15/20

Computational Experience with Heuristics for the p-Median Problem

---

Marcelo Treviño, Sebastian Mayorga, Jesus R. Gómez, Luis F. Medellin

Facultad de Ingeniería Mecánica y Eléctrica, UANL

Selected Topics on Optimization

Roger Z. Rios, Ph.D.

March 24th, 2024

Introduction

The p-Median problem is a fundamental optimization challenge with practical implications spanning urban planning, logistics, and healthcare management. This problem revolves around determining the optimal locations for a specified number of facilities, such as warehouses or hospitals, to efficiently serve a dispersed population.

Research efforts have yielded various methodologies, from mathematical models to computational algorithms, aimed at solving this problem. Noteworthy contributions include the seminal works of Hakimi (1964) and Church and ReVelle (1974), which have laid the groundwork for subsequent advancements in this field.

The significance of the p-Median problem lies in its ability to inform decision-making processes across diverse domains. In urban planning, it aids in the strategic placement of public amenities and emergency services to enhance accessibility and alleviate congestion. Similarly, in logistics and supply chain management, it facilitates the optimization of distribution networks and inventory management strategies, thereby improving operational efficiency.

Moreover, the p-Median problem finds practical applications in healthcare delivery, guiding the strategic siting of medical facilities to ensure equitable access and enhance patient outcomes. By identifying optimal locations for clinics, hospitals, and specialized care centers, healthcare providers can better allocate resources and address community needs effectively.

In essence, the p-Median problem serves as a valuable tool for making informed decisions that impact societal well-being and resource allocation in various sectors.

Keywords: p-Median, optimization, mathematical model, algorithm, urban planning, logistics.

Problem description

The p-Median problem involves making strategic decisions regarding the placement of facilities to efficiently serve a dispersed population. This problem can be defined by four fundamental components:

1. Data

The input consists of:

- The location of potential facility sites.
- The spatial distribution of the population or demand points that these facilities are intended to serve.
- The costs or distances associated with providing service from each facility site to every demand point.

2. Decisions

The decision involves selecting p locations out of the potential facility sites where facilities will be placed.

3. Optimization

The objective is to minimize the total cost or distance required to provide service from the chosen facility locations to all demand points. Mathematically, this can be represented as minimizing the sum of the distances (or costs) between each demand point and its assigned facility, where the assignment is based on proximity.

4. Constraints

The feasible solution is defined by the requirement that each demand point must be assigned to exactly one facility. Additionally, exactly p facilities must be selected from the potential facility sites.

Mathematical method

The p-Median problem can be formulated as an integer linear programming (ILP) model. Let's denote:

- n as the number of potential facility sites.
- m as the number of demand points.
- p as the number of facilities to be selected.

We can use binary decision variables x_i to indicate whether facility i is selected or not, and binary decision variables y_{ij} to represent whether demand point j is assigned to facility i. The objective function can be formulated as minimizing the total distance or cost, which is the sum of the distances (or costs) multiplied by the assignment variables y_{ij}. The constraints ensure that each demand point is assigned to exactly one facility and that exactly p facilities are selected.
The ILP formulation of the p-Median problem can be expressed as follows:

Minimize:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} y_{ij}$$

Subject to:

$$\sum_{i=1}^{n} x_i = p$$

$$\sum_{i=1}^{n} y_{ij} = 1,\ \forall_j = 1,\ldots,m$$

$$y_{ij} \leq x_i,\ \forall_i = 1,\ldots,n,\ \forall_j = 1,\ldots,m$$

$$x_i \in \{0,\ 1\},\ \forall_i = 1,\ldots,n$$

$$y_{ij} \in \{0,\ 1\},\ \forall_i = 1,\ldots,n,\ \forall_j = 1,\ldots,m$$

Problem example

Let's consider a simplified scenario to illustrate the p-Median problem. Suppose we have a small town with 5 potential locations for a new healthcare clinic (facilities) and 8 demand points representing households in the town that need medical services. We want to select 2 clinic locations out of the 5 potential locations to minimize the total distance traveled by residents to access healthcare services.

Data

- Potential clinic locations (facilities): A, B, C, D, E
- Demand points (households): 1, 2, 3, 4, 5, 6, 7, 8
- Distance matrix (in kilometers) between potential clinic locations and demand points:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
A	3	5	2	7	4	6	5	8
B	6	8	4	9	3	7	6	10
C	5	7	3	8	2	6	4	9
D	4	6	2	7	3	5	3	8
E	2	4	1	6	2	4	2	7

Problem illustration

- ***Facility selection:*** Suppose we select clinics at locations B and D
- ***Assignment of demand points:*** Assign demand points to the nearest clinic locations based on the distance matrix

Demand Point	**Nearest**
1	D
2	D
3	D
4	D
5	B
6	D
7	D
8	D

- ***Objective Function Evaluation:*** The objective function evaluates the total distance traveled by residents to access healthcare services from the selected clinic locations. Let's calculate this for our example:

  Total distance = 6 + 8 + 3 + 2 + 3 + 2 + 3 + 7 = 34 kilometers

  So, for this feasible solution, the total distance traveled by residents is 34 kilometers.

Description of Heuristics

No constructive heuristic was used, but instead the Genetic Algorithm was used. It employs principles of natural evolution, such as selection, crossover, and mutation, to evolve a population of solutions over multiple generations. Initially, a population of random solutions is generated, each representing a set of *p* facilities. The fitness of each solution is evaluated based on the total distance to the demand points. The best-performing solutions are selected as parents to produce the next generation through crossover, which combines parts of two parents to create a child solution. Mutation introduces small random changes to some solutions to maintain genetic diversity. This process iterates over several generations, with the goal of converging to an optimal or near-optimal solution. While more

computationally intensive than the Greedy Heuristic, the Genetic Algorithm can explore a broader solution space and potentially find better solutions.

Local search can be used in conjunction with the Genetic Algorithm to improve the solutions generated during the evolutionary process. This hybrid approach, often referred to as a memetic algorithm, combines the global search capability of the Genetic Algorithm with the fine-tuning ability of local search to enhance solution quality.

In the context of the p-Median problem, once the Genetic Algorithm generates a new solution through crossover and mutation, a local search procedure can be applied to this solution before it is added to the population. The local search operates by exploring the neighborhood of the current solution. For each facility location in the solution, the algorithm evaluates the potential benefit of swapping it with a non-selected location. It performs these swaps iteratively, selecting the swap that results in the greatest reduction in the total distance between demand points and their nearest facility, until no further improvements can be made.

This integration of local search allows the Genetic Algorithm to refine each solution, ensuring that each offspring is locally optimized before competing with other solutions in the population. As a result, the overall population maintains high-quality solutions, leading to faster convergence and better final results. By combining the exploratory power of the Genetic Algorithm with the exploitative strength of local search, this hybrid approach can effectively tackle the p-Median problem, balancing global and local optimization to achieve superior outcomes.

Computational Work

The computational experiments for the p-median problem were conducted in two main phases. The first phase involved using a genetic algorithm heuristic to solve the problem for three different sample sizes. The second phase utilized a genetic algorithm followed by a local search to refine the solutions obtained from the genetic algorithm. This section details the experiments conducted, the results obtained, and an analysis of the performance improvements achieved through the local search.

Experiment 1. Genetic Algorithm Heuristic

The genetic algorithm heuristic was applied to the p-median problem across three different sample sizes: 1,000, 10,000, and 20,000. Each sample size included 20 instances, and the parameters for each instance (size of "n", size of "m", and size of "p") were kept consistent across the instances within each sample size. The instances with size n=1,000 used p=10 and m=3 as parameters, the instances with size n=10,000 used p=25 and m=5 as parameters, and the instances with size n=20,000 used p=50 and m=10 as parameters.

Instance	Size of "n"	Size of "m"	Size of "p"	Genetic Algoritm Heuristic	Time
Data1	1000	10	3	7903.4	35.34s
Data2	1000	10	3	7601.9	34.84s
Data3	1000	10	3	7207.1	35.34s
Data4	1000	10	3	8072.1	33.40s
Data5	1000	10	3	7605.5	34.50s
Data6	1000	10	3	8166.1	33.82s
Data7	1000	10	3	7451.7	37.34s
Data8	1000	10	3	7812.6	34.49s
Data9	1000	10	3	7351.0	33.07s
Data10	1000	10	3	7236.9	34.01s
Data11	1000	10	3	7672.1	33.19s
Data12	1000	10	3	7208.3	33.66s
Data13	1000	10	3	8779.8	33.63s
Data14	1000	10	3	7588.8	34.00s
Data15	1000	10	3	7662.5	34.07s
Data16	1000	10	3	7398.3	33.83s
Data17	1000	10	3	8543.4	33.24s
Data18	1000	10	3	7484.4	33.69s
Data19	1000	10	3	7800.2	33.75s
Data20	1000	10	3	8039.3	31.17s

Figure 1. Experiment 1.1 with size n=1,000

Instance	Size of "n"	Size of "m"	Size of "p"	Genetic Algoritm Heuristic	Time S
Data1	10 000	25	5	116121.4	161.20s
Data2	10 000	25	5	121180.7	163.64s
Data3	10 000	25	5	104109.8	173.08s
Data4	10 000	25	5	97387.5	158.22s
Data5	10 000	25	5	100827.7	167.77s
Data6	10 000	25	5	112131.8	176.98s
Data7	10 000	25	5	102062.3	165.75s
Data8	10 000	25	5	112476.4	175.30s
Data9	10 000	25	5	109332.4	158.51s
Data10	10 000	25	5	104273.0	182.33s
Data11	10 000	25	5	102693.9	176.25s
Data12	10 000	25	5	125373.3	189.28s
Data13	10 000	25	5	107250.2	153.39s
Data14	10 000	25	5	118078.3	183.43s
Data15	10 000	25	5	126643.8	186.03s
Data16	10 000	25	5	130483.2	170.55s
Data17	10 000	25	5	106921.5	169.89s
Data18	10 000	25	5	111971.0	30.573s
Data19	10 000	25	5	104941.1	147.60s
Data20	10 000	25	5	104947.9	190.87s

Figure 2. Experiment 1.2 with size n=10,000

Instance	Size of "n"	Size of "m"	Size of "p"	Genetic Algoritm Heuristic	Time S
Data1	20 000	50	10	230504.4	279.96s
Data2	20 000	50	10	231840.9	231.16s
Data3	20 000	50	10	228598.7	238.02s
Data4	20 000	50	10	248845.8	254.57s
Data5	20 000	50	10	219550.5	235.21s
Data6	20 000	50	10	223935.4	260.71s
Data7	20 000	50	10	199164.8	283.47s
Data8	20 000	50	10	217348.5	245.76s
Data9	20 000	50	10	250451.9	229.12s
Data10	20 000	50	10	212493.1	250.24s
Data11	20 000	50	10	203685.4	246.56s
Data12	20 000	50	10	205393.7	241.94s
Data13	20 000	50	10	200296.8	251.51s
Data14	20 000	50	10	221142.6	259.92s
Data15	20 000	50	10	227078.5	252.70s
Data16	20 000	50	10	223813.7	253.79s
Data17	20 000	50	10	227050.2	249.230s
Data18	20 000	50	10	256901.0	239.13s
Data19	20 000	50	10	206865.8	218.44s
Data20	20 000	50	10	215691.0	257.43s

Figure 3. Experiment 1.3 with size n=20,000

Experiment 2. Genetic Algorithm with Local Search

In the second phase of the experiments, the solutions obtained from the genetic algorithm heuristic were further refined using a local search. The same instances and sample sizes were used to ensure consistency and comparability of results.

Instance	Size of "n"	Size of "n"	Size of "m"	Genetic Algoritm Heuristic	Time S	Local Search	Time S	% Improve
Data1	1000	10	3	7903.4	35.34s	7432.9	0.06s	5.95%
Data2	1000	10	3	7601.9	34.84s	7217.7	0.47s	5.05%
Data3	1000	10	3	7207.1	35.34s	7169.5	0.062s	0.52%
Data4	1000	10	3	8072.1	33.40s	7743.5	0.18s	4.07%
Data5	1000	10	3	7605.5	34.50s	7328.6	0.26s	3.64%
Data6	1000	10	3	8166.1	33.82s	8001.7	0.20s	2.01%
Data7	1000	10	3	7451.7	37.34s	7300.2	0.51s	2.03%
Data8	1000	10	3	7812.6	34.49s	7675.6	0.26s	1.75%
Data9	1000	10	3	7351.0	33.07s	7217.7	0.14s	1.81%
Data10	1000	10	3	7236.9	34.01s	7080.3	0.25s	2.16%
Data11	1000	10	3	7672.1	33.19s	7487.2	0.16s	2.41%
Data12	1000	10	3	7208.3	33.66s	6972.9	0.23s	3.27%
Data13	1000	10	3	8779.8	33.63s	8563.8	0.19s	2.46%
Data14	1000	10	3	7588.8	34.00s	7270.6	0.24s	4.19%
Data15	1000	10	3	7662.5	34.07s	7440.9	0.28s	2.89%
Data16	1000	10	3	7398.3	33.83s	7292.6	0.35s	1.43%
Data17	1000	10	3	8543.4	33.24s	8076.9	0.16s	5.46%
Data18	1000	10	3	7484.4	33.69s	7305.4	0.15s	2.39%
Data19	1000	10	3	7800.2	33.75s	7643.7	0.27s	2.01%
Data20	1000	10	3	8039.3	31.17s	7423.2	0.39s	7.66%
							Avarage	**3.16%**

Figure 4. Experiment 2.1 with size n=1,000

Instance	Size of "n"	Size of "n"	Size of "m"	Genetic Algoritm Heuristic	Time S	Local Search	Time S	% Improve
Data1	10 000	25	5	116121.4	161.20s	113840.7969	1.71s	1.96%
Data2	10 000	25	5	121180.7	163.64s	114845.1753	0.74s	5.23%
Data3	10 000	25	5	104109.8	173.08s	103996.1201	0.62s	0.11%
Data4	10 000	25	5	97387.5	158.22s	97046.39614	0.51s	0.35%
Data5	10 000	25	5	100827.7	167.77s	97508.52762	1.62s	3.29%
Data6	10 000	25	5	112131.8	176.98s	107481.1094	1.14s	4.15%
Data7	10 000	25	5	102062.3	165.75s	95290.90191	0.60s	6.63%
Data8	10 000	25	5	112476.4	175.30s	111048.1283	0.7s	1.27%
Data9	10 000	25	5	109332.4	158.51s	107316.319	1.11s	1.84%
Data10	10 000	25	5	104273.0	182.33s	102767.2235	1.31s	1.44%
Data11	10 000	25	5	102693.9	176.25s	96259.09259	0.98s	6.27%
Data12	10 000	25	5	125373.3	189.28s	125373.3227	0.05s	0.00%
Data13	10 000	25	5	107250.2	153.39s	104788.4007	4.60S	2.30%
Data14	10 000	25	5	118078.3	183.43s	116671.5507	0.40s	1.19%
Data15	10 000	25	5	126643.8	186.03s	126643.7816	0.40s	0.00%
Data16	10 000	25	5	130483.2	170.55s	129491.1095	0.83s	0.76%
Data17	10 000	25	5	106921.5	169.89s	104384.1236	0.44s	2.37%
Data18	10 000	25	5	111971.0	30.573s	111697.0195	33.35s	0.24%
Data19	10 000	25	5	104941.1	147.60s	102998.6932	2.76s	1.85%
Data20	10 000	25	5	104947.9	190.87s	104090.0588	0.27s	0.82%
							Avarage	**2.10%**

Figure 5. Experiment 2.2 with size n=10,000

Instance	Size of "n"	Size of "n"	Size of "m"	Genetic Algoritm Heuristic	Time S	Local Search	Time S	% Improve
Data1	20 000	50	10	230504.4	279.96s	228369.0	0.749s	0.93%
Data2	20 000	50	10	231840.9	231.16s	212154.7	0.82s	8.49%
Data3	20 000	50	10	228598.7	238.02s	212984.9	0.65s	6.83%
Data4	20 000	50	10	248845.8	254.57s	240957.9	1.33s	3.17%
Data5	20 000	50	10	219550.5	235.21s	211147.2	0.85s	3.83%
Data6	20 000	50	10	223935.4	260.71s	222942.9	0.98s	0.44%
Data7	20 000	50	10	199164.8	283.47s	198282.6	1.03s	0.44%
Data8	20 000	50	10	217348.5	245.76s	207469.6	0.51s	4.55%
Data9	20 000	50	10	250451.9	229.12s	241901.1	4.95s	3.41%
Data10	20 000	50	10	212493.1	250.24s	200871.6	1.78s	5.47%
Data11	20 000	50	10	203685.4	246.56s	200207.4	1.27s	1.71%
Data12	20 000	50	10	205393.7	241.94s	194296.1	0.68s	5.40%
Data13	20 000	50	10	200296.8	251.51s	196449.3	1.79s	1.92%
Data14	20 000	50	10	221142.6	259.92s	215806.9	1.36s	2.41%
Data15	20 000	50	10	227078.5	252.70s	226958.0	0.55s	0.05%
Data16	20 000	50	10	223813.7	253.79s	214820.1	1.190s	4.02%
Data17	20 000	50	10	227050.2	249.230s	225961.5	0.764s	0.48%
Data18	20 000	50	10	256901.0	239.13s	239232.5	1.34s	6.88%
Data19	20 000	50	10	206865.8	218.44s	188734.2	1.47s	8.76%
Data20	20 000	50	10	215691.0	257.43s	208860.4	0.86s	3.17%
							Avarage	**3.62%**

Figure 6. Experiment 2.3 with size n=20,000

Analysis

The results indicate that the local search significantly improved the solution quality across all sample sizes. The average percentage improvement for the smallest sample size (1,000) was 3.16%, for the medium sample size (10,000) it was 2.10%, and for the largest sample size (20,000) it was 3.62%.

The local search consistently provided better solutions in a relatively short amount of time, making it a valuable addition to the genetic algorithm heuristic. The performance improvement, however, varied across different instances, suggesting that the effectiveness of the local search may depend on specific problem characteristics.

The computational experiments demonstrated the effectiveness of the genetic algorithm heuristic in solving the p-median problem for various sample sizes. The addition of a local search further enhanced the solution quality, achieving significant improvements across all tested instances. These findings underscore the potential of combining genetic algorithms with local search techniques to solve large-scale optimization problems efficiently.

Conclusions

The p-Median problem is a pivotal optimization challenge with significant applications across various domains such as urban planning, logistics, and healthcare. By determining the optimal locations for a fixed number of facilities, this problem enables efficient service delivery to dispersed populations, thereby enhancing accessibility and operational efficiency.

The computational experiments demonstrated the effectiveness of the genetic algorithm heuristic in solving the p-median problem for various sample sizes. The addition of a local search further enhanced the solution quality, achieving significant improvements across all tested instances. These findings underscore the potential of combining genetic algorithms with local search techniques to solve large-scale optimization problems efficiently.

Key takeaways from our exploration of the p-Median problem include:

1. **Versatility and Practicality**: The p-Median problem's versatility makes it applicable to diverse fields, from strategic placement of public amenities and emergency services in urban planning to optimizing distribution networks in logistics and ensuring equitable access to healthcare facilities.
2. **Mathematical Rigor and Computational Techniques**: The problem can be rigorously formulated as an integer linear programming (ILP) model, enabling precise mathematical solutions. However, due to the computational complexity of solving ILP models for large instances, heuristic and metaheuristic approaches, such as Genetic Algorithms and Local Search, play a crucial role in finding near-optimal solutions efficiently.
3. **Hybrid Heuristic Approaches**: Combining global search techniques like Genetic Algorithms with local optimization methods, such as Local Search, can significantly enhance the solution quality and convergence speed. This hybrid approach leverages the strengths of both methods, balancing exploration and exploitation to achieve superior outcomes.
4. **Impact on Decision-Making**: The insights derived from solving the p-Median problem inform critical decision-making processes in various sectors. For instance, in urban planning, it aids in alleviating congestion and improving accessibility, while in healthcare, it ensures better resource allocation and patient outcomes.

In conclusion, the p-Median problem serves as a valuable tool for optimizing resource allocation and service delivery in complex, real-world scenarios. Its ability to inform strategic decisions across multiple domains underscores its importance and the necessity for continued research and development of advanced computational techniques to tackle increasingly complex instances of the problem.

References

1. Hakimi, S. L. (1964). Optimum Distribution of Switching Centers in a Communication Network and Some Related Graph Theoretic Problems. Operations Research, 12(3), 450–459.
2. Church, R. L., & ReVelle, C. S. (1974). The Maximal Covering Location Problem. Papers of the Regional Science Association, 32(1), 101–118.
3. Bixby, R. E., & Boyd, E. A. (1998). Integer Programming and Combinatorial Optimization. Springer-Verlag Berlin Germany.
4. Bard, J. F. (2003). The flow shop scheduling polyhedron with setup times. Journal of Combinatorial Optimization, 7(3), 291-318.
5. Puyol, C. (2002). Natural gas pipeline optimization. In P. M. Pardalos & M. G. C. Resende (Eds.), Handbook of Applied Optimization (pp. 813-825).
6. Wu, S., Scott, L. R., & Boyd, E. A. (2002). A reduction technique for natural gas transmission network optimization problems. Annals of Operations Research, 117(1-4), 217-234.
7. Dempe, S., & Puyol, C. (2002). Discrete bilevel programming: Application to a gas shipper's problem. Working Paper 2002-02, Freiberg University of Mining and Technology, Freiberg, Germany.