UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

OTDM - Lab Assignment

Integer Optimization: The cluster-median problem

André Nogueira Sousa (andre.nogueira)
José Miguel Hernández Cabrera (jose.miguel.hernandez)

January 6, 2022

1 Introduction

This work addresses the cluster-median problem, where we used three data matrices, of $A = (i, j)$, $i = 1, \ldots, m$, $j = 1, \ldots, n$ of m points and n variables. We group them in k clusters such that the points in each clusters are similar; the definition of the k cluster over each dataset is explained in Section 2. The used median for a subset of points $\mathcal{I} \subseteq \{1, \ldots, m\}$, defined as the nearest point to all points of $\mathcal{I}$:

$$r \text{ is the median of } \mathcal{I} \text{ if } \sum_{i \in \mathcal{I}} d_{ir} = \min_{j \in \mathcal{I}} \sum_{i \in \mathcal{I}} d_{ij} \tag{1}$$

Where $D = (d_{ij})$, $i = 1, \ldots, m$, $j = 1, \ldots, m$ is the matrix of Euclidian distances.

2 Finding Matrices A

We tested three datasets, iris dataset, dermatology cases and random generated cluster datasets. Each one of them was parsed into `.dat` files for the AMPL program, exporting A matrices with their respective numeric datasets.

2.1 Iris dataset

We use the famous the Fisher's Iris dataset [Fis36]. We selected this dataset since it has three classes from the iris flower species: *setosa*, *virginica* and *versicolor*, with 50 instances each. Furthermore, it contains features: width and length of petals and sepal, respectively. In Figure 1 we show how each of the pairwise features are organized in three different clusters, with their own densities as well.

2.2 Dermatology

We used the Dermatology dataset [GDI98] published from the UCI Machine Learning Repository [DG17]. It contains 366 instances and 34 features. One of them is nominal and the rest are numeric. More importantly, it contains 6 classes that are described in Table 1.

Table 1: Dermatology classes

Code	Class	N. of instances
1	Psoriasis	112
2	Seboreic dermatitis	61
3	Lichen planus	72
4	Pityriasis rosea	49
5	Cronic dermatitis	52
6	Pityriasis rubra pilaris	20

2.3 Blobs

The final dataset is randomly generated isotropic Gaussian blobs containing 6 clusters with the help of the function `make_blobs` from the Python's library Scikit-Learn [Ped+11]. It contains 1 000 instances and 5 features with 6 clusters. The resulting description is done is Figure 2. Our main interest with this dataset was to test how the integer linear programming behaves with large m.

Figure 1: Iris dataset distribution

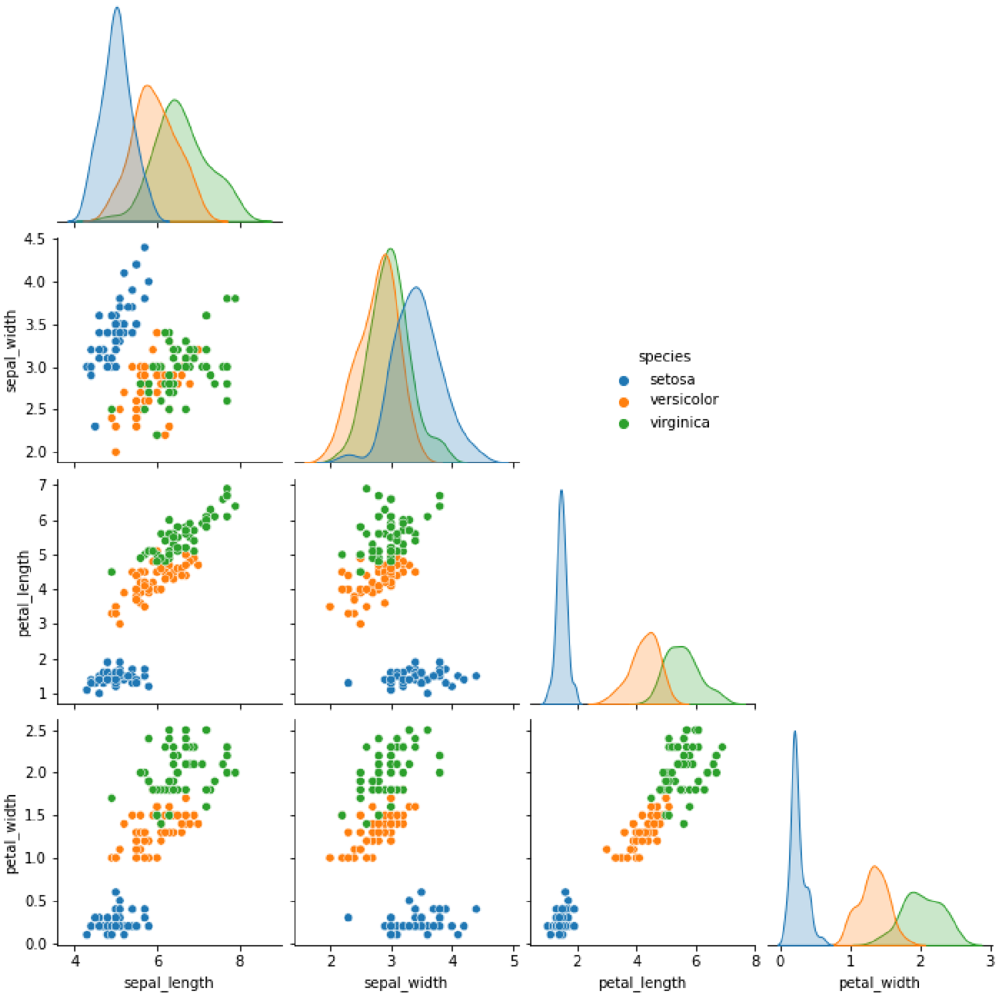
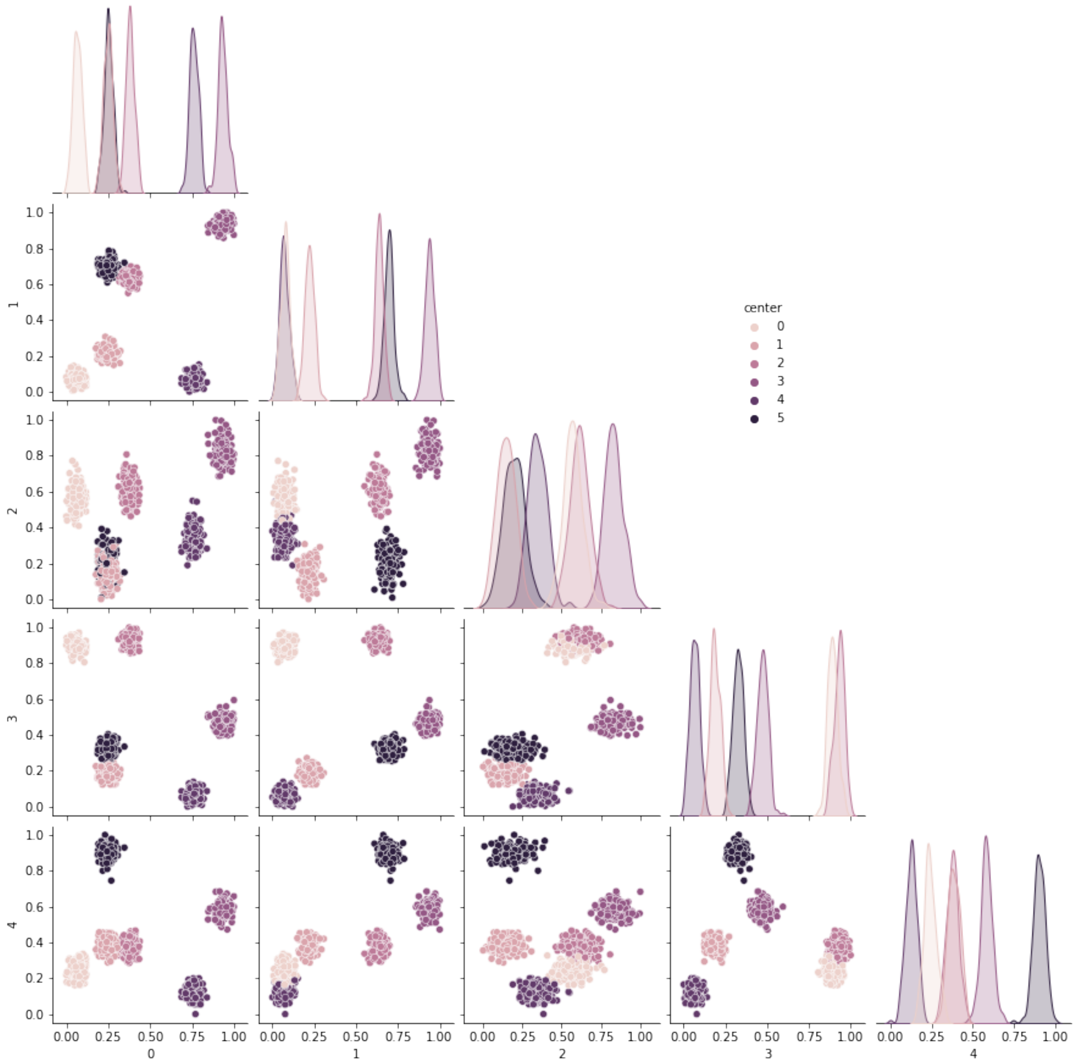


Figure 2: Blobs random datasets



3 Integer Linear Problem

To formulate the problem as an integer optimization problem, for simplicity we considered m clusters in the formulation, such that $m-k$ of them will be empty. We also denoted ”cluster-j” as the cluster whose median is the element j. Regarding of the variables of the formulation, for each $i, j = 1, \dots, m$,

$$x_{ij} = \begin{cases} 1 & \text{if element } i \text{ belongs to cluster-} j, \\ 0 & \text{otherwise.} \end{cases}$$

This way we minimize the overall distance between points and their cluster medians by making k and only k ”correct” clusters.

Considering that if cluster-j exists, then element j will be its median and it will belong to cluster-j; in other words, if cluster-j exists then $x_{jj} = 1$. Otherwise, the total distance would not be minimized. To force that k cluster form, we use the following formulation:

$$\begin{aligned} \text{min} & \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} x_{ij} \\ \text{subject to} & \sum_{j=1}^{m} x_{ij} = 1 \quad i = 1, \dots, m \\ & \sum_{j=1}^{m} x_{jj} = k \\ & x_{jj} \geq x_{ij} \quad i, j = 1, \dots, m \\ & x_{ij} \in \{0, 1\} \end{aligned}$$

The last group of constrains could be replaced by the following:

$$\begin{aligned} \text{min} & \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} x_{ij} \\ \text{subject to} & \sum_{j=1}^{m} x_{ij} = 1 \quad i = 1, \dots, m \\ & \sum_{j=1}^{m} x_{jj} = k \\ & m x_{jj} \geq \sum_{j=1}^{m} x_{ij} \quad j = 1, \dots, m \\ & x_{ij} \in \{0, 1\} \end{aligned}$$

We tested both configurations obtaining equal results, however, in a considerably higher time. We know that even though the second formulation presents less constraints, m instead of m^2, this is not a guarantee of better formulation. As an example for the Iris data set the solving time changed from about 2 seconds to over 46 seconds when we changed from the first to the second formulation. Therefore, we kept the first formulation.

To calculate K-means in CPLEX solver we used the following model:

```
int m = ...;   // Number of observations
int n = ...;   // Number of variables per observation
int k = ...;   // Number of Classes

// Range variables
range M = 1..m;
range N = 1..n;

// Input matrix A with all data used in the clustering
float A[M][N] = ...;
// Matrix with the euclidian distance between every two observations in A
float D[M][M];

// Decision boolean variable telling if observation 'i' is in cluster 'j'
dvar boolean X[M][M];

// Calculating the matrix with the euclidian distances D
execute {
  for (var i=1;i<=m;i++) {
    for (var j=1;j<=m;j++){
        var sum2 = 0;
      for (var l=1;l<=n;l++) {
          sum2 = sum2 + Math.pow((A[i][l]-A[j][l]), 2);
      }
      D[i][j] = Math.pow(sum2, 0.5);
}}}

// Objective function is to minimize the summation of the euclidian distances ...
// ... of each observation to the assigned cluster.
minimize
sum(i in M)
  (sum(j in M)
    (D[i][j]*X[i][j]));

subject to {
  // Constraint 1: Each observation must be assigned to only 1 cluster
    forall (i in M)
      sum(j in M) X[i][j] == 1;
    // Constraint 2: There must be k clusters
    sum(j in M) X[j][j] == k;
    // Constraint 3: An observation can belong to a cluster only if the cluster exists
    forall (i in M, j in M)
    X[j][j] >= X[i][j];
  // Same as Constraint 3 but slower (49:65 vs 2:20)
//  forall (j in M)
//    m*X[j][j] >= sum (i in M) X[i][j];
}

// Printing the results considering the element and its assigned cluster
execute {
  writeln("Element, Class");
    for (var i in M) {
      for (var j in M) {
          if (X[i][j] == 1) {
            writeln(i + "," + j);
}}}}
```

Listing 1: K-means ILP: k_means.mod

The CPLEX model is also presented in the attachments along with the *.dat* file.

4 Heuristics: MST

Complex problems, such as the k-means clustering presented in this project, are candidates for heuristic methods, once for large data sets its solution may take too long using integer linear optimization. Therefore, we applied an heuristic method using minimum spanning tree to find clusters.

To do so, we first create a graph where each node is an observation of the data set and edges are added connecting all pairs of nodes, with the Euclidean distances between the observations as the weight of the edges. The graph is them implemented as and adjacency matrix. After that the Prim algorithm is applied to find the minimum spanning tree of the graph. The final step is to remove the $k-1$ edges with largest weights from the minimum spanning tree, resulting in k disconnected subgraphs that are our k clusters.

The minimum spanning tree is a subgraph of the original graph which includes all the nodes of the graph with the smallest possible amount of edges and the minimum sum of edges weight. To find this subgraph we adopted the Prim algorithm. This algorithm follows the greedy approach, starting from a source node it adds edges with the lowest weight.

Algorithm 1 Prim Algorithm

```
$MST \longleftarrow \varnothing$
$M \longleftarrow$ First Node of graph
$N \longleftarrow$ All Nodes in graph
while $M \neq N$ do
    $Weight_{best} \longleftarrow \infty$
    for $n1\ in\ N - M$ do
        for $n2\ in\ M$ do
            if $Weight(n1, n2) \leq Weight_{best}$ then
                $Weight_{best} \longleftarrow Weight(n1, n2)$
                $Node_{included} \longleftarrow n1$
                $Edge_{included} \longleftarrow (n1, n2)$
            end if
        end for
    end for
    $MST \longleftarrow MST \cup \{Edge_{included}\}$
    $M \longleftarrow M \cup \{Node_{included}\}$
    return $MST$
end while
```

The presented algorithm returns the Minimum Spanning Tree (MST) that is used to obtain the k clusters as explained in the previous paragraph and presented in the following pseudo code. All algorithms were implemented in *Python* and are attached to this report.

Algorithm 2 k-means clustering Heuristics algorithm

```
$A \longleftarrow$ Input data $mxn$ matrix
$D \longleftarrow$ Empty $mxn$ matrix
$k \longleftarrow$ Number of clusters target
$i, j \longleftarrow 0$
for $row1inA$ do
    for $row2inA$ do
        $D[i][j] \longleftarrow euclidian\_distance(row1, row2)$
        $j \longleftarrow j + 1$
    end for
    $i \longleftarrow i + 1$
end for
$MST \longleftarrow Prim\_Algorithm(D)$
$MST_k \longleftarrow MST$
for $i\ in\ range(k - 1)$ do
    $MST_k \longleftarrow MST_k - \{(n1, n2) \mid weight(n1, n2) > weight(n3, n4)\ \forall\ n3, n4 \in MST_k\}$
end for
$Clusters \longleftarrow subgraphs(MST_k)$
return $Clusters$
```

5 Results

Results are presented for the ILP and for the Heuristic method for the three data sources. The performance of the two methods are compared in terms of:

- Objective function value: the *Python* code developed to do the Heuristic method was adapted to identify the center of each subgraph and sum the distances of each observation in each subgraph to the central node, in order to present the same metric used in the ILP.
- Time to reach a solution.
- Quality of the clustering: Once we have the class label of each observation in the three data sources we can check if the clusters are joining together observations of the same class.

Once we are using euclidean distances to measure the similarity between observations, we opt to standardize the variables before applying the clustering.

5.1 Iris

The Iris data set has 150 observation distributed in three different classes: *Iris-setosa*, *Iris-versicolor* and *Iris-virginica*. Therefore, we executed both clustering methods using $k = 3$. Table 2 presents the time and cost of the solutions for the Iris data set.

Table 2: Results for Iris data set

	Objective Function Cost	**Time (sec)**
ILP	29.76	2.31
Heuristics	41.96	0.52

From Table 2 we can notice that the objective function value of the Heuristic method is about 40% higher than the ILP, however the solution is obtained in $1/4$ of the time. In order to better understand what this difference in the objective function represents in terms of clustering, we verified if the clusters are joining together observations of the same class. Table 3 presents how the different classes are clustered resulting from the ILP and from the Heuristic method. We can notice that the ILP is able to separate the individuals in clusters with the three classes well distinguished, only 13 observations are not clustered exclusively with individuals of the same class. While the heuristics creates two main clusters, one composed by observations of the Setosa class and another composed by Versicolor and Virginica, the third contains only one observation.

Table 3: Results for Iris data set - Clustering

ILP - CPLEX			
Cluster	**Setosa**	**Versicolor**	**Virginica**
1	50	0	0
2	0	40	3
3	0	10	47
Heuristic			
Cluster	**Setosa**	**Versicolor**	**Virginica**
1	49	0	0
2	0	50	50
3	1	0	0

The graphs and subgraphs resulting from the heuristic process are presented in the attachments.

5.2 Dermatology

The Dermatology data set has 366 distributed in six different classes: *psoriasis*, *seboreic dermatitis*, *lichen planus*, *pityriasis rosea*, *cronic dermatitis*, and *pityriasis rubra pilaris*. Therefore, we executed both clustering methods using $k = 6$. This data set was chosen due to its increased complexity when compared with Iris, once it has 34 variables and 366 observations with 6 classes. We aimed to evaluate the behaviour of both methods in more complexes data structures. Table 4 presents the time and cost of the solutions for the Dermatology data set.

Table 4: Results for Dermatology data set

	Objective Function Cost	**Time (sec)**
ILP	435.19	43.58
Heuristics	675.79	3.27

From Table 4 we can notice that the objective function value of the Heuristic method is about 55% higher than the ILP, however the solution is obtained in 8% of the time. Again to better understand what this difference in the objective function represents in terms of clustering, we verified if the clusters are joining together observations of the same class. Table 5 presents how the different classes are clustered resulting from the ILP and from the Heuristic method. We can notice that the same behaviour from Iris data repeats, ILP is able to separate the individuals in clusters with the six classes well distinguished, only 22 observations are not clustered exclusively with individuals of the same class. While the heuristics creates two main clusters, one composed by observations of the *lichen planus* class and another composed by the combination of all other classes, the clusters 3, 4, 5 and 6 contains only one observation.

Table 5: Results for Dermatology data set - Clustering

ILP - CPLEX						
Cluster	**psoriasis**	**seboreic**	**lichen**	**rosea**	**cronic**	**rubra**
1	104	0	0	0	0	0
2	5	46	0	4	0	0
3	0	0	71	0	0	0
4	1	14	0	44	2	0
5	1	0	0	0	46	0
6	0	0	0	0	0	20
Heuristic						
Cluster	**psoriasis**	**seboreic**	**lichen**	**rosea**	**cronic**	**rubra**
1	109	60	0	48	48	20
2	0	0	69	0	0	0
3	0	0	1	0	0	0
4	1	0	0	0	0	0
5	0	0	1	0	0	0
6	1	0	0	0	0	0

The main conclusion to be extracted from these results is that ILP presents better results in therms of quality of the clustering, as expected once it is a deterministic method, however when we increase the complexity of the data the time performance of ILP worsens exponentially, so we expect that beyond a certain level of complexity, solving the problem with ILP will became unpractical, forcing us on rely in heuristic methods.

The graphs and subgraphs resulting from the heuristic process are presented in the attachments.

5.3 Blobs

As stated in Section 2, the dataset is configured by 6 clusters, hence we also set parameter k to 6 in both ILP and Heuristic methods. In contrast with the previous datasets, Iris and Dermatology, the blobs dataset contains 1 000 instances, whose impact over performance is depicted in Table 6, showing similar behaviour to the previous datasets: we obtain lower objective function cost in ILP, but the amount of seconds it takes to complete two order of magnitude higher than the MST method.

Table 6: Results for Blobs data set

	Objective Function Cost	**Time (sec)**
ILP	82.63	421.20
Heuristics	90.16	9.88

Regarding the clustering membership, as a difference between the previous datasets, both results have 100 % accuracy. This might be explained due to the fact that the blobs are already divided in clusters with small standard deviation. This results might differ greatly by different parameters, but this is a task beyond the scope of the project. Nonetheless, what we stated before regarding ILP being a better method for clustering quality still stands. However, Heuristics methods show that in well behave data perform almost as good as the ILP in said terms, and clearly outperforms in terms of time.

In this particular case, once both methods present the same clustering with 100 % accuracy, the Objective Function Cost from table 6 should be equal. The results differs due to the method adopted to calculate the cost of the objective function in the Heuristic, that differs from the ILP. In the Heuristic, we find the central node of each subgraph applying builtin centrality algorithms from *networkx* library, and then we sum the distances of each node in the subgraph to the central node. The central node found this way may differ from the central node resulting from the ILP, therefore, equal clusters would result in slightly different objective function costs.

Lastly, the resulting MST graphs are shown in the attachments.

Table 7: Results for isotropic Gaussian blobs - Clustering

ILP - CPLEX						
Cluster	**1**	**2**	**3**	**4**	**5**	**6**
1	167	0	0	0	0	0
2	0	167	0	0	0	0
3	0	0	167	0	0	0
4	0	0	0	167	0	0
5	0	0	0	0	166	0
6	0	0	0	0	0	166
Heuristics						
Cluster	**1**	**2**	**3**	**4**	**5**	**6**
1	167	0	0	0	0	0
2	0	167	0	0	0	0
3	0	0	167	0	0	0
4	0	0	0	167	0	0
5	0	0	0	0	166	0
6	0	0	0	0	0	166

References

[DG17] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. url: `http://archive.ics.uci.edu/ml`.

[Fis36] Ronald A Fisher. “The use of multiple measurements in taxonomic problems”. In: *Annals of eugenics* 7.2 (1936), pp. 179–188.

[GDI98] H Altay Güvenir, Gülşen Demiröz, and Nilsel Ilter. “Learning differential diagnosis of erythemato-squamous diseases using voting feature intervals”. In: *Artificial intelligence in medicine* 13.3 (1998), pp. 147–165.

[Ped+11] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

6 Attachments

6.1 Attached files description

- NB: Jupyter notebooks containing code for dataset exploration and generation, as well as results from the blobs datasets.
- Heuristic: Folder containing the *Python* code developed to perform the clustering using our implementation of the Prim algorithm to generate MST. The data sources used are included.
- ILP: Folder containing CPLEX model and data used to perform clustering with CPLEX.

6.2 Results Attachments - Iris

Figure 3 presents the Minimum Spanning tree obtained for the Iris data set.

Figure 4 presents the subgraphs obtained from the MST of the Iris data set, the third subgraph which is composed only by one node is not presented. The node marked in read is the calculated center of the cluster.

6.3 Results Attachments - Dermatology

Figure 5 presents the Minimum Spanning tree obtained for the Dermatology data set.

Figure 6 presents the subgraphs obtained from the MST of the Dermatology data set; the subgraphs 3, 4, 5 and 6 which are composed only by one node are not presented. The node marked in read is the calculated center of the cluster.

6.4 Results Attachments - Blobs

Figure 7 depicts the Prim's MST from the Isotropic Gaussian Blobs and Figure 8 shows the six subgraphs, where the red dot is the calculated center of the cluster.

Figure 3: Minimum Spanning Tree - Iris

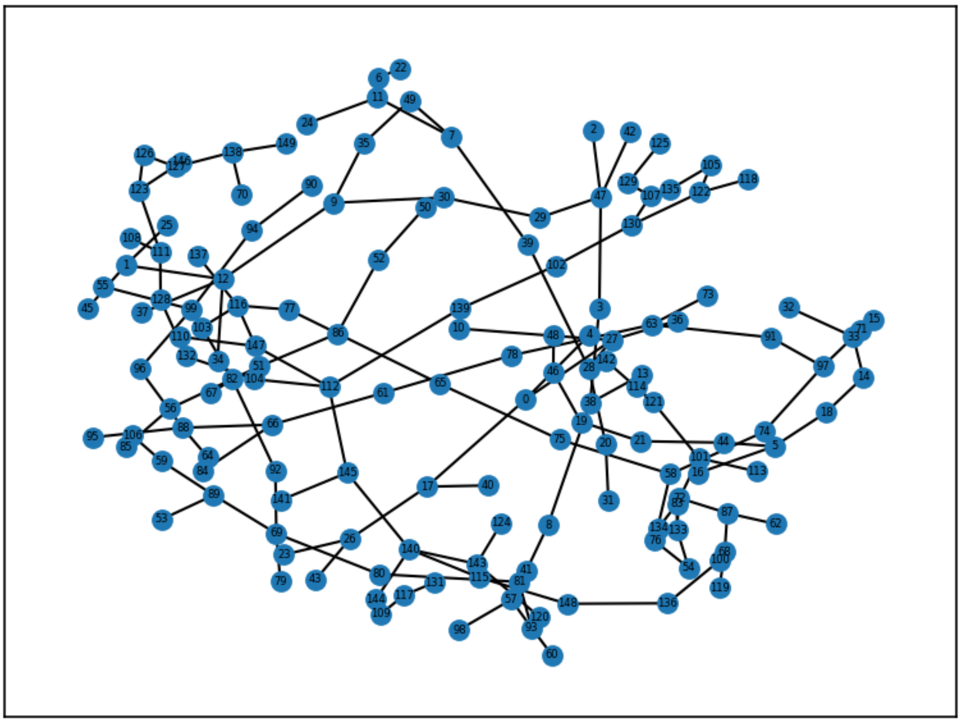
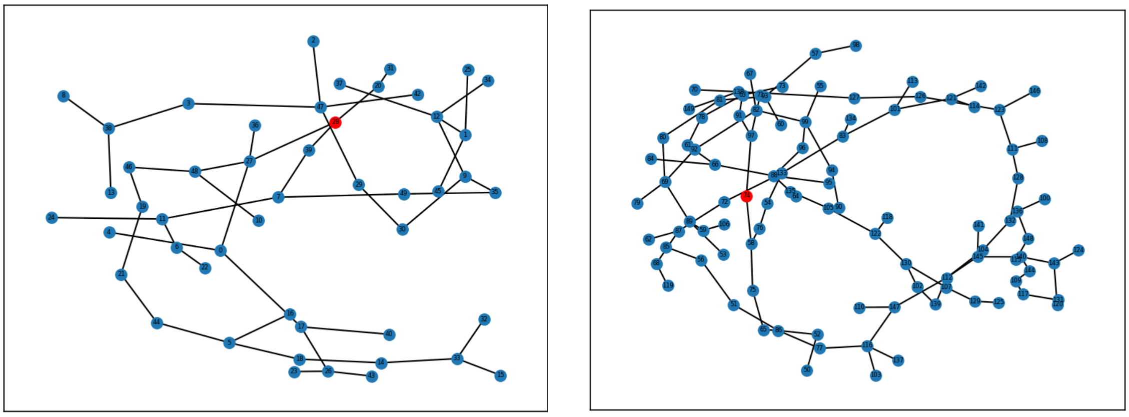


Figure 4: Minimum Spanning Tree Subgraphs - Iris



Subgraph 1

Subgraph 3

Figure 5: Minimum Spanning Tree - Dermatology

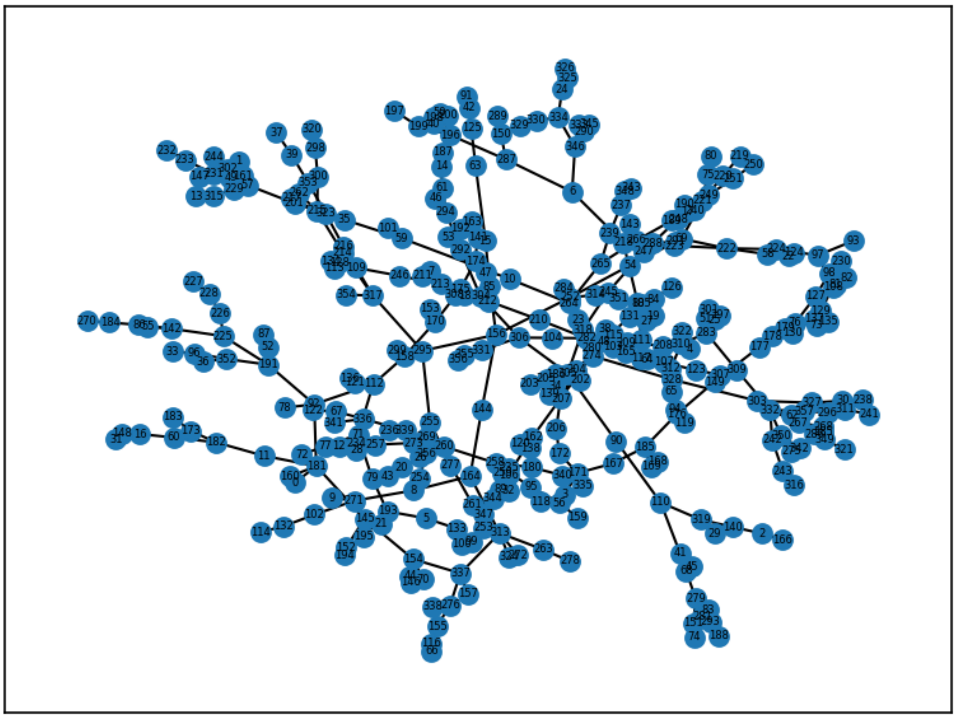
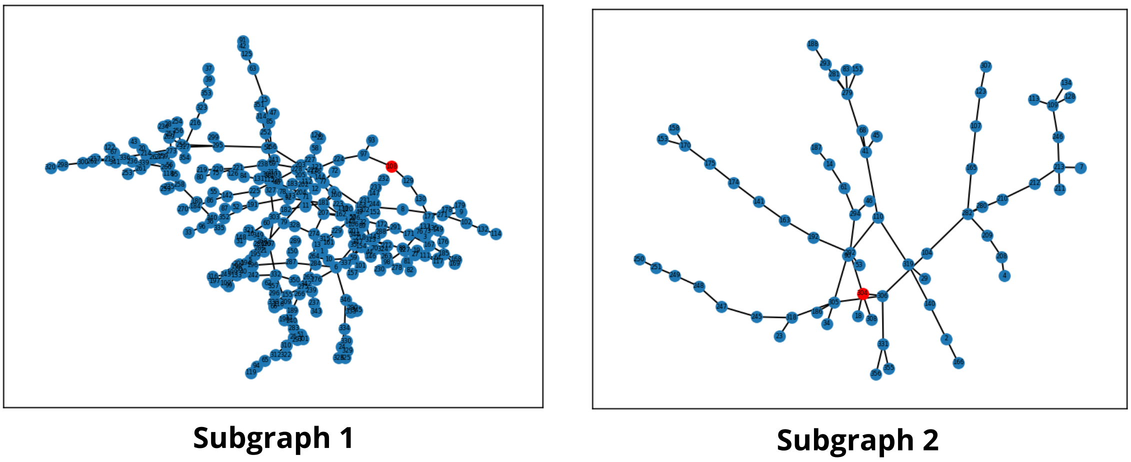


Figure 6: Minimum Spanning Tree Subgraphs - Dermatology



Subgraph 1

Subgraph 2

Figure 7: Minimum Spanning Tree - Isotropic Gaussian Blobs

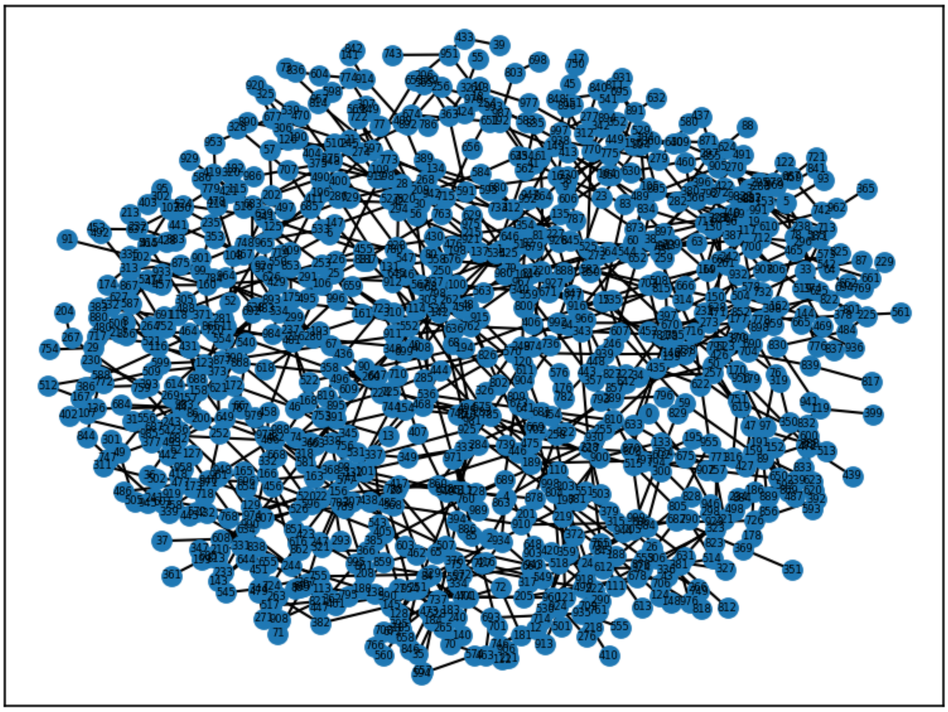
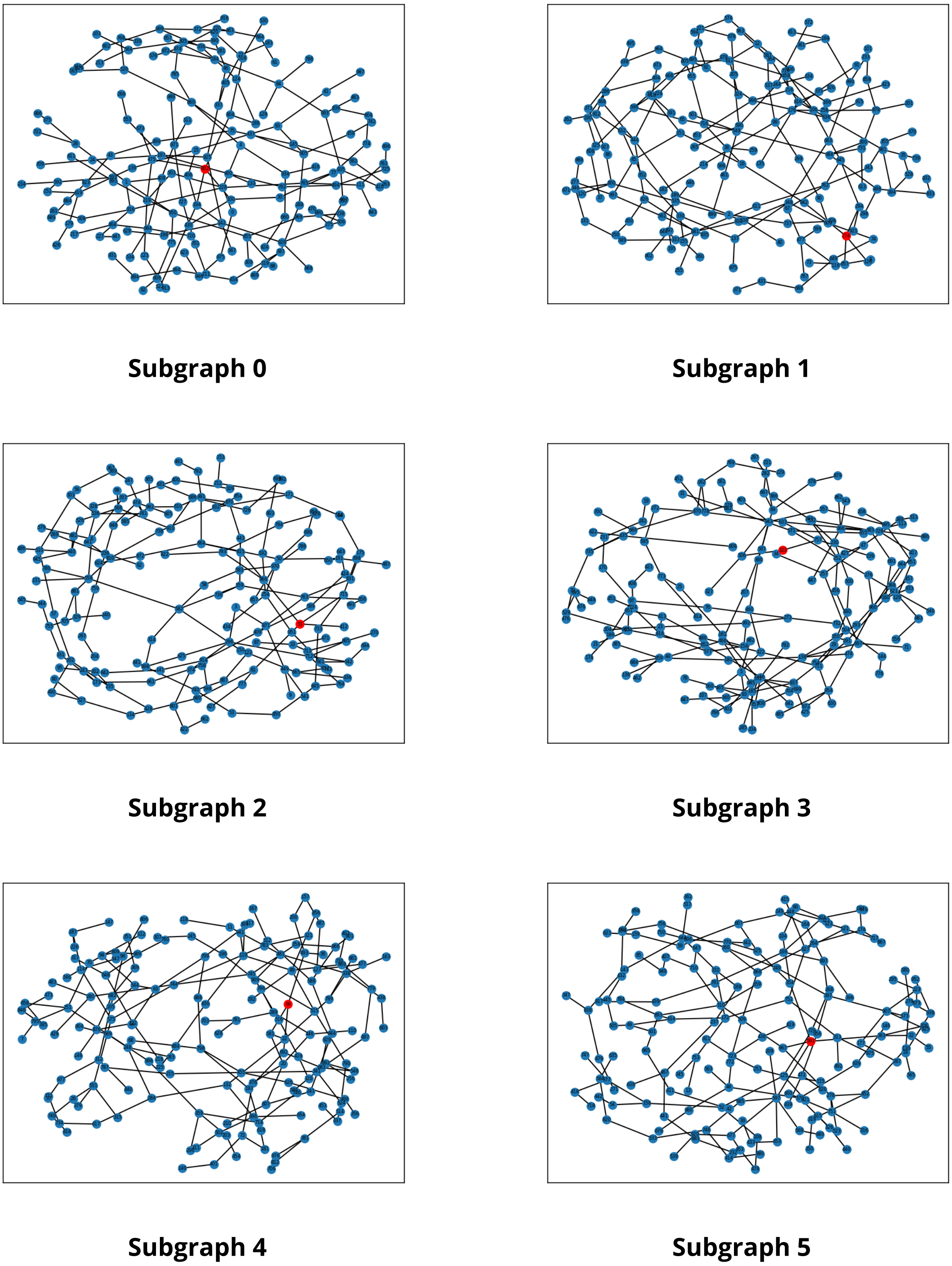


Figure 8: Minimum Spanning Tree Subgraphs - Isotropic Gaussian Blobs



Subgraph 0

Subgraph 1

Subgraph 2

Subgraph 3

Subgraph 4

Subgraph 5