COL-761 Assignment 2

Kshitij Kumar Singh-2020CS10353
Darshan Rakhewar-2020CS10340
Harshit Rastogi-2020AM11021

January 2024

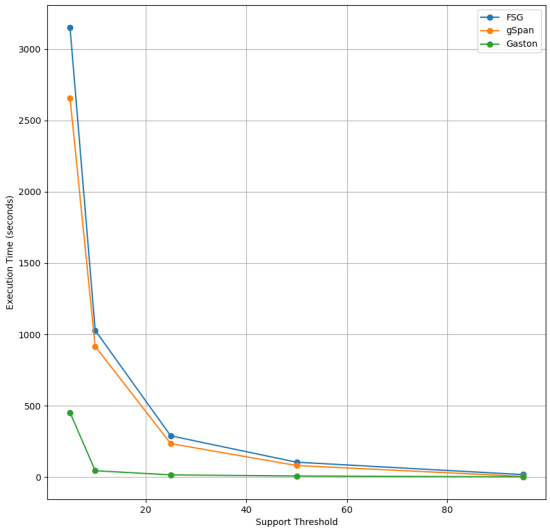
1 Question 1

1.1 Observations

- We observe out of all the 3 algorithms the gaston algorithm performs the absolute best, followed by the Gspan algorithm while FSG algorithm performs the worst.
- Growth sensitivity to support threshold
  - We see an exponential growth trend in number of frequent subgraphs as the support is decreased
  - Decreasing the minimum support threshold leads to exponential growth in the number of potential subgraphs that qualify as 'frequent,' expanding the search space and requiring more isomorphism checks.
  - Conversely, exceptionally high minSup values (e.g., 95 percent) lead to a significant reduction in the count of qualifying frequent subgraphs, potentially resulting in decreased running times due to a more focused search space.
  - Hence we see exponential like growth in the algorithms.
- Reasons for Gaston being the fastest
  - Gaston employs statistical heuristics designed for common datasets, facilitating a quick start to frequent subgraph computation, and also uses efficient data structures for the quick counting of support values for all the frequent items.
  - Gaston innovates in frequency calculation using an occurrence list-based approach, recording occurrences of a small set of graphs in main memory.
  - Gaston uses a level-wise strategy that prioritizes paths proving highly efficient for datasets, especially those in molecular databases.

– Gaston has an efficient polynomial time algorithm for the graph-isomerism problem that uses the idea of using ordering in the graph.

- Gspan uses a DFS strategy to mine frequent subgraphs where as the FSG uses a BFS strategy and as in the DFS strategy the recently used memory is used it leads to less RAM usage and hence results in less page faults, where as the FSG suffers from this affect and hence we see that Gspan outperforms FSG by a margin.
- FSG uses candidate generation and test mechanisms to remove false candidates in principle this results is better storage utilisation but more time consumed in extra computations which leads it to losing to Gspan in time.



support	FSG	Gspan	Gaston
95	16.9	4.42	0.92
50	104.11	80.78	7.352
25	288.98	234.30	15.12
10	1028.26	916.23	44.40
5	3145.84	2656.23	452.40

2 Question 2

2.1 Feature Generation Algorithm

- First of all we read the graph file and separated the graphs into inactive and active based on their labels.

- Now create 2 text files in the format needed by the gSpan algorithm 1 for active graphs and another for inactive graphs.
- We set the support values for active as 0.5 and for inactive as 0.6, these parameters were fine-tuned by us by testing on the data files given.
- Now we find the frequent subgraphs for both the active and inactive labels.
- Now we take the subgraphs in frequent active which are not present in frequent inactive subgraphs and vice-versa.
- Take at max 50 subgraphs from the active and 50 subgraphs at max from the inactive which become our labels for the classification problem.

2.2 Intuition behind the idea

We realized that a feature must be taken if it is present significantly in one class label graph and not significantly present in another label graph. So it is best to run frequent subgraph mining algo for the different class labels separately and then remove the frequent subgraphs that might be present in both the class labels(they may be some kind of common structure present in most molecules). Keeping a mix of features from both active and inactive (50 from both in this case) makes sure that there is each representation for both the labels and hence must improve in reducing the false positives at the testing time and hence will help in better accuracy and precision.

2.3 Results

Below are the results generated by us on a train and test split of 80, 20 in our data

Data	Train ROC-AUC score	Test ROC-AUC score
AIDS	1.0	0.99
Mutagenicity	1.0	1.0
NCI1	1.0	1.0

From the above-shown results we can see that the feature vectors generated are very good as the ROC-AUC score generated for the data is near perfect and hence our algorithm is able to efficiently choose features that help us in differentiating the labels. Hence it can be concluded that the feature vector generated for the SVM by the algorithm are very good.