Combinatorial Optimization Report

Luis Marcelo Fonseca [luismarcelovaf@gmail.com](mailto:luismarcelovaf@gmail.com)

Adriel Garcia Hernandez [adriel.gh871217@gmail.com](mailto:adriel.gh871217@gmail.com)

Gabriel Pujol [gabrielpujol87@gmail.com](mailto:gabrielpujol87@gmail.com)

Daniel Gribel [dgribel@inf.puc-rio.com](mailto:dgribel@inf.puc-rio.com)

# Problem description

Consider objects defined by c binary characteristics and each of those objects belonging to g different groups. A group is defined by the probabilities of having characteristics with probabilities p, with 0.5 *p* 1.0 and characteristics with probability 1 - *p*. Consider also that , with the remaining number of characteristics up to *c* being noises with probability = 0.5 of occurring.

Given a *n* number of objects, generated following the group categorization rule given above, elaborate a classifier that can arrange each object to its respective group. This classifier must also consider that each group has a number of objects between and , which is generated randomly. Assume that the *label* of each object, that is, the group it belongs to, is known before hand with the purpose of testing the precision your algorithm. Elaborate tests with instances, which will be provided by this exercise, for sets of objects with the number of groups *g* previously known and also for sets of objects with the number of groups *g* unknown.

# Methods

The project consists on the implementation and testing of the following 4 methods:

1. **K-Means**
2. **UFL** – Uncapacitated Facility Location Problem
3. **P-Center** – Centroids Localization Problem
4. **K-Median** As an alternative clustering method

# Method descriptions

## **K-Means**

This method aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

The algorithm consists of an initialization and two main steps. We perform the initialization by randomly choosing k data points as the “means”, or centroids, and assigning to them the closest points of the dataset. Then, we execute the two main steps (update and assignment) until there are no changes in the assignment of clusters (our convergence criteria). The Update step consists of calculating the new means to be the centroids of the observation points in the new clusters. The Assignment step, assigns each observation to the cluster whose centroid has the minimum Euclidean distance to it: intuitively the "nearest" centroid.

## **K-Median**

This method is a variation of k-means clustering where instead of calculating the mean for each cluster to determine its centroid, one instead takes the median . The median is the element that is in the middle of all the elements in a cluster. This has the effect of minimizing error over all clusters with respect to the 1-norm distance metric, as opposed to the square of the 2-norm distance metric (which k-means does.)

## ULP – Uncapacitated Facility Location Problem.

This problem is about selecting a subset of the data points known as “*facilities*”, such as this selection minimizes the *distances* to another set of data points known as “*clients*”, and taking into account that each facility has a *cost* for being selected. The problem assumes that each facility can produce and ship unlimited quantities of the commodity under consideration, and that we have no capacity restrictions while transporting and storing the commodity in the clients.

In our case study, we consider the facilities and the clients as the same set of data points. Since we do not take into account the cost of opening a facility, there is a term missing from the original formulation’s objective function, and since that term was used to establish the amount of facilities that would be open, our problem needs to receive the amount of groups g as part of the input. We had to add a restriction to force the amount of facilities to be equal to the amount of groups we need.

## P-Center

The *p*-center problem is a NP-Hard facility location problem that requires locating *p* facilities on a given network and assigning clients (nodes) to these facilities so that the maximum of the distances between clients and the facilities they are assigned to is minimized.

# Method Formulations

## K-Means & K-Median

The algorithms takes a dataset of points as input, together with a parameter specifying how many clusters to create. The output is a set of cluster centroids and a labeling of that assigns each of the points in to a unique cluster. All points within a cluster are closer in distance to their centroid than they are to any other centroid.

The mathematical condition for the clusters and the K centroids can be expressed as:

Finding the solution is unfortunately [*NP-hard*](http://en.wikipedia.org/wiki/NP-hard). Nevertheless, an iterative method known as Lloyd’s algorithm exists that converges (albeit to a local minimum) in few steps. The procedure alternates between two operations. (1) Once a set of centroids is available, the clusters are updated to contain the points closest in distance to each centroid. Given a set of clusters, the centroids are recalculated.

The update of the centroids for each algorithm is shown below:

### K-Means

### K-Median

is the element that is in the middle of all the elements in .

The two-step procedure continues until the assignments of clusters and centroids no longer change. For the starting set of centroids, several methods can be employed, in our work we use random assignation.

## P-Center

The *p*-center problem is an NP-Hard facility location problem that requires locating *p* facilities on a given network and assigning clients (nodes) to these facilities so that the maximum of the distances between clients and the facilities they are assigned to is minimized.

To formulate an integer linear program (ILP) to solve this problem, we define a decision variable:

That describes which nodes are serviced by which facility location. We need to define another decision variable.

That describes the locations at which a facility is placed. We must define an additional decision variable that represents the maximum distance between any demand node and its servicing facility. Given these decision variables, we can now formulate the vertex p-center problem as the following MILP.

This modeling since has two integer variables takes a long time to converge to the solution. Therefore, to solve the problem efficiently, it is necessary to solve some relaxations of it. The idea of our solution is determine whether possible to build a solution where the greatest distance from one node to a facility is less than or equal to a distance . A modeling of the relaxation of this problem is as follows:

Where is defined as:

Finally, the idea is to explore the possible distances and selecting the smallest distance that makes fits the previous model. To optimize the amount of iterations, the distances are ordered and have a binary search performed on them. In our case study, we consider the facilities and the clients as the same set of data points.

## UFL

The usual input for this problem will consist of:

* A set *D* of clients
* A set *F* of potential facility locations
* A distance function
* A cost function

The output of the problem is the set:

The optimization model used was:

Subject to:

Where the variable represents if the facility will be open, the variable, the connections between each client and its facility, and the constant is the given amount of groups.

The restriction (1) indicates that each client is connected to one facility. The restriction (2) indicates that the connections client-facility depends on the facility being open. The restriction (3) indicates that the total amount of facilities must be the given amount of groups. Finally, the restrictions (4) and (5) define the variables and as binary.

The relaxation of the integrality constraint (4) of the former optimization model is:

Subject to:

With this relaxation, we preserve the variable as binary, but we change the type of to continuous.

By observation of the results, we found that all values of the variable were either zero or one. We believe that these results are due to the fact the objective function minimizes the sum of the edges, therefore, the algorithm never tries to allocate fractional edges to furthest facilities, instead, it gives the whole edge to the closest facility.

# Instances Descriptions

We have generated three dataset types with different parameters. We have labeled each dataset name with their type followed by the number *n* of data points existing in it.

## Alpha

This dataset contains the following parameters:

*c* = 25

= 5

= 3

= 0.85

= =

## Beta

This dataset contains the following parameters:

*c* = 25

= 3

= 6

= 0.85

= =

## Gamma

Gamma is a dataset with values . This means each group has a random amount of elements between the values of and . The parameters described below are the same for all Gamma datasets:

*c* = 25

= 5

= 3

= 0.85

= \* 0.8

= \* 1.2

Furthermore, we have the unique parameters for each dataset, with being the number of elements in the group *i*.

### Gamma-203

= 163

= 244

= 60

= 58

= 85

### Gamma-503

= 403

= 604

= 197

= 153

= 153

### Gamma-1014

= 811

= 1216

= 364

= 346

= 304

# Method Results

The results for each implemented method is shown in the tables below for the number *g* of groups previously known. The *time* column defines the amount of time in seconds the algorithm took to compute the results, and the *accuracy* column defines the metric we have used to evaluate the precision of the algorithm. This metric is calculated as where is the number of data points classified correctly and *n* is the total number of points in the dataset.

## K-Means

This algorithm also includes a number of iterations before achieving convergence, where the algorithm stops the approximation of the centroids.

|  |  |  |  |
| --- | --- | --- | --- |
| **Instance** | **Time (sec)** | **Accuracy** | **Iterations** |
| alpha-204 | 0.0246 | 0.9951 | 4 |
| alpha-504 | 0.0722 | 0.9901 | 7 |
| alpha-1014 | 0.1001 | 0.9970 | 5 |
| beta-204 | 0.0855 | 0.8971 | 12 |
| beta-504 | 0.1918 | 0.8988 | 11 |
| beta-1014 | 0.3909 | 0.8915 | 11 |
| gamma-203 | 0.0273 | 0.9852 | 7 |
| gamma-503 | 0.0643 | 0.9960 | 6 |
| gamma-1014 | 0.3039 | 0.9970 | 15 |

**Table 1 - K-Means Results**

## K-Median

|  |  |  |
| --- | --- | --- |
| **Instance** | **Time (sec)** | **Accuracy** |
| alpha-204 | 2.6600 | 0.9706 |
| alpha-504 | 27.2260 | 0.9762 |
| alpha-1014 | 81.0410 | 0.9625 |
| beta-204 | 2.5681 | 0.7402 |
| beta-504 | 45.9349 | 0.8016 |
| beta-1014 | 123.8577 | 0.7909 |
| gamma-203 | 2.5250 | 0.9754 |
| gamma-503 | 32.3567 | 0.9722 |
| gamma-1014 | 145.5027 | 0.9398 |

**Table 2 - K-Median Results**

## UFL

|  |  |  |
| --- | --- | --- |
| **Instance** | **Time (sec)** | **Accuracy** |
| alpha-204 | 2.6849 | 0.9706 |
| alpha-504 | 26.1469 | 0.9722 |
| alpha-1014 | 79.8623 | 0.9625 |
| beta-204 | 2.5644 | 0.7304 |
| beta-504 | 29.4747 | 0.7996 |
| beta-1014 | 97.8371 | 0.7959 |
| gamma-203 | 2.5255 | 0.9754 |
| gamma-503 | 26.2553 | 0.9682 |
| gamma-1014 | 116.9546 | 0.9359 |

**Table 3 - UFL Results**

## P-Center

|  |  |  |
| --- | --- | --- |
| Instance | Time (sec) | Accuracy |
| alpha-204 | 28.4797 | 0.8137 |
| alpha-504 | 3293.1292 | 0.9226 |
| alpha-1014 | - | - |
| beta-204 | 26.9533 | 0.1863 |
| beta-504 | 1852.0402 | 0.5060 |
| beta-1014 | - | - |
| gamma-203 | 9.6121 | 0.8276 |
| gamma-503 | 568.3972 | 0.8410 |
| gamma-1014 | - | - |

**Table 4 - P-Center results**

# Analysis of Results

After testing the instances described above we can draw the following conclusions: the algorithm p-center turned out to be the worst that showed runtime and accuracy, we can only test it on instances down to 502 elements, since instances with 1000 elements, according to the calculations, are estimated to take more than 5 hours to complete. The K-Median and UFL algorithms showed very similar results, being the last one slightly better in terms of speed and the first was better in terms of accuracy. Finally, K-Means algorithm was the most efficient, its execution time in all instances was far superior to the rest, and his accuracy was higher than the other three algorithms.

We have also built the following table with the comparison of each algorithm and their respective accuracies and run time:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instance | UFL | P-Median | P-Center | K-Means |
| alpha-204 | 0.9706 | 0.9706 | 0.8137 | 0.9951 |
| alpha-504 | 0.9722 | 0.9762 | 0.9226 | 0.9901 |
| alpha-1014 | 0.9625 | 0.9625 | - | 0.9970 |
| beta-204 | 0.7304 | 0.7402 | 0.1863 | 0.8971 |
| beta-504 | 0.7996 | 0.8016 | 0.5060 | 0.8988 |
| beta-1014 | 0.7959 | 0.7909 | - | 0.8915 |
| gamma-203 | 0.9754 | 0.9754 | 0.8276 | 0.9852 |
| gamma-503 | 0.9682 | 0.9722 | 0.8410 | 0.9960 |
| gamma-1014 | 0.9359 | 0.9398 | - | 0.9970 |
| AVERAGE | 0.9012 | 0.9033 | 0.6829 | **0.9609** |

**Table 5 - Algorithm accuracy per instance**

**Figure 1- Algorithm accuracy per instance**

**Figure 2 - Algorithm average accuracy**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Instance** | **UFL** | **P-Median** | **P-Center** | **K-Means** |
| alpha-204 | 2.6849 | 2.6600 | 28.4797 | 0.0246 |
| alpha-504 | 26.1469 | 27.2260 | 3293.1292 | 0.0722 |
| alpha-1014 | 79.8623 | 81.0410 | - | 0.1001 |
| beta-204 | 2.5644 | 2.5681 | 26.9533 | 0.0855 |
| beta-504 | 29.4747 | 45.9349 | 1852.0402 | 0.1918 |
| beta-1014 | 97.8371 | 123.8577 | - | 0.3909 |
| gamma-203 | 2.5255 | 2.5250 | 9.6121 | 0.0273 |
| gamma-503 | 26.2553 | 32.3567 | 568.3972 | 0.0643 |
| gamma-1014 | 116.9546 | 145.5027 | - | 0.3039 |
| AVERAGE | 42.7006 | 51.5191 | 963.1020 | **0.1401** |

**Table 6 - Algorithm run times per instance**

# Conclusion

In this work, some experiments regarding clustering tasks were performed, in order to test different objective functions.

Datasets were generated with known labels, which allows the produced methods to be used for classification purposes, which is the main goal of this work. The approach considers the existence of different groups and the generation of objects (instances) according to some probability regarding each one of the characteristics (attributes). Then, the proposed solution try to assign each object to one group. We consider two scenarios for each dataset: when the number of groups (classes) is known and when it is unknown.

In order to test the proposed approach, four different methods were considered: K-means, P-center, P-Median and Uncapacitated Facility Location. According to results obtained, K-means overcame the other algorithms in terms of both accuracy and processing time.

K-median and Uncapacitated Facility Location were both competitive in terms of accuracy and processing time. P-center performed worse in terms of accuracy and much slower for the largest instances. Thus, K-means was chosen as the best algorithm for this problem.

With exception of K-means, results were achieved in an exact way according to each objective function, by running experiments with Gurobi MIP solver and Python programming language. K-means was implemented in Python according to its definition.

# References

[1] Dantrakul. S, Likasiri. Ch. 2013. Applied *p*-median and *p*-center algorithms for facility location problems.

[2] Drezner. Z. 1984. The p-Centre Problem-Heuristic and Optimal Algorithms

[3] Hartigan. J. A, Wong M. A. 1979. A K-Means Clustering Algorithm.

[4] Jain, A. K. 2009. Data clustering: 50 years beyond K-means.

[5] <https://datasciencelab.wordpress.com/2013/12/12/clustering-with-k-means-in-python/>

[6] [www.cs.cmu.edu/~anupamg/adv-approx/lecture4.pdf](http://www.cs.cmu.edu/~anupamg/adv-approx/lecture4.pdf)

[7] <http://www.or.uni-bonn.de/~vygen/files/fl.pdf>

# Appendix

GitHub repository in:

<https://github.com/danielgribel/oc-clustering>