

**Complexity and Approximations  
FINAL ASSIGNMENT**

Submitted to   
**Pascal Rebreyend**

Submitted by  
**Murshid Saqlain**([msq@du.se](mailto:msq@du.se))

June, 2019

**TABLE OF CONTENTS**

[1. INTRODUCTION 1](#_Toc12450125)

[2. DATA 2](#_Toc12450126)

[3. METHODOLOGY 3](#_Toc12450127)

[3.1 Model 3](#_Toc12450128)

[3.2 Simulated Annealing 4](#_Toc12450129)

[3.3 Greedy Algorithm 5](#_Toc12450130)

[4. RESULTS 6](#_Toc12450131)

[4.1 Simulated Annealing 6](#_Toc12450132)

[4.2 Greedy 7](#_Toc12450133)

[5. DISCUSSION 8](#_Toc12450134)

[6. REFERENCES 9](#_Toc12450135)

[7. APPENDIX 10](#_Toc12450136)

# 1. INTRODUCTION

For this assignment, we will address a type of problem called the *p*-median problem which was introduced by Hakimi [1] when he considered the optimal location of switching centers in a communication network. The objective function of this problem is to find the locations of *p* facilities to minimize the demand weighted total distance (total cost) between each node and the nearest facility. For the p-median problem the cost of serving demand nodes *i* is the product of the demand at node *i* and the distance between demand node *i* and the nearest facility to node *i*.

Particularly, we are required to find 19 optimal locations for Ikea stores in Sweden by using two different optimization algorithms. The optimal locations are chosen from 1938 candidate locations in such a way that the total cost of serving all customers from these locations are minimized, as defined by the *p*-median problem above.

First, we must consider how many total possible solutions exist for our problem. Given that there are 1938 total possible nodes or candidate nodes and that we have to find 19 optimal locations, the total number of possible solutions is

To get a perspective of the time needed to perform such an exhaustive calculation, let us consider an AMD Ryzen 7 1800X processor which was released in 2017. It can perform approximately 300,000 million instructions per minute (MIPS) [2]. The time taken for this processor would thus be:

years

This is approximately times the age of our known Universe ( years) [3]. Thus, an exhaustive search is by no means reasonable and we must consider other forms of solution, particularly heuristic and meta-heuristic methods (or algorithms). These methods have been tested in practice and have been shown to produce good solutions in reasonable amount of time. While these solutions may not be the absolute optimum, the simplicity in implementing these algorithms and the feasible computation times make them extremely appealing and useful.

For this assignment, I have chosen to use the two algorithms called Simulated Annealing algorithm and Greedy algorithm to solve our *p*-median problem with the Ikea store locations in Sweden.

The objectives are:

* Find the 19 optimal Ikea locations in Sweden using a Simulated Annealing algorithm.
* Find the 19 optimal Ikea locations in Sweden using a Greedy algorithm.
* Compare the cost of the above algorithms to the cost of the true Ikea locations.
* Compare the computation time of above algorithms.

# 2. DATA

The following datasets are used for this assignment:

**Matrix of Distance** (labelled PHD\_Matrix.txt in the codes): This is a 1938 x 187679 matrix with the 1938 rows representing each candidate node in Sweden and the 187,679 columns containing the distance of each demand point from the candidate node.

**List of Settlements** (labelled PHD\_Settlements.txt in the codes): This file contains 1938 rows that represent each of the candidate nodes, and three columns consisting of the X-coordinate, Y-coordinate, and ID of the node. This is used to extract the coordinates of the chosen nodes.

**List of population points** (labelledPHD\_Pop.txt in the codes): This file contains 187,679 rows that represent each of the demand points (settlements or approximate location of the customers), and three columns consisting of the X-coordinate, Y-coordinate, and the number of persons in each demand point. The number of persons in each demand point is used as the weight for each demand point.

**Locations of the 19 Ikea stores** (labelled Ikea.txt): This contains the X and Y coordinates of the actual Ikea stores in Sweden currently, the distance to the closest settlement, and ID of the settlements. These locations are used to calculate the benchmark cost with which we will compare the costs of our algorithms.

# 3. METHODOLOGY

## 3.1 Model

The p-median problem has a well known mathematical model and will be presented in this section. The notations used in the model are shown in Table 1.

|  |  |
| --- | --- |
| **Table 1** Explanation of notations used in the model | |
| **Notation** | **Explanation** |
| *I* = {1, …,D} | Set of demand points or customer location |
| *J* = {1,…,N} | Set of candidate nodes or locations for stores |
| *p* | Number of nodes/locations we want to select for stores |
|  | Weight/population of the demand of the customer *i* |
|  | Distance between customer *i* and store *j* |
|  | Decision variable, indicating the store *j* is selected or not (1 or 0) |
|  | Decision variable, indicating the customer *i* is served by store *j* or not (1 or 0) |

The mathematical formulation of the p-median problem can be specified as follows [1,4,5]:

(1)

subject to

, (2)

(3)

(4)

, (5)

The objective (1) is to minimize the total distance from customers or clients to their nearest facility or store. Constraint (2) shows that the demand of each customer or client must be met. From constraint (3), the number of stores to be located is *p*. Constraint (4) shows that customers must be supplied from an open facility, and constraint (5) presents the problem as a binary integer programming. The above formulation assumes that the potential facility sites are nodes on the network. Garey and Johnson (1979) showed that the p-median problem is *NP*-hard for variable values of *p* [6].

Two algorithms are used to find reasonable solutions to the above problem: Simulated Annealing and Greedy. For both algorithms, as a first step, only a small portion of the data is used as a test for the algorithm to check whether it works and whether we are getting reasonable solutions. Once the algorithm has been tested to work, they are applied on the entire dataset.

The algorithms used in this report will be compared to the reference cost, which is calculated by using the actual 19 Ikea store locations in Sweden. The reference cost is the value of the objective function (1) and is found to be 262664839929.

## 3.2 Simulated Annealing

Simulated annealing (SA) is a probabilistic technique for approximating the global optimum of a given function. It is a metaheuristic to approximate globassl optimization in a large search space for an optimization problem. The terminology of this algorithm stems from the technique of heating and controlled cooling of metal, also known as annealing in metallurgy.

At each step, the SA heuristic considers some neighbouring states of the current state *s*, and probabilistically decides whether to move to the new state *sn* or stay in the current state *s*. The system thus gradually moves to states of lower energy. The algorithm is iterated until a good enough state has been reached by the system or the computation budget has been exhausted.

That is, the algorithm move slowly from a random walk to a “gradient” descent by accepting only better solutions. The algorithm accepts a new state with the probability:

(6)

Here, is the change in system going from the current state to the new state, or the change in cost. T represents the temperature, which we gradually decrease with each step of the iteration. So, as the algorithm progresses with increasing number of iteration, the temperature goes down and it is ensured that the a bad solution is less likely to be accepted.

The temperature T is cooled with a cooling scheme using the following equation:

(7)

where *stepsize* determines how fast the temperature cools and *k* is the number of iteration the algorithm is at. I chose a *stepsize* of 0.01.

At higher temperatures, the probability is close to 1 and so the chance to choose worse solutions is higher. But with lower temperatures, the probability approaches 0, ensuring that worse solutions are less likely to be accepted. The process terminates after a certain number of iterations have been reached.

The initial solution is 19 randomly sampled locations from the 1938 candidate nodes. At each iteration, to find a neighbouring state, one of the locations is randomly replaced. The total number of iterations is set to 1000.

When the best solution state is found, the cost is calculated using (1). The algorithm is outlined below:

Let *f(s)* denote the value of the objective function (1) evaluation at a state *s*.

|  |  |
| --- | --- |
| **Algorithm 1** Simulated Annealing | |
| **Step 1** | Randomly choose *p* = 19 rows to form an initial solution from the 1938 candidate nodes. |
| **Step 2** | Set  Set  Set the number of iteration, *k*, to 1. |
| **Step 3** | Set T according to equation (7) |
| **Step 4** | Randomly replace one of the locations from to find using the following scheme:  *Step 3.1* Choose one of the 19 locations from randomly.  *Step 3.2* Use the row number as the mean of a normal distribution with standard deviation of 5, and pick a new row from this distribution with the constraint that the number is between 1 and 1938 and that this row is not already in . |
| **Step 5** | If < Set .  Otherwise, if , generate a uniformly distributed random number, . If , set ; else, keep unchanged. |
| **Step 6** | If < , Set and |
| **Step 7** | Increase the iteration number, *k*, by 1, and repeat Steps 3 to 6 until *k* =1000. |

SA algorithm is performed in R, and the code has been attached as an appendix.

## 3.3 Greedy Algorithm

For the Greedy algorithm, stores are added to our solution set one by one, in such a way so that the total cost for all customers is minimized, until *p* is reached. For this algorithm, the location that gives the minimum cost is always selected. The main drawback with the Greedy algorithm is that once a store is selected it stays in all subsequent solutions. Consequently, the final solution attained may be far from optimal. This algorithm is specifically known as Greedy-Add since stores are added one-by-one to attain the required number of stores.

The outline of the Greedy algorithm as indicated by Daskin [7] is presented below as follows:-

|  |  |
| --- | --- |
| **Algorithm 2** Greedy | |
| **Step 1** | Initialise *k* = 0 and = {}, the empty set. |
| **Step 2** | Increase *k*, the counter on the number of facilities located. |
| **Step 3** | Compute for each node *j*, which is not in the set , where is the demand at node *i*. Note that gives the value of the *p*-median of the objective function if we locate the *kth* facility at node *j*, given that the first *k-1* facilites are the locations given in the set . |
| **Step 4** | Find the node that minimizes . Add node to the set to obtain the set Note that gives the best location for the *kth* facility, given the location of the first *k-1* facilities. |
| **Step 5** | If *k* = *p*, stop. Go to step 2 if *k* < *p*. |

I implemented the Greedy algorithm in Python with the help of classmate, Ross May. The cost is calculated using the solution state with the 19 stores chosen by the Greedy algorithm.

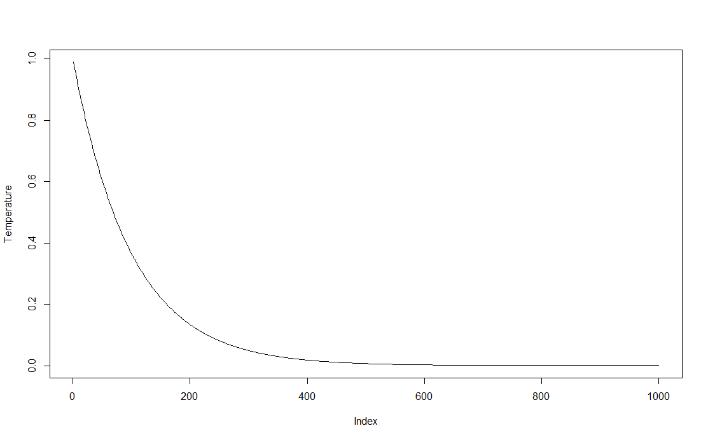
# 4. RESULTS

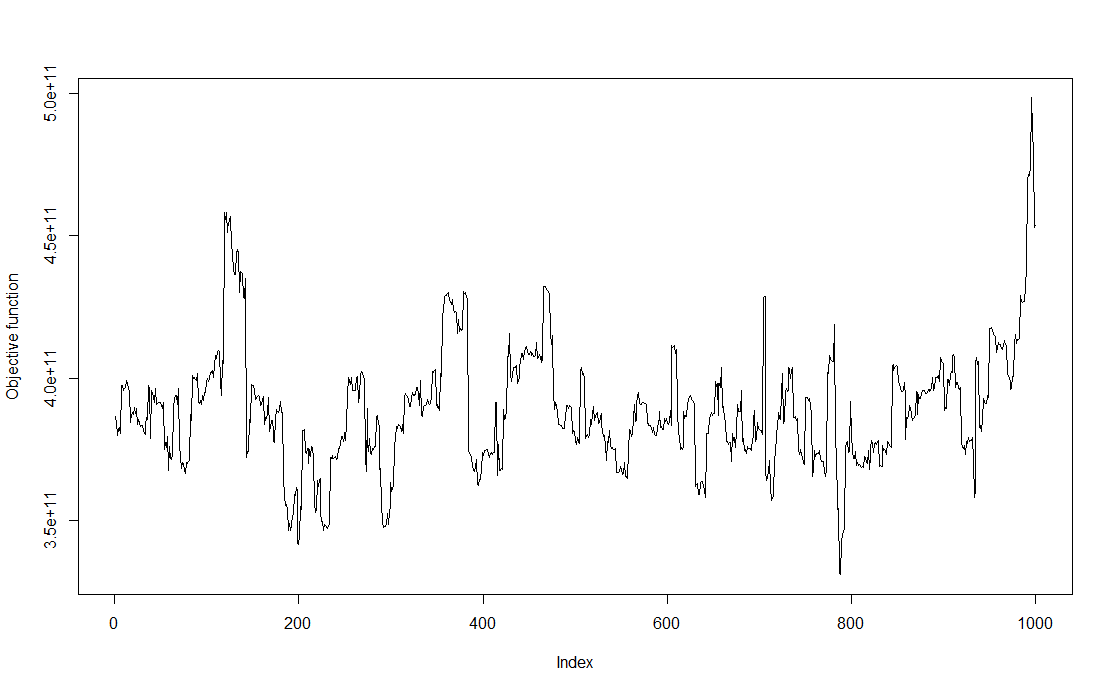
## 4.1 Simulated Annealing

The time taken for this algorithm was a few hours.

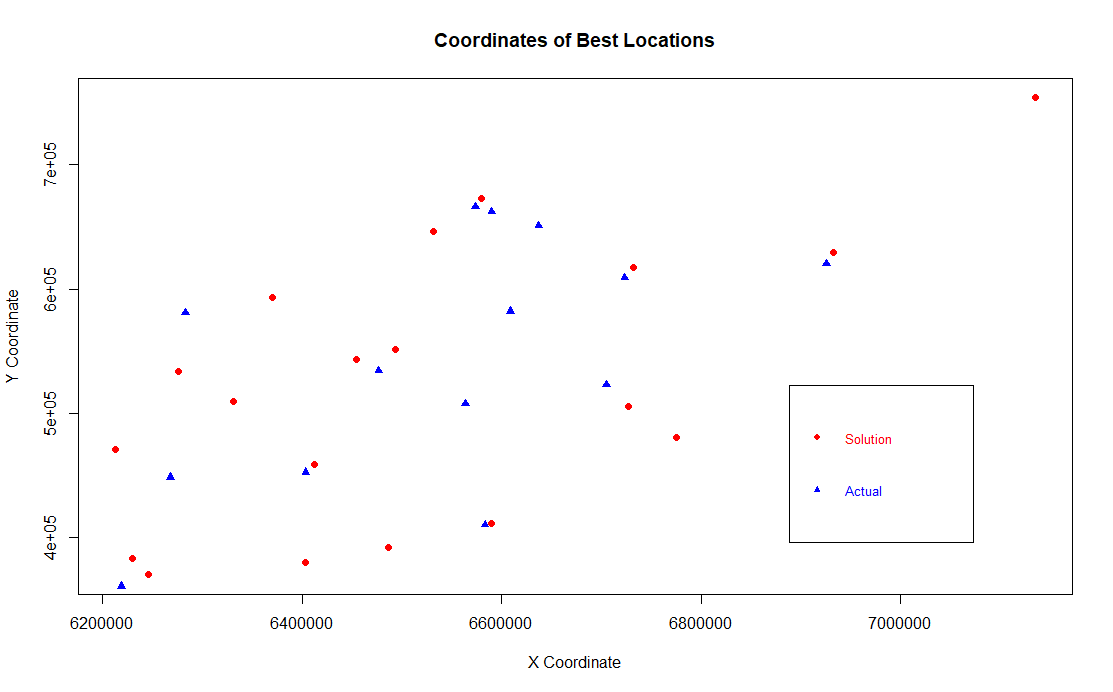
Cost of the Simulated Annealing algorithm was 453420259384. This is about 72.6% higher than the reference cost of 262664839929.

The temperature plot and the objective function over time plot are shown below:





The coordinates of the solutions are shown below:



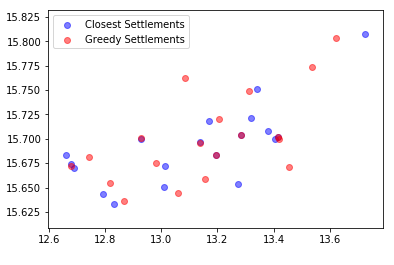
In the above figure, the blue triangles are actual locations, and the red circles are the 19 locations chosen by the Simulated Annealing algorithm.

## 4.2 Greedy

The time taken for this algorithm was about 20 minutes.

Cost of Greedy algorithm was 213152055193. This is about 20.8% lower than the reference cost of 262664839929.

A graph showing the actual store locations and the locations selected by the Greedy algorithm are shown below:



# 5. DISCUSSION

The purpose of this project was to tackled the *NP*-hard *p*-median problem for Ikea stores.

The Simulated Annealing algorithm took a few hours to run in R, and the results were far from optimal. The difference in cost was about 72% higher than the benchmark solution. But the goal here was not to get the best or most optimal solutions, but get some form of solution in a reasonable time frame and understand how the algorithm works and how it may be implemented.

I believe a few key choices could possibly improve the results of the SA algorithm that was not carried out for this report:

* Change the number of iterations to a higher value so that a wider search space is covered.
* Change the algorithm to randomly select 19 locations at every iteration instead of one at a time.
* Change the distribution from which the randomly replaced row is selected.
* Changing the *stepsize*, so the temperature does not approach 0 as fast. This also needs depends on the number of iterations we are performing. Ideally we want the temperature to approach 0 near the end of the iterations. If it approaches 0 too fast, then it will be very unlikely to accept “worse” solutions at higher iteration numbers, which defeats the purpose of SA algorithm.

The Greedy algorithm was much faster, taking about only 20 minutes. It was also 20.8% lower than the benchmark cost. There is not much we can do to change the results of this algorithm, since it is deterministic in nature.

I also decided to implement the algorithms using different languages. While it may not be directly compared since Python was used for Greedy and R was used for SA, Python seemed to save the computation time significantly more than R.

There are also other algorithms which have been shown to yield good results that have not been implemented for this report, such as genetic algorithms or hybrid algorithms.

This course taught me the importance of heuristic methods to tackle large-scale problems that may not be feasibly tractable. I implemented two algorithms to find heuristic solutions to a *NP*-hard *p*-median problem for Ikea stores. But these algorithms could be applied in other contexts as well, such as when choosing the location of a new Hosptial or transportation station.

# 6. REFERENCES

[1] Hakimi, S. Louis. "Optimum locations of switching centers and the absolute centers and medians of a graph." Operations research 12.3 (1964): 450-459.

[2] Chiappetta, Marco (2017-03-02). "AMD Ryzen 7 1800X, 1700X, And 1700 Review And Benchmarks: Zen Brings The Fight Back To Intel". HotHardware

[3] Chaboyer, Brian, et al. "A Lower Limit on the Age of the Universe." Science 271.5251 (1996): 957-961.

[4] Dzator, Michael, and Janet Dzator. "An Efficient Modified Greedy Algorithm for the P-Median Problem." (2015).

[5] Rebreyend, Pascal. Lecture slides, Complexity and Approximations course, Dalarna Univeristy. (2018)

[6] Johnson, David S., and Michael R. Garey. Computers and intractability: A guide to the theory of NP-completeness. WH Freeman, 1979.

[7] Daskin, Mark S. Network and discrete location: models, algorithms, and applications. John Wiley & Sons, 2011.

# 7. APPENDIX

The following files are attached to this report:

**Complexity\_Project.R** – R Script containing codes for the Simulated Annealing algorithm.

**Complexity\_Greedy.py**  – Python code for running the Greedy algorithm.