<u>**Assignment-2**</u>

<u>**COL-761**</u>

<u>**By**</u>

Bivas Ranjan Dutta (2021MCS2128)

Kriti Kaushal (2021MCS2135)

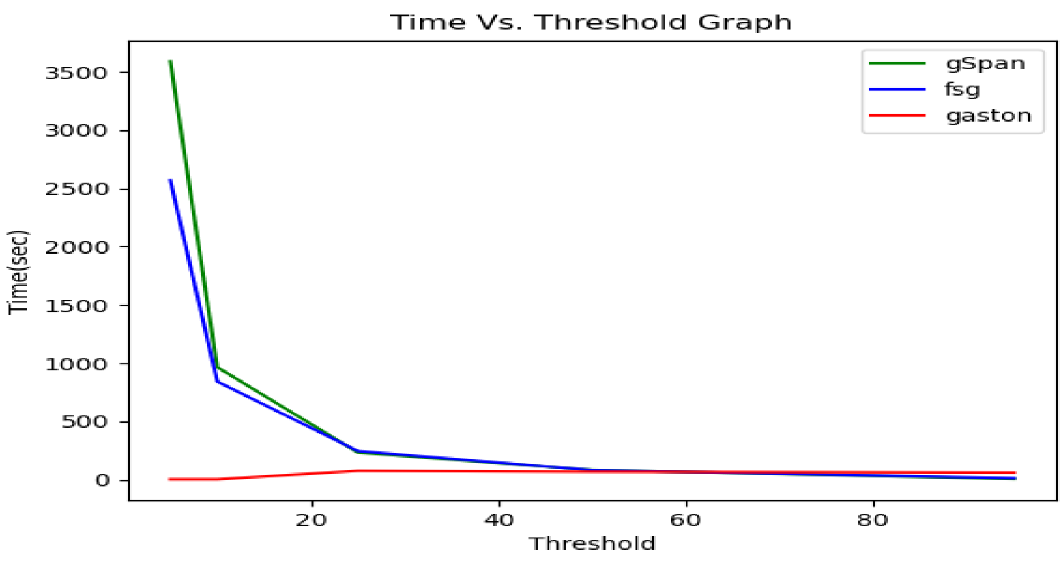
Battala DivyaTeja (2021MCS2126)

Q1.

Computation of frequent subgraphs is a computationally intensive problem as we need to first identify the possible sub-graphs of a similar size and then count the frequency in the target graph.

<u>**Observations:**</u>

If we lower the threshold, the time is increasing exponentially, because at a lower threshold, we are mining more and more subgraphs.
Comparing three of the algorithms side by side, Gaston seems to have the least computation time at any given threshold.
With least threshold as input the Gspan has the greatest computation time. At the maximum threshold as input FSG [Frequent sub-Graph mining] algorithms' computation time even falls below the Gaston algorithm. When given the average threshoold i.e. 50%, it is interesting that all the three algorithms took exactly the same computation time.



Reasoning

Gspan adopts the depth first search strategy to mine the frequent sub-graphs. Gaston and FSG finds the sub-graphs on the level-based approach in which the simple paths are considered and then grows to finding the complex cyclic graphs. As most of the frequent patterns occur in simple forms it is easy for Gaston to find them. Another reason for Gaston to score minimal computation time is because the Gaston uses an occurrence list-based approach which stores all the occurrences of the small graph in the main memory thus simplifying the run time.

FSG uses Apriori property, so it has candidate generation and candidate pruning steps, that takes lot of time, but in Gspan it uses DFS code, so for every step we don't need candidate generation and candidate pruning step, so it is generally faster than FSG.

Gaston breaks the whole task into three smaller tasks: first it searches for frequent paths, second frequent free trees, and at last cyclic graphs. It takes elements which are forest or trees first, and after that computes to normal graphs with cycles.

Q3.

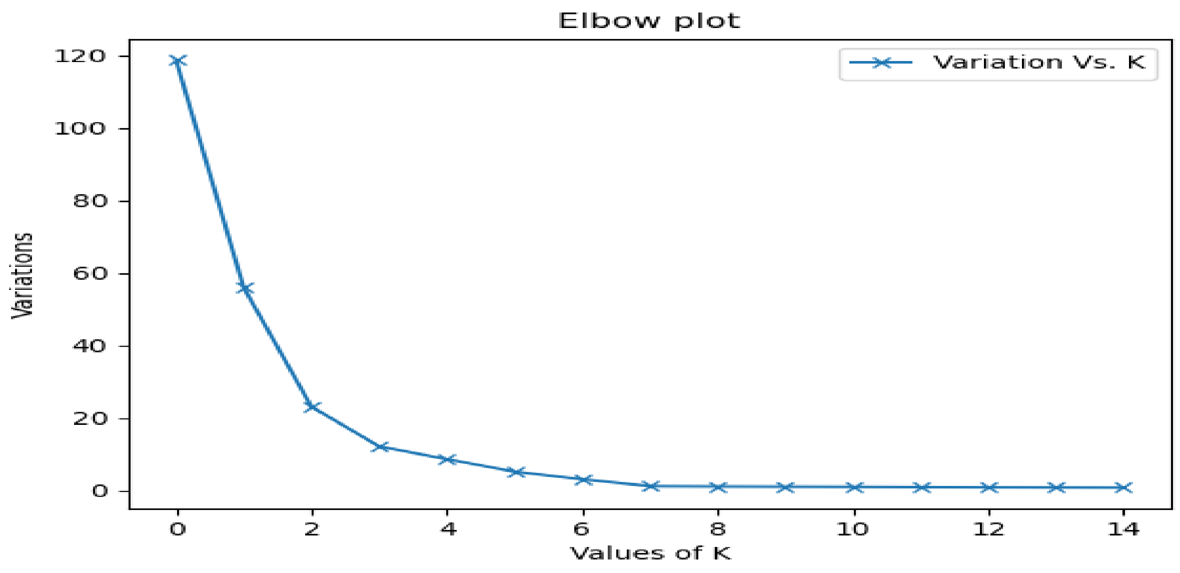
We have generated the dataset from generateDataset_d_dim_hpc_compiled file for 2D,3D and 4D. We have generated the elbow plots to see the optimal K points.

The y axis shows the variations within the clusters and x axis denotes the K values. As the k increases, the variations within the cluster decreases, and in some certain point it becomes stagnant. For the minimum k value, the variation change becomes stagnant or minute change, that K value is called Optimal K value.

The variation within cluster decreases as K increases. As we increase the k values, we are generating more or more clusters, As the number of clusters are increasing, the variation of cluster will decrease because every cluster contain a smaller number of points.

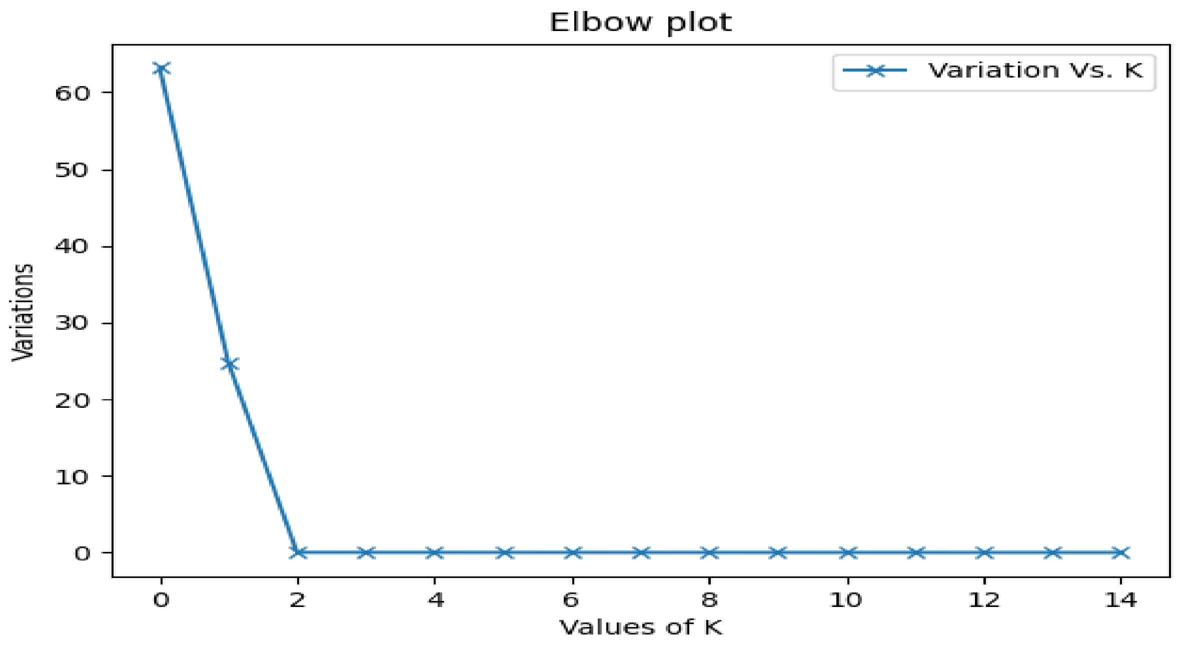
The variation is calculated as square distance of every point to its centroid, so as the k value increases, the variation decreases.

Plot-1: For 2D data:



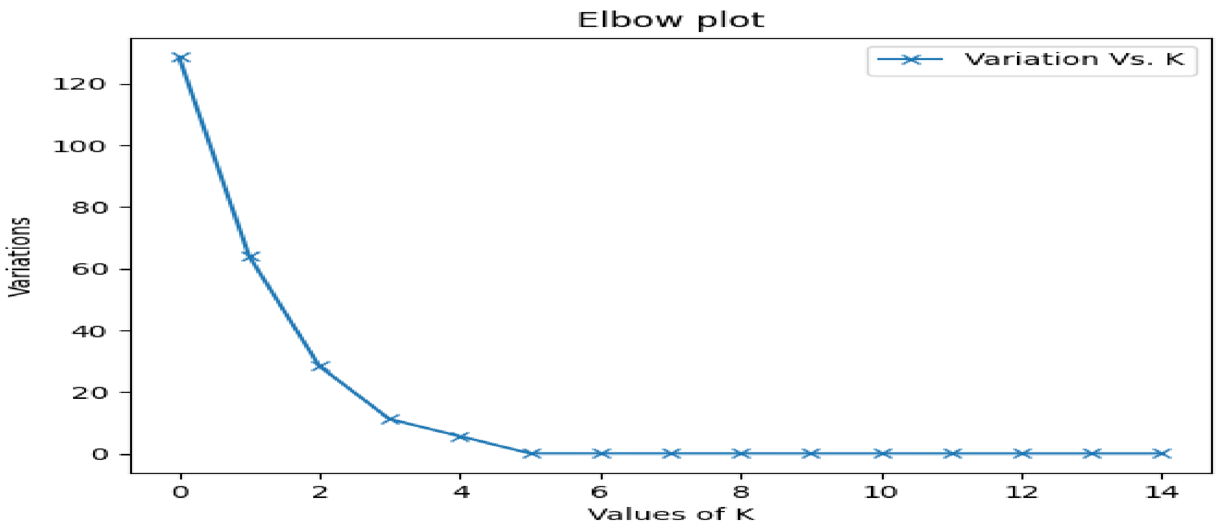
The optimal k value for the dataset is 6.

Plot-2: For 3D data:



The Optimal K value is 2.

Plot-3: For 4D data



The optimal K value is 5.