COL 761 (Data Mining)

Homework-2

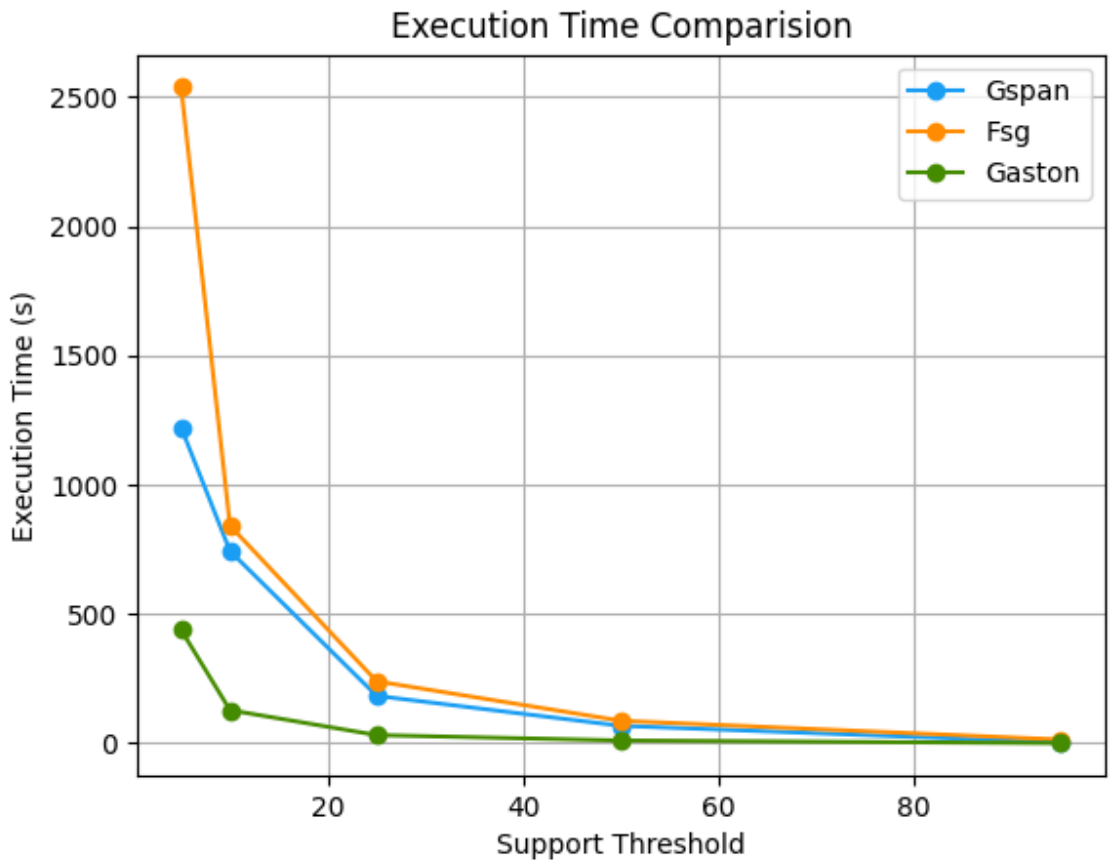
Manish Srivastava	Nikhil Sharma	Vidya Swarupa
(2021MCS2138)	(2021MCS2142)	(2021MCS2158)

Q1: Comparison of various subgraph mining tools.

We used three subgraph mining tools, gSpan, FSG, and Gaston to measure performance of graph mining algorithms. We tested them on the **"yeast"** dataset for varying supports and plotted their runtime with given support threshold.

Observation:

1. Since all three algorithms showing the same trend as support threshold increasing correspondingly Execution time decreasing exponentially.



2.The order of the algorithms based on their running times is as:

FSG > gSpan > Gaston.

3. FSG takes maximum time in all three algorithms since the FSG algorithms uses apriori-based approach and adjacency list representation for storing graphs.It uses multiple database scans, difficulties at mining long patterns.It requires larger memory and execution time because it uses a level-wise joins-based approach which generates large volume of candidate patterns. It scans database many times so have larger run times and not so efficient for large size subgraph patterns.

4. gSpan takes execution time less than FSG but greater than Gaston, reason behind this is gSpan does two things together, it uses subgraph isomorphism test and frequent subgraph growth together into a single procedure for this reason mining process increases in extreme fashion. It uses a DFS lexicographic ordering which makes Tree style lattice structure. Overall its running time is comparable to FSG.

5. Gaston is fastest in all three algorithms since Gaston scans database only once because it uses embedding lists stored in main memory. Gaston does the entire task into three chunks: frequent path searching, then frequent free trees, and cyclic graphs.It is faster than others due to the fact that trees are considered first and graphs with cycle are enumerated at the end.

References:

1. W¨orlein, Marc, et al. ”A quantitative comparison of the subgraph miners MoFa, gSpan, FFSM, and Gaston.” European Conference on Principles of Data Mining and Knowledge Discovery. Springer, Berlin, Heidelberg, 2005

2. https://perso.liris.cnrs.fr/rihab.ayed/ESFSM.pdf

Q2. Subgraph Search

In this question, we were given a graph dataset initially. We were expected to identify all the graphs in this dataset that contain the give query graph Q.

An algorithm to reduce the time is already suggested to us. But several questions remain unanswered.

First of all I will be explaining about the algorithm I used. Then I will try to answer the questions.

1. At first for all the graphs in the databases I extracted the edges out of them and made a set *S*. And for each edge $e \epsilon S$, I maintained a list of graphs L_e in which the edge *e* is present.
2. Now, I ran the **FSG** and got the frequent subgraphs along with the list of graphs in which the subgraph is present. The reason of going with **FSG** instead of **Gastion** and **GSpan** was that I found **FSG** was a little stable compared to **GSpan** and extensively gave the list of matched graphs.
3. Among all the frequent subgraphs that I had, I selected 500 smallest and most feature rich graphs *F* and stored them for the query part. By feature rich I mean I selected only those graphs *g* in which the $|\bigcap_{e \epsilon g} L_e| / |L_g|$ is greater than a threshold.
4. Now I have the L_e for each *e* and the feature rich subgraphs *F* along with the list of graphs D_g in which the given graph *g* is present where $g \epsilon F$. I got the idea of using feature rich graphs by going through the code of one of our senior whose repo link is given in the references part below.
5. So for a given query graph *Q*, I took intersection of the *A* and B where $A = \bigcap_{e \epsilon Q} L_e$ and $B = \bigcap_{g \epsilon F,\, Q \text{ contains } g} D_g$. Let $C = A \cap B$.
6. Now I traversed through all the graphs present in *C.* And performed an expensive subgraph isomorphism test with graph *Q* on them. Still the time these test will take will be less as I am making sure that by the end of step 5, the size of set *C* should be under 500.

What should be the frequency threshold?
A: I fixed the threshold at 50%. For the yeast dataset it **FSG** was pretty fast and ran under 5 minutes. I tried several other threshold values but the total query increasing time was increasing as I was increasing the threshold values. So, I decided to fix the threshold to 50% keeping the indexing time limit given to us.

What should be the value of *m*? The number of frequent subgraphs is likely to be very high. How do you prune it down to m?

A: The value of *m* that we fixed to was 500 + $|\{ G_e | \ e \epsilon S\}|$ where G_e is a single edge graph. I felt that using single edges greatly narrowed down my search space. So keeping these was very beneficial for us in terms of saving time and expensive subgraph isomorphism test.

References:

https://github.com/shubhamguptaiitd/fsg

Q3. Kmeans clustering is implemented

In this algorithm we have used euclidean distance to measure the distance.
In the graph the avg eucleadean distance sum is given on the Y-axis in the cluster.
From the plots it can be clearly seen the elbow plots are forming well and good as per the Output given by mcs212141 data set.

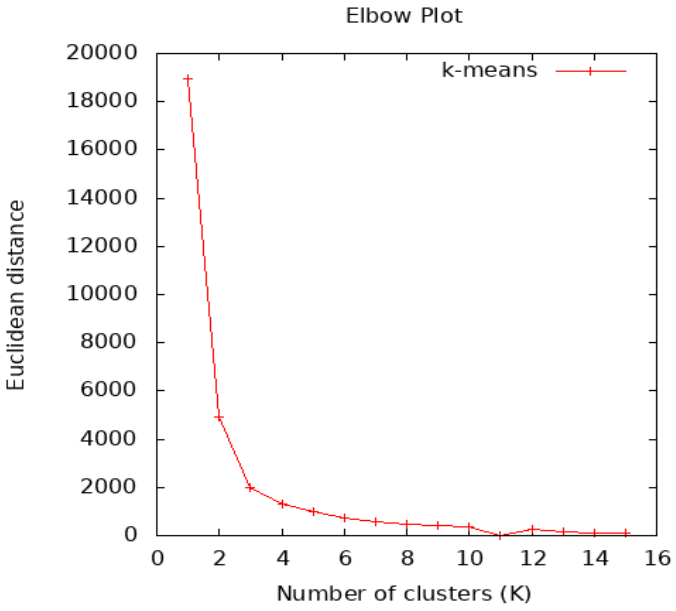


Figure 1. 2- D Dataset

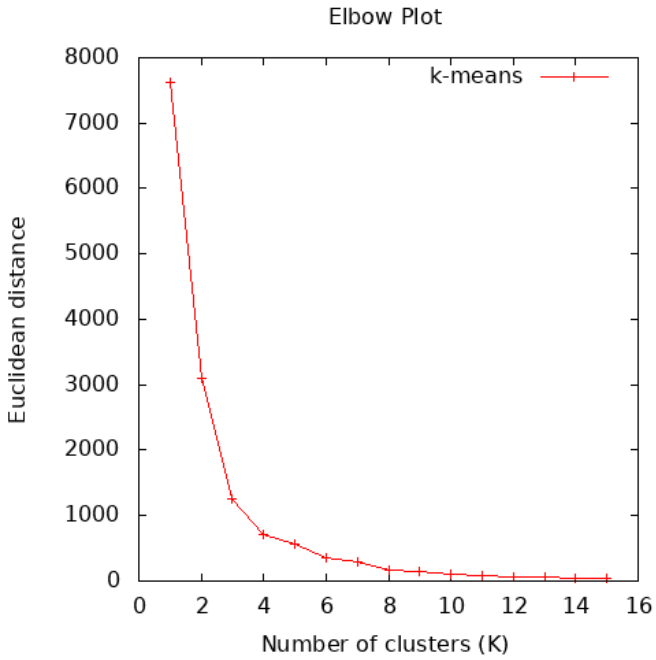


Figure 2. 3-D Dataset

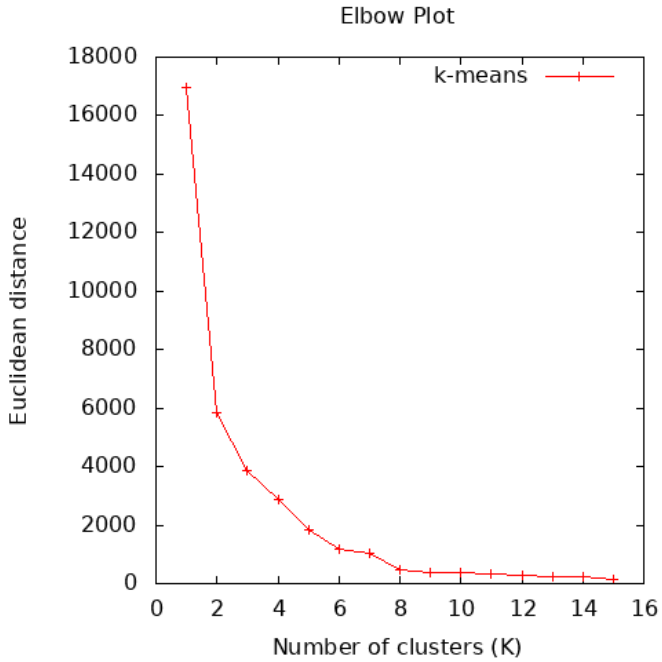


Figure 3. 4-D Dataset

DATASET	MEAN EUCLIDEAN DISTANCE	BEST NUMBER OF CLUSTER VALUE
MCS212141_generated_dataset_2D.dat	~ 2000	3
MCS212141_generated_dataset_2D.dat	~ 900	4
MCS212141_generated_dataset_2D.dat	~ 1900	6