size (n) 13
5.1.2 Effect of Number of Medoids (p) 14
5.2 Graphical Analysis 15
5.3 Robustness Analysis 17
5.3.1 Robustness with increasing n 17
5.3.2 Robustness with increasing p 17

6 Future Work and Conclusion **19**
6.1 Future Work 19
6.2 Conclusion 19

Bibliography **20**

Chapter 1

Literature Review

The p-median problem was first formulated for a graph-based version by S.L. Hakimi in his 1964 paper [1]. Charles ReVelle and Ralph Swain explored the central facility location problem in their 1970 paper, providing the integer linear programming formulation and discussing strategies for exact optimality [10].

Heuristic methods for solving the p-median problem have been explored in depth historically, with one of the earliest being the work of Teitz and Bart (1968), who developed heuristic methods for estimating the generalized vertex median of a weighted graph. [11].

A comprehensive survey of solution approaches was presented in a 2007 paper by Mladenović et al., focusing on metaheuristic techniques including Ant Colony Optimization and Genetic Algorithms [8].

Recent developments include scalability enhancement techniques such as the Random Sampling and Spatial Voting (RSSV) approach proposed by Mu and Tong in 2020, which significantly reduces runtime while delivering near-optimal results on large-scale instances [9]. Additionally, deep learning-based solutions have emerged, including the AIAM (Adaptive Interactive Attention Model) developed by Liang et al., which applies deep reinforcement learning to solve the p-median problem efficiently [6].

The NP-hardness of the p-median problem has been formally established in the work of Kariv and Hakimi in their 1979 paper [3].

The genetic algorithm as an optimization algorithm was first introduced in optimization by John H. Holland in 1975, with his work *Adaptation in Natural and Artificial Systems* [2]. Whereas, Leonard Kaufman and Peter J. Rousseeuw's 1987 paper on *Clustering by Means of Medoids* introduced the Partitioning Around Medoids (PAM) algorithm [4].

In addition, the concept of simulated annealing as an optimization technique was first proposed by Kirkpatrick, Gelatt, and Vecchi in 1983, who presented it as a probabilistic technique for approximating the global optimum of a function [5].

Chapter 2

Problem Identification and Research Objectives

The problem involves setting p McDonald's hubs, choosing from n ($n > p$) locations of McDonald to cater as a "central distribution hub" network. This is formulated as a p-median problem. Our formulation assumes that

- Every store is assigned to a single hub.
- The set of hubs is a subset of the set of all the stores.
- The road distance between the hubs can be closely approximated by Haversine distance.

Our goal is to minimize the travel distance from the store to the nearest hub. We aim to compare various algorithmic structures and compare them on various metric over our dataset. The focus of this study is to compare the algorithms on 2 vital metrics:- accuracy and runtime. The study seeks to contribute to the domain of facility location optimization, comparing classical approaches such as Integer Linear Programming(ILP), heuristic based approaches such as Greedy Heuristic and meta-heuristic based approaches such as Genetic Algorithm(GA) to identify structures and algorithms best suited for practical applicability, focusing on a balance between accuracy and scalability which are vital for commercial use cases.

Chapter 3

Dataset Description and Visualization

3.1 Data Collection and Preprocessing

We began by loading the McDonald's store location data from a structured CSV file available [7]. The dataset included store IDs, names, addresses, and geographic coordinates (latitude and longitude).

For performance comparison, we created data subsets of various sizes — including the first 50, 100, 150, 200, 250, and 300 entries. These subsets were used to benchmark the accuracy and scalability of different algorithmic structures used.

Each store location was treated as a node in the network, and pairwise distances were later computed through Haversine formula to build the distance matrices.

3.2 Dataset Description

Table 3.1: A Subset of USA McDonald's Locations Used for the Evaluation

Store ID	Name	Address	City	State	Zipcode	Phone	Latitude	Longitude
1183	KEY WEST-ROOSVLT	3704 N Roosevelt Blvd	Key West	FL	33040	(305) 293-7606	24.5719	-81.7569
10927	MARATHON	5595 Overseas Hwy	Marathon	FL	33050	(305) 743-0707	24.7164	-81.0733
5090	TAVERNIER	91400 Overseas Hwy	Tavernier	FL	33070	(305) 852-2463	25.0046	-80.5226
37956	KEY LARGO	101000 Overseas Hwy	Key Largo	FL	33037	(305) 451-5861	25.1087	-80.4273
10537	FLORIDA CITY	425 Se 1st Ave	Florida City	FL	33034-5009	(305) 248-9285	25.4426	-80.4746
36983	NORTH HOMESTEAD	13690 Biscayne Dr	Homestead	FL	33033	(305) 247-2704	25.4992	-80.4119
6103	NARANJA	26601 Us Hwy 1	Naranja	FL	33032	(305) 258-0341	25.5194	-80.4261
34014	SIVLER PALMS	23351 Sw 112th Ave	Homestead	FL	33032	(305) 258-7837	25.5503	-80.3716
31341	HOMESTEAD, CAMPB	2880 Ne 8th St	Homestead	FL	33033	(305) 242-9774	25.4770	-80.4346
1108	HOMESTEAD	30335 S Federal Hwy	Homestead	FL	33030	(305) 247-6484	25.4852	-80.4598

3.3 Visualizing Dataset Locations

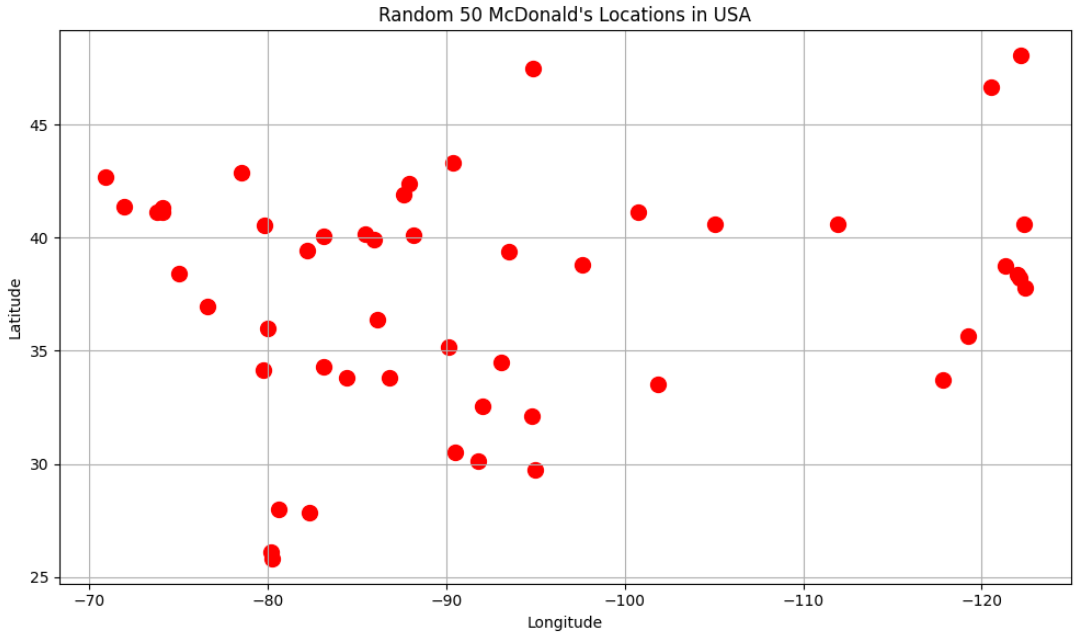

Figure 3.1: Scatter plot of 50 randomly selected McDonald's locations in the USA.

Chapter 4

Methodology and Solution Techniques

4.1 Distance Computation via Haversine Formula

To construct the $n \times n$ distance matrix, we used the Haversine formula [12] to compute "great-circle distance" between 2 locations using their latitude and longitude values. This method was preferred due to its simplicity and effective approximation.

Each distance d_{ij} between store i and store j was calculated as:

$$d_{ij} = 2R \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

Where:

- $R = 6371$ km is the Earth's radius.
- $\Delta\phi = \phi_2 - \phi_1$, $\Delta\lambda = \lambda_2 - \lambda_1$
- ϕ and λ are the latitudes and longitudes in radians.

4.2 Mathematical Formulation of the p-Median Problem

The p-median problem aims to choose p facilities (hubs) from a set of n locations such that the sum of distances from all demand points to their assigned facilities is minimized. To formulate our problem in hand as a p-median problem

Let:

- $x_{ij} = 1$ if location i is assigned to facility j, else 0.
- $y_j = 1$ if location j is selected as a facility, else 0.
- d_{ij} = Haversine distance from location i to location j.

Objective:

$$\text{Minimize:} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} \cdot x_{ij}$$

Subject to:

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \qquad \text{(Each location assigned to exactly one facility)}$$

$$x_{ij} \leq y_j \quad \forall i, j \qquad \text{(Can only assign to an open facility)}$$

$$\sum_{j=1}^{n} y_j = p \qquad \text{(Exactly } p \text{ facilities are selected)}$$

$$x_{ij}, y_j \in \{0, 1\} \quad \forall i, j \qquad \text{(Binary decision variables)}$$

4.3 Solution Approaches

We performed an initial analysis for $n = 50$ to $n = 300$ with step values of 50 to study the effectiveness of six algorithms as solution techniques for our problem: Brute Force, ILP, Greedy Heuristic, PAM Heuristic, Genetic Algorithm (GA), and Simulated Annealing (SA). These algorithms were categorized as follows:

- **Exact Methods:** Brute Force, ILP
- **Heuristic Methods:** Greedy Heuristic, PAM Heuristic
- **Metaheuristic Methods:** Genetic Algorithm (GA), Simulated Annealing (SA)

This initial analysis helped us assess the runtime performance and solution quality of each algorithm. Based on the results of this analysis, we narrowed our focus to the algorithms that demonstrated computationally feasible performance in terms of both efficiency and solution optimality for larger datasets. These selected algorithms were then subjected to a large-scale analysis.

4.3.1 Brute Force Approach

Firstly, we implemented a brute force method. It checks all $\binom{n}{p}$ combinations of potential medians and evaluates the total assignment cost for each combination.

This method provides an exact solution, but it is computationally infeasible except for small values of n, due to the factorial growth in the number of combinations. Due to time constraints, it was tested only up to $n = 300$.

Time Complexity: $\mathcal{O}\left(\binom{n}{p} \cdot n \cdot p\right)$

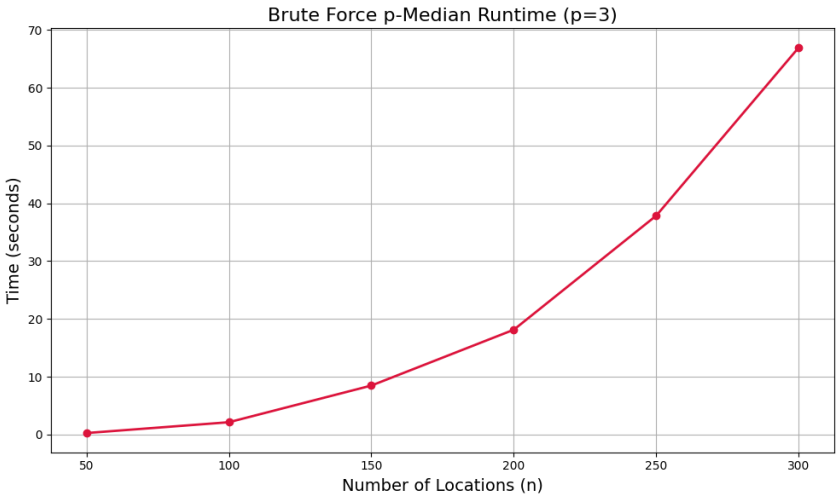

Figure 4.1: Runtime performance of brute force approach

4.3.2 Integer Linear Programming (ILP)

We implemented an exact Integer Linear Programming (ILP) formulation of the p-median problem using the PuLP library. This model aims to minimize the total distance cost by introducing binary variables as mentioned in the formulation. The ILP was solved using the CBC solver. This method guarantees the optimal solution, but being no better than the brute force approach at runtime, it is not scalable for larger datasets. It was unable to handle larger datasets as runtime grew substantially beyond $n = 150$.

Time Complexity: Exponential in the worst case due to binary variables and combinatorial constraints.

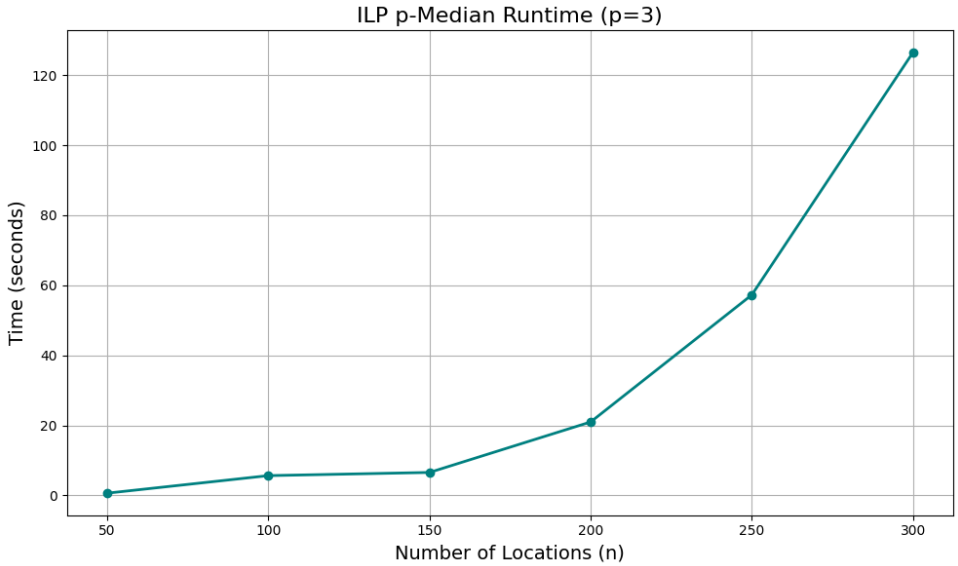

Figure 4.2: ILP p-Median Runtime(p=3)

4.3.3 Greedy Heuristic

The Greedy Heuristic method is an iterative approach in which we start with an empty set of selected medians, then after each iteration, we add the location that minimizes the

total distance to the chosen medians. This continues until the desired number of medians is selected.

The Greedy Heuristic does not guarantee the optimal solution but it is significantly faster than exact methods like ILP, especially for larger datasets. This heuristic was applied to each subset of the data and compared to the ILP benchmarks to test the difference in runtime as well as solution provided by the 2 methods.

Time Complexity: $O(n^2 \cdot p)$, where n is the number of locations and p is the number of medians.

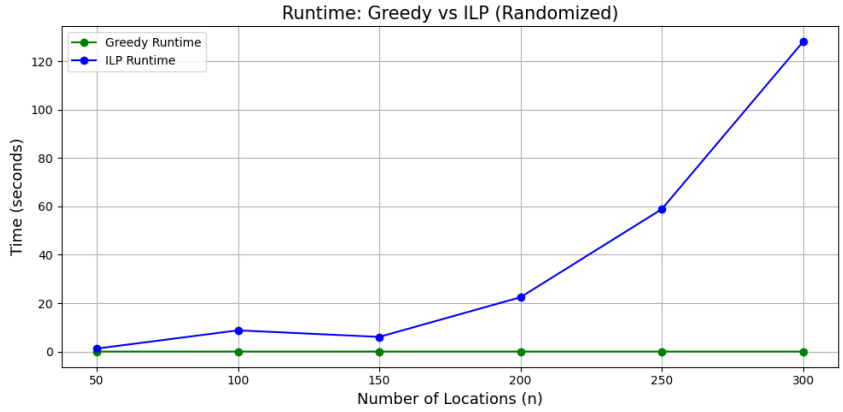

Figure 4.3: Runtime Comparison: Greedy Heuristic vs ILP for $p = 3$

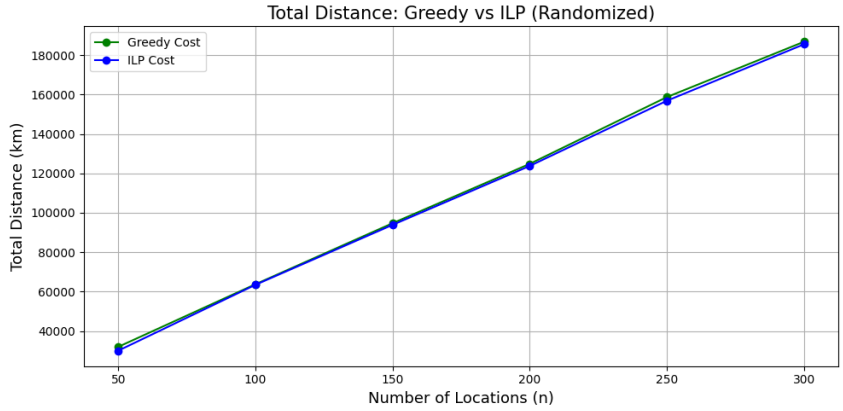

Figure 4.4: Solution Optimality Comparison: Greedy Heuristic vs ILP for $p = 3$

4.3.4 PAM Heuristic Approach

We also implemented a clustering heuristic method, Partitioning Around Medoids (PAM) algorithm.

PAM starts with a randomly selected set of facilities(called medoids), it works by iteratively swapping medoids with non-medoids to minimize total cost(distance). While not guaranteed to be optimal, PAM typically offers much better runtime efficiency than the ILP and accuracy with respect to optimal solution is greatly improved as compared to the greedy heuristic approach.

Time Complexity: $O(n^2 \cdot p)$ per iteration

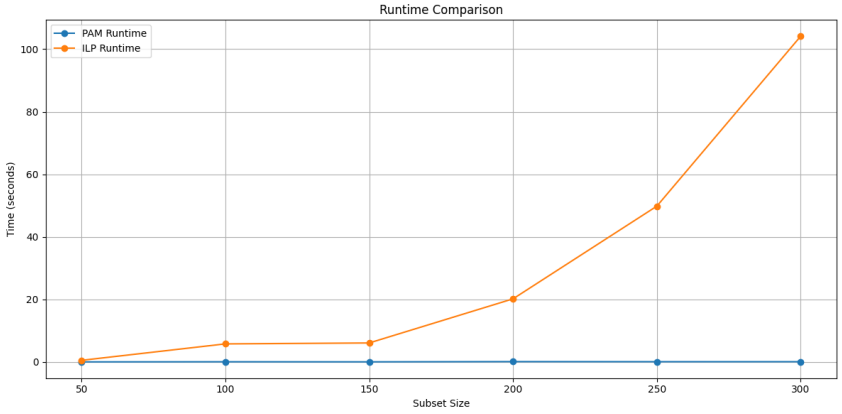

Figure 4.5: Runtime Comparison: PAM vs ILP for $p = 3$

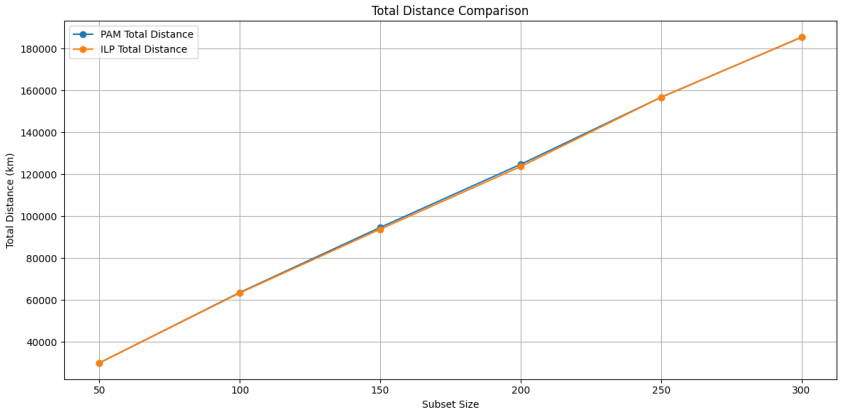

Figure 4.6: Solution Optimality Comparison: PAM vs ILP for $p = 3$

4.3.5 Genetic Algorithm Approach

We implemented a population-based metaheuristic algorithm, called the Genetic Algorithm (GA). It is an algorithm that evolves a population of candidate medoid sets through selection, crossover, and mutation to minimize the total distance between all points and their nearest selected medoid.

GA begins with a random population of medoid sets, similar to PAM, then the algorithm iteratively selects the fittest candidates based on total cost(distance). Crossover operations combine partial solutions from the two parents and mutation introduces diversity by randomly changing a medoid, simulating an environment of natural selection.

Although GA does not guarantee optimal results, it demonstrates great scalability along with a near-optimal accuracy.

We tested the algorithm with hyperparameters as $p = 3$, using 100 generations and a population size of 500 and a mutation ratio of 0.2.

Time Complexity: Approx. $O(g \cdot s \cdot p \cdot n)$ per generation, where g is the number of generations and s is the size of the population.

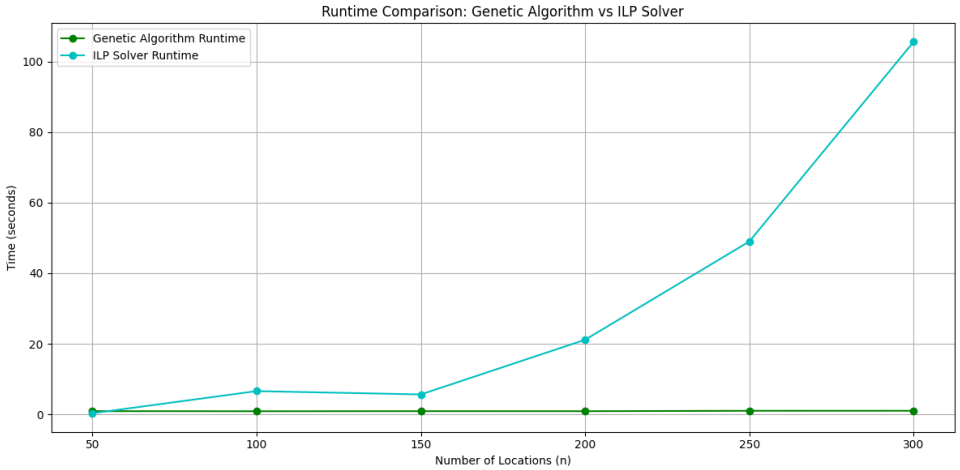

Figure 4.7: Runtime Comparison: Genetic Algorithm vs ILP for $p = 3$

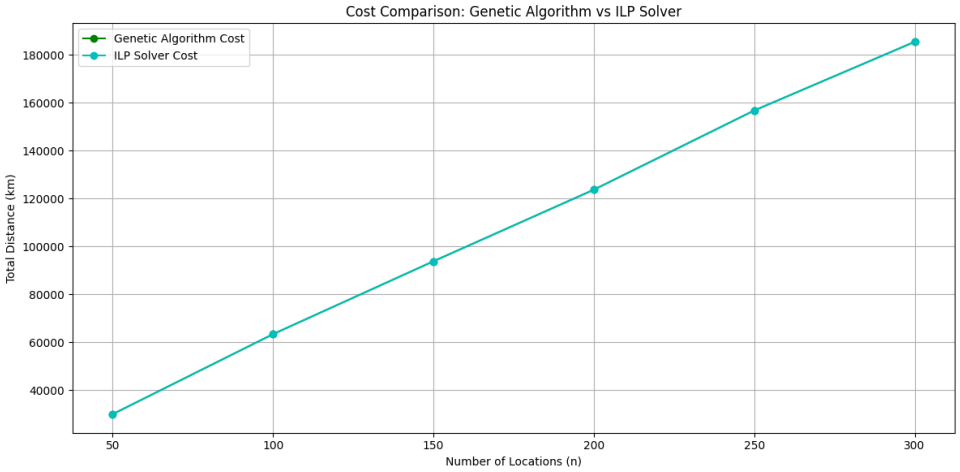

Figure 4.8: Solution Optimality Comparison: Genetic Algorithm vs ILP for $p = 3$

4.3.6 Simulated Annealing Approach

We finally implemented the Simulated Annealing (SA) algorithm, a probabilistic metaheuristic based on process of annealing in metallurgy. At each step, the algorithm randomly selects a neighboring solution which is accepted on the basis of a temperature-dependent probability. The temperature decreases over time, reducing the chances of acceptance of worse solutions in subsequent steps.

SA starts with a random set of medoids and iteratively explores neighboring solutions. If a neighboring solution improves the total cost, it is accepted. If not, its acceptance depends on the value of temperature hyperparameter which decreases as the algorithm progresses.

SA showed similar performance to the GA technique for $n <= 300$ in our initial analysis phase.

We tested the algorithm with hyperparameters as $p = 3$, an initial temperature of $T_0 = 1000$, and a cooling rate $\alpha = 0.90$. The algorithm runs for 100 iterations per temperature, with a minimum temperature of 1×10^{-2}.

Time Complexity: Approx. $O(i \cdot p \cdot n)$ where i is the number of iterations, p is the number of medoids, and n is the number of points in the dataset.

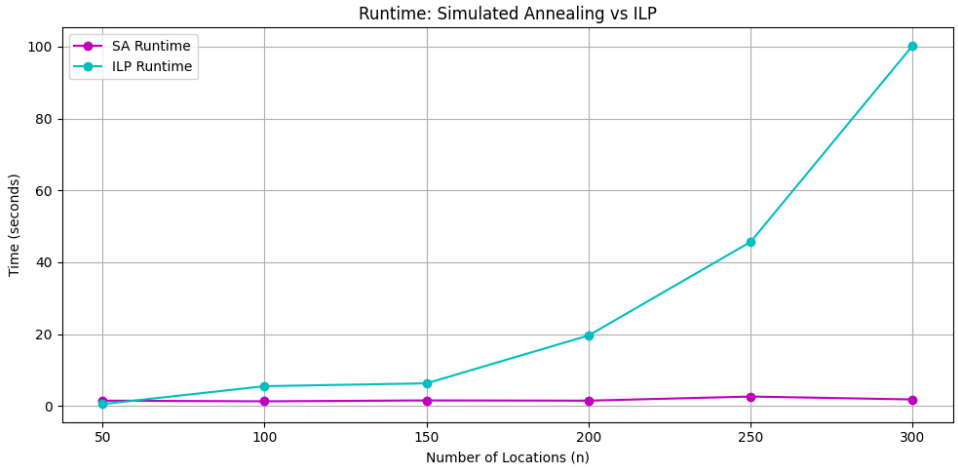

Figure 4.9: Runtime Comparison: Simulated Annealing vs ILP for $p = 3$

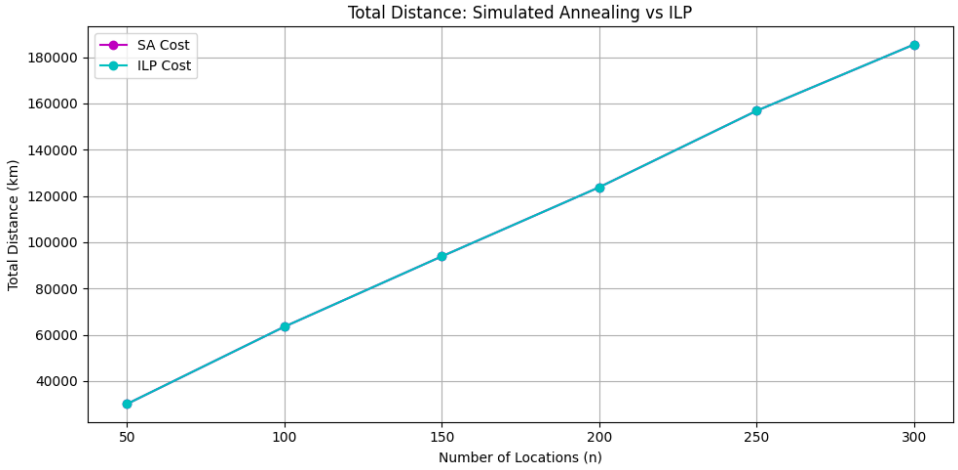

Figure 4.10: Solution Optimality Comparison: Simulated Annealing vs ILP for $p = 3$

Based on our initial analysis, we selected Greedy Heuristic, Partitioning Around Medoids (PAM), Genetic Algorithm (GA), and Simulated Annealing (SA) for a large-scale detailed analysis due to their scalability for larger dataset sizes and near-optimal performances exhibited for sizes $n <= 300$.

Chapter 5

Results Analysis

5.1 Performance Evaluation and Analysis

After initial analysis, we compare the performance of four methods—Greedy Heuristic, PAM, GA, and SA—under two in-depth experimental setups: varying dataset size n (with $p = 3$) and varying number of medoids p (with $n = 1000$).

5.1.1 Effect of Dataset Size (n)

We fixed number of medoids at $p = 3$ and sampled datasets of size $n = 1000, 1250, \ldots, 4750$. Tables 5.1 and 5.2 summarize the execution time and resulting distance cost for each method.

Execution Time

n	Greedy (s)	PAM (s)	GA (s)	SA (s)
1000	0.08	1.31	2.84	3.71
1250	0.16	1.80	4.34	4.85
1500	0.23	1.82	5.66	4.88
1750	0.35	5.17	6.86	5.93
2000	0.46	9.22	7.40	7.66
2250	0.73	6.60	8.70	8.22
2500	0.76	6.61	9.45	8.96
2750	0.89	10.05	10.62	10.02
3000	1.11	28.42	11.92	11.04
3250	1.51	21.00	13.02	12.14
3500	1.52	34.86	14.34	13.02
3750	1.81	17.59	15.33	14.01
4000	2.07	22.02	16.41	14.41
4250	2.44	19.32	17.54	16.04
4500	2.82	39.48	19.95	16.05
4750	3.44	31.08	21.44	17.02

Table 5.1: Execution times for varying dataset sizes ($p = 3$).

Solution Cost

n	Greedy (km)	PAM (km)	GA (km)	SA (km)
1000	628748.1	619262.9	621917.3	616372.9
1250	778210.8	767858.0	769950.9	772512.1
1500	938625.7	914359.7	915931.9	916841.8
1750	1086017.0	1073634.0	1064200.0	1068210.0
2000	1250223.0	1220241.0	1220298.0	1220241.0
2250	1410895.0	1386514.0	1372497.0	1374945.0
2500	1559630.0	1529101.0	1523457.0	1526653.0
2750	1717571.0	1689167.0	1689296.0	1689296.0
3000	1880305.0	1831273.0	1831484.0	1844372.0
3250	2041635.0	1995818.0	1984780.0	1991765.0
3500	2194522.0	2136760.0	2140417.0	2150138.0
3750	2349712.0	2287330.0	2300743.0	2306882.0
4000	2504360.0	2446643.0	2453112.0	2432638.0
4250	2650719.0	2576890.0	2588653.0	2596942.0
4500	2798150.0	2725382.0	2732616.0	2743418.0
4750	2949551.0	2877388.0	2881977.0	2888896.0

Table 5.2: Total distance cost for varying dataset sizes ($p = 3$).

5.1.2 Effect of Number of Medoids (p)

Next, we fix $n = 1000$ and vary the number of medoids $p = 5, 10, 15, 20, 25$. Tables 5.3 and 5.4 describe the change in execution time and distance cost with respect to p for each method.

Execution Time

p	Greedy (s)	PAM (s)	GA (s)	SA (s)
5	0.18	4.54	4.15	5.17
10	0.59	30.93	7.66	6.46
15	1.49	77.74	10.12	9.82
20	2.18	53.43	12.93	12.22
25	3.65	97.04	16.37	14.10

Table 5.3: Execution time for varying number of medoids ($n = 1000$).

Solution Cost

p	Greedy (km)	PAM (km)	GA (km)	SA (km)
5	449334.45	413024.15	413024.15	413096.16
10	281784.15	265763.64	262297.90	261829.95
15	211256.44	198447.86	196751.66	195325.06
20	172573.46	178614.66	160368.72	159100.55
25	148449.76	164910.83	137557.51	136338.95

Table 5.4: Total distance cost for varying number of medoids ($n = 1000$).

5.2 Graphical Analysis

In addition to the tabular data, we present the following graphs for better visualization of the performance of the four methods under the experimental setups.

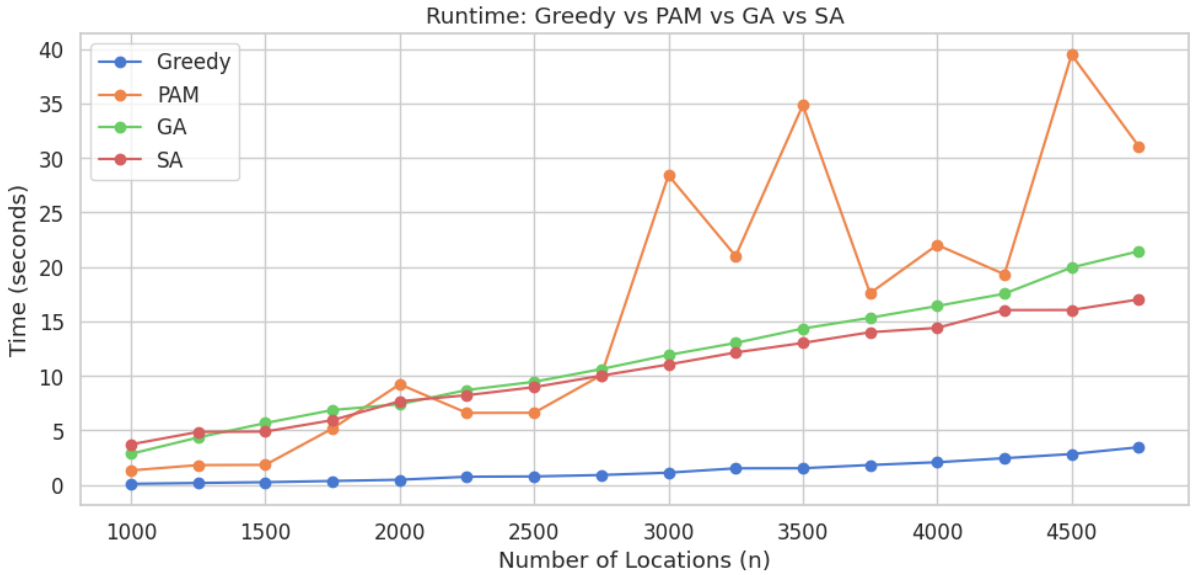

Figure 5.1: Runtime vs Dataset Size (n) for $p = 3$.

Figure 5.2: Solution Cost vs Dataset Size (n) for $p = 3$.

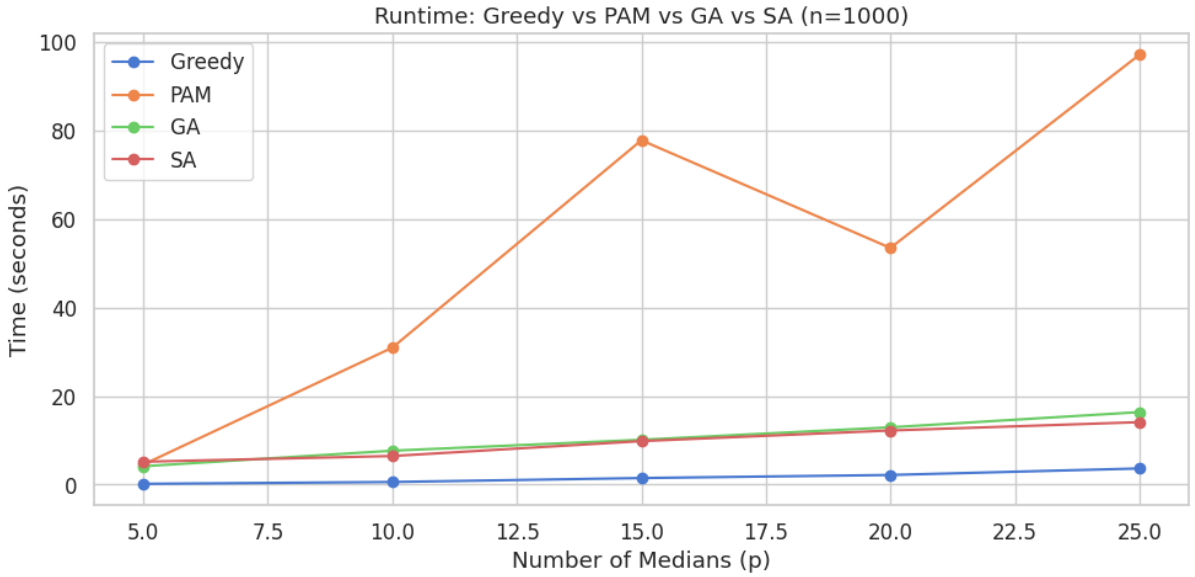

Figure 5.3: Runtime vs Number of Medoids (p) for $n = 1000$.

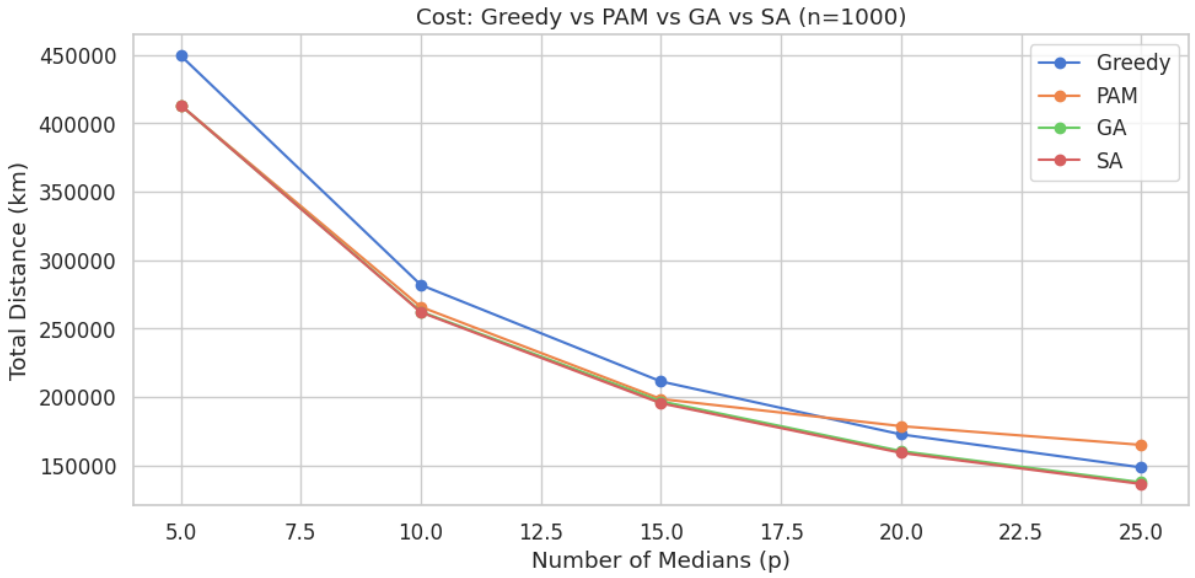

Figure 5.4: Solution Cost vs Number of Medoids (p) for $n = 1000$.

5.3 Robustness Analysis

5.3.1 Robustness with increasing n

- **Greedy**
  - Runtime grows nearly linearly.
  - Fast but sub-optimal solutions, with increasing deviation from optimality as dataset size increases.
- **PAM**
  - Runtime escalates unpredictably due to data dependence.
  - Cost remains best for higher values of n but suffers from excessive execution time.
- **Genetic Algorithm (GA)**
  - Runtime increases moderately and predictably.
  - Maintains good solution quality.
- **Simulated Annealing (SA)**
  - Moderate and predictable runtime growth
  - Solution quality similar to GA.

5.3.2 Robustness with increasing p

- **Greedy**
  - Runtime rises steadily(quadratically).

  - Poor solution quality for high values of p.

- **PAM**
  - Runtime is very high and changes unpredictably due to data dependence.
  - Very poor solution quality for higher values of p.

- **Genetic Algorithm (GA)**
  - Runtime scales predictably and linearly.
  - Exhibits good solution quality even at large p.

- **Simulated Annealing (SA)**
  - Runtime similar to GA
  - Yields best solution at higher values of p.

Chapter 6

Future Work and Conclusion

6.1 Future Work

Building upon the current research, several advanced topics can be explored to improve the efficiency and scalability of the solution methods:

- **Evaluation of Latest Techniques:** Recent advancements, such as Random Sampling and Spatial Voting (RSSV)[3] and deep learning-based approaches[6], offer alternative techniques for solving the p-median problem. Evaluating their effectiveness and scalability could provide valuable insights.
- **Evaluation of Hybrid Algorithms:** Combining the strengths of different methods, such as integration of latest methods with traditional metaheuristics could lead to significant performance improvements.
- **In-depth Parameter Tuning for Metaheuristics:** Systematic experiments to optimize parameters of Genetic Algorithm (GA) and Simulated Annealing (SA) can prove to be essential for improving the solution quality and reducing the computational time complexity.

6.2 Conclusion

In this study, we evaluated six solution methods—Brute Force (BF), Integer Linear Programming (ILP), Greedy Heuristic, Partitioning Around Medoids (PAM), Genetic Algorithm (GA), and Simulated Annealing (SA)—across various performance parameters. Our analysis concluded that metaheuristic techniques outperformed heuristic techniques in both robustness and solution quality. Additionally, SA demonstrated a marginal advantage over GA with the increase in number of medoids (p).

While exact algorithms, such as BF and ILP, guarantee optimal solutions, their exponential time complexity renders them impractical for real-world uses, given the NP-hard nature of the p-median problem.Heuristic and metaheuristic methods provide near-optimal solutions within practical computational time. Among these, metaheuristics like GA and SA were found to be more effective demonstrating superior performance with the increase in size of the dataset or the number of medoids. These findings highlight the practical utility of metaheuristic approaches in solving large-scale p-median optimization problems, offering efficient solution quality and computational efficiency.

Bibliography

[1] S. L. Hakimi. "Optimum locations of switching centers and the absolute centers and medians of a graph". In: *Operations Research* 12.3 (1964). DOI: 10.1287/opre.12.3.450.

[2] John H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.

[3] O. Kariv and S. L. Hakimi. "An Algorithmic Approach to Network Location Problems. II: The p-Medians". In: *SIAM Journal on Applied Mathematics* 37.3 (1979), pp. 539–560. DOI: 10.1137/0137041.

[4] Leonard Kaufman and Peter J. Rousseeuw. "Clustering by Means of Medoids". In: *Statistical Data Analysis Based on the L1-Norm and Related Methods* (1987), pp. 405–416.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by simulated annealing". In: *Science* 220.4598 (1983), pp. 671–680. DOI: 10.1126/science.220.4598.671.

[6] Haojian Liang et al. "AIAM: Adaptive interactive attention model for solving p-Median problem via deep reinforcement learning". In: *International Journal of Applied Earth Observation and Geoinformation* 138 (Apr. 2025), p. 104454. DOI: 10.1016/j.jag.2025.104454.

[7] Jacopo Mazzoni. *McDonald's Locations 2025 Dataset*. https://www.kaggle.com/datasets/jacopomazzoni/mcdonalds-locations-2025. Accessed: 2025-04-23. 2024.

[8] Nenad Mladenović et al. "The p-median problem: A survey of metaheuristic approaches". In: *European Journal of Operational Research* 179.3 (2007). DOI: 10.1016/j.ejor.2005.05.034.

[9] Wangshu Mu and Daoqin Tong. "On solving large p-median problems". In: *Environment and Planning B: Urban Analytics and City Science* 47.6 (2020). DOI: 10.1177/2399808319892598.

[10] Charles S. ReVelle and Ralph W. Swain. "Central facilities location". In: *Geographical Analysis* 2.1 (1970). DOI: 10.1111/j.1538-4632.1970.tb00142.x.

[11] M. B. Teitz and P. Bart. "Heuristic Methods for Estimating the Generalized Vertex Median of a Weighted Graph". In: *Operations Research* 16.5 (1968), pp. 955–961. DOI: 10.1287/opre.16.5.955.

[12] Wikipedia contributors. *Haversine formula*. Accessed: 2025-04-23. 2025. URL: https://en.wikipedia.org/wiki/Haversine_formula.