COL761 Homework 2

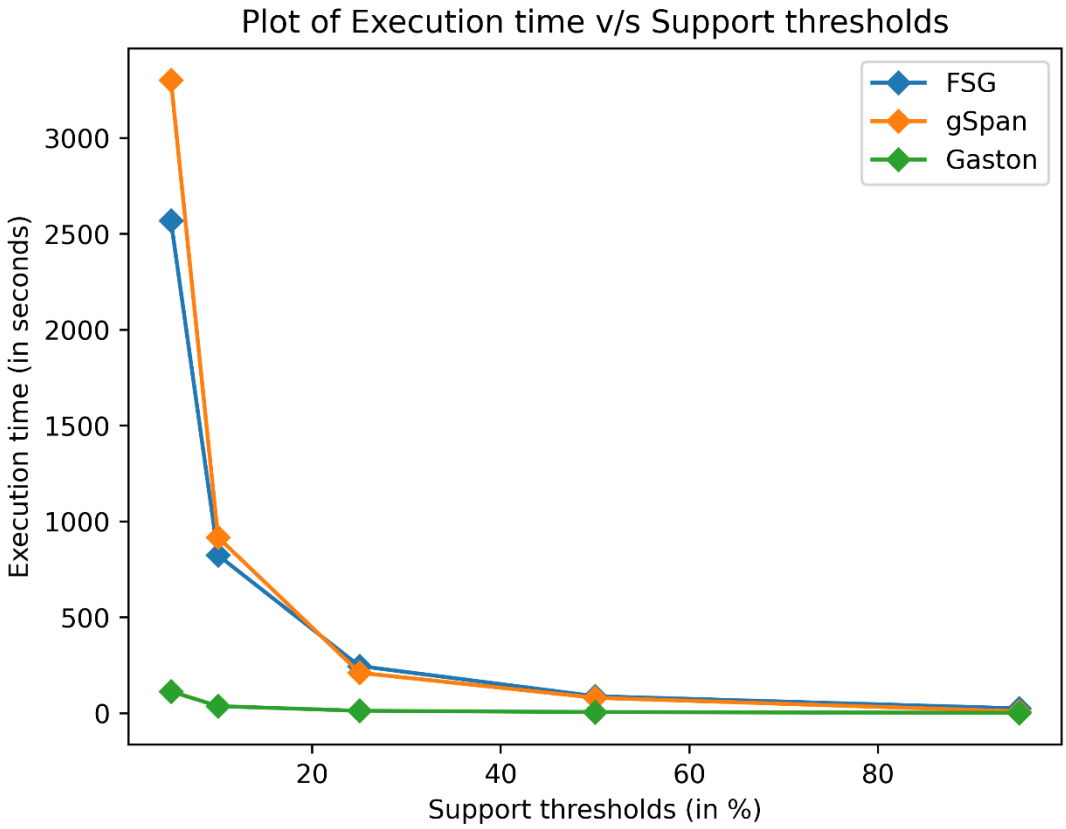
Name: Anwoy Chatterjee

Entry No.: 2022AIB2675

Question 1:

Instruction to run: sh plot.sh <dataset>

Implementations of gSpan, FSG and Gaston were run on the Yeast dataset at minSups = 5%, 10%, 25%, 50% and 95%. Their running times were plotted and are shown in the following graph:



Observations:

1. With decrease in the support threshold the running time increases.
2. The increase in time for gSpan and FSG is greater than Gaston. There is an exponential increase for the former two while the time for latter remains very less.
3. The best out of the three is Gaston- requiring the least amout of time . FSG performs the worst on the given dataset.

Explainations:

1. Lowering the support threshhold results in mining of more frequent subgraphs which increases the time exponentially.
2. FSG works on the principle of breadth- first strategy, it generates all possible candidates at each level and then prunes accordingly. This procedure of redundant candidate generation is very computationally expensive. gSpan and Gaston work on depth-first strategy- hence remove infrequent edges preventing the generation of all candidates and pruning.
3. Gaston is the best among all three because it improves on gSpan by firstly mining out the frequent paths and trees, followed by tress with cycles and then others.
    - Global order on cycle-closing edges is followed by Gaston thus generating those cycles which are bigger than th eprevious ones.
    - Firstly hashing is performed to pre-sort and then an isomorphism test is carried out for duplicate detection. This leads to a very efficient and fast Duplicate detection.

(Ref. : A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, FFSM, and Gaston, Marc Wo ¨rlein, Thorsten Meinl, Ingrid Fischer, and Michael Philippsen https://www2.informatik.uni-erlangen.de/publication/download/PKDD05.pdf)

Question 2:

For finding the frequent subgraphs, which can be potential candidate for features, we have used the frequency threshold as 40%. Next, from the frequent subgraphs found, we retain only those which are discriminative.

To find the subgraphs which are discriminative, we have found a discriminative ratio for each subgraph. Discriminative ratio of a frequent subgraph G is defined as the ratio of the number of graphs in the dataset which contains all frequent subgraphs of G to the number of graphs in the dataset which contains G.

If Discriminative ratio = 1 for any frequent subgraph, then it is completely redundant as the combination of its subgraphs contains all information which it contains.

We have considered a frequent subgraph to be a (discriminative) feature, if its discriminative ratio is greater than 1.4

The value of m (no. of features) thus varies with the dataset considered, according to the number of features (or, subgraphs) which are discriminative among the frequent subgraphs discovered at 40% support threshold.

[**Reference:** Xifeng Yan, Philip S. Yu, and Jiawei Han. 2004. Graph indexing: a frequent structure-based approach. In Proceedings of the 2004 ACM SIGMOD international conference on Management of data (SIGMOD '04). Association for Computing Machinery, New York, NY, USA, 335–346. https://doi.org/10.1145/1007568.1007607]

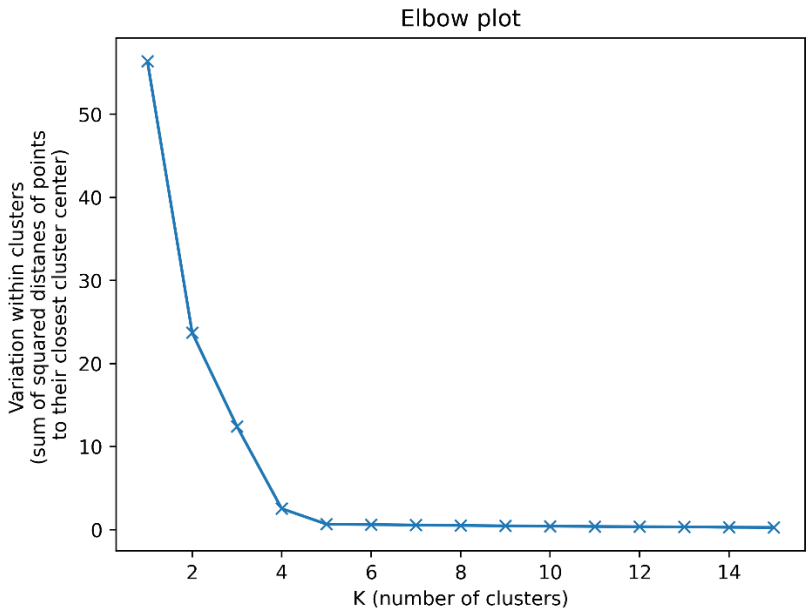
Each line of the output file contains the graph-ids of graphs [ids are not sorted] from the orginal graph dataset containing each query graph.

Question 3:

From elbow plot, we can find the best value of k (i.e number of clusters) to perform k-means clusterung on a given set of datapoints. From the elbow plot, we select the value of k at the point after which the variation within clusters start to decrease linearly.

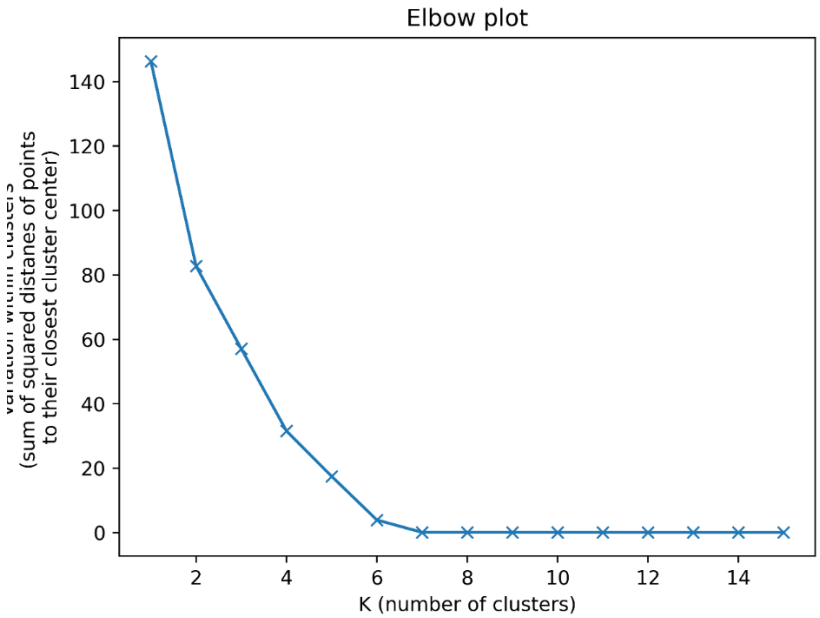
For smaller values of k, the datapoints are not attributed to the correct cluster and as number of clusters are less, variation/distortion within clusters is very which. As k is gradually decreased, the variation also decreases exponentially. But after $k = k_{optimal}$, the rate of decrease of intra-cluster variation falls, and variation starts decreasing in a linear fashion. This is because, when k is increased beyond $k_{optimal}$, the correct clusters are further sub-divided into smaller clusters, and hence it doesn't bring any drastic change to the overall intra-cluster variation, as the new small clusters fall within the previous clusters.

2-Dimensional case:



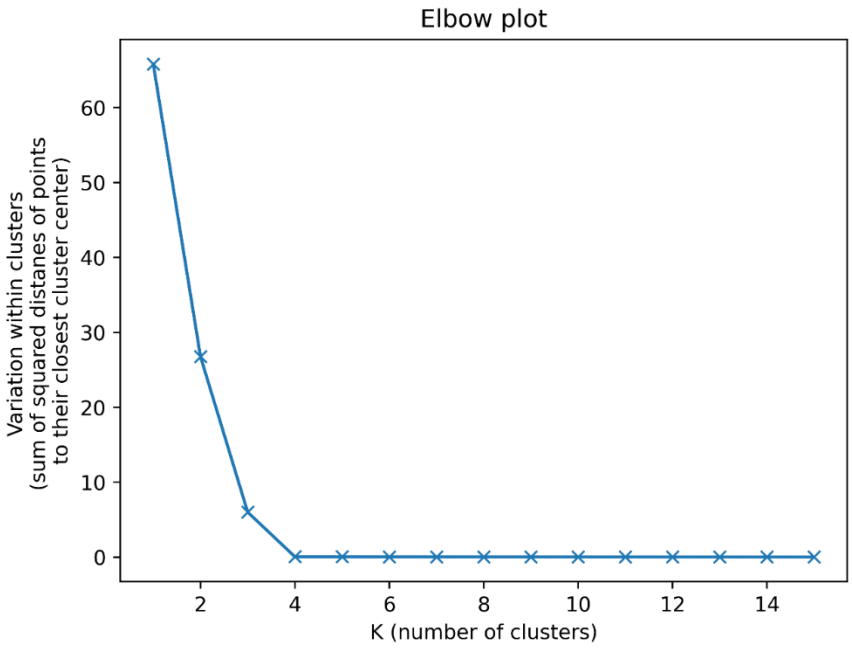
In this case of 2-dimensional datapoints, we observe that the variation starts to decrease linearly after k = 5. Hence, in this case, $\mathbf{k_{optimal} = 5}$

3-Dimensional case:



In this case of 3-dimensional datapoints, we observe that the variation starts to decrease linearly after k = 7. Hence, in this case, $\mathbf{k_{optimal} = 7}$

4-Dimensional case:



In this case of 4-dimensional datapoints, we observe that the variation starts to decrease linearly after k = 4. Hence, in this case, $\mathbf{k_{optimal} = 4}$