Frequent Subgraph Mining on the Yeast Dataset

COL761 Assignment 1 - Q2

1 Introduction

Frequent subgraph mining is a fundamental problem in graph data mining, with applications in bioinformatics, chemistry, and social network analysis. Given a database of graphs, the goal is to identify subgraphs that occur frequently under a specified minimum support threshold. However, this task is computationally challenging due to the NP-complete nature of subgraph isomorphism and the exponential growth of candidate patterns at low support thresholds.

In this work, we empirically compare three classical frequent subgraph mining algorithms-FSG, gSpan, and Gaston-on the Yeast dataset. We analyze how their runtimes vary with the minimum support and explain the observed trends by relating algorithmic design choices to the structural properties of the dataset.

2 Methodology

The three algorithms evaluated in this study differ fundamentally in their mining strategies.

FSG (Frequent Subgraph Discovery) follows an Apriori-style breadth-first search approach. It generates candidate subgraphs level by level and relies on the downward-closure property of support for pruning. While conceptually simple, this strategy often suffers from severe candidate explosion at low support thresholds.

gSpan adopts a depth-first pattern growth strategy and uses DFS codes to provide a canonical representation of graphs. This eliminates explicit candidate generation, but requires expensive DFS-code construction and minimality checks during recursive exploration.

Gaston uses a hybrid approach that decomposes the search space into frequent paths, frequent trees, and general graphs. Each class is mined using specialized techniques, and expensive general graph isomorphism is postponed until simpler structures have been exhausted.

3 Experimental Setup

All three algorithms were executed on the Yeast dataset, which consists of labeled, undirected graphs representing biological interaction structures. These graphs are typically sparse, contain repeated node and edge labels, and are dominated by small path- and tree-like motifs.

Each algorithm was run with minimum support thresholds of 5%, 10%, 25%, 50%, and 90%. For each run, the total execution time was recorded. The resulting runtimes were plotted against the minimum support values in a single graph to enable direct comparison across algorithms.

4 Results and Analysis

The runtime plot shows that for all three algorithms, execution time decreases monotonically as the minimum support increases. This is expected, since higher support thresholds reduce the number of frequent subgraphs and consequently the number of subgraph isomorphism checks.

At low support thresholds (5% and 10%), Gaston is significantly faster than both FSG and gSpan. This is because the Yeast dataset is dominated by simple path- and tree-like patterns, which Gaston mines efficiently in its early phases without invoking costly general graph isomorphism.

Between FSG and gSpan, an important observation is that gSpan is slower than FSG at low support on the Yeast dataset. Although gSpan avoids explicit candidate generation, the repeated labels and highly similar local structures in Yeast graphs lead to a large number of DFS extensions and expensive DFS-code minimality checks. The overhead of these canonical checks dominates runtime at low support. FSG, while affected by candidate explosion, incurs lower per-candidate overhead and therefore performs slightly better than gSpan in this regime.

At higher support thresholds (50% and 90%), the runtimes of all three algorithms converge to very small values. In this regime, only trivial subgraphs such as single edges or very short paths remain frequent, making the computational differences between the algorithms negligible.

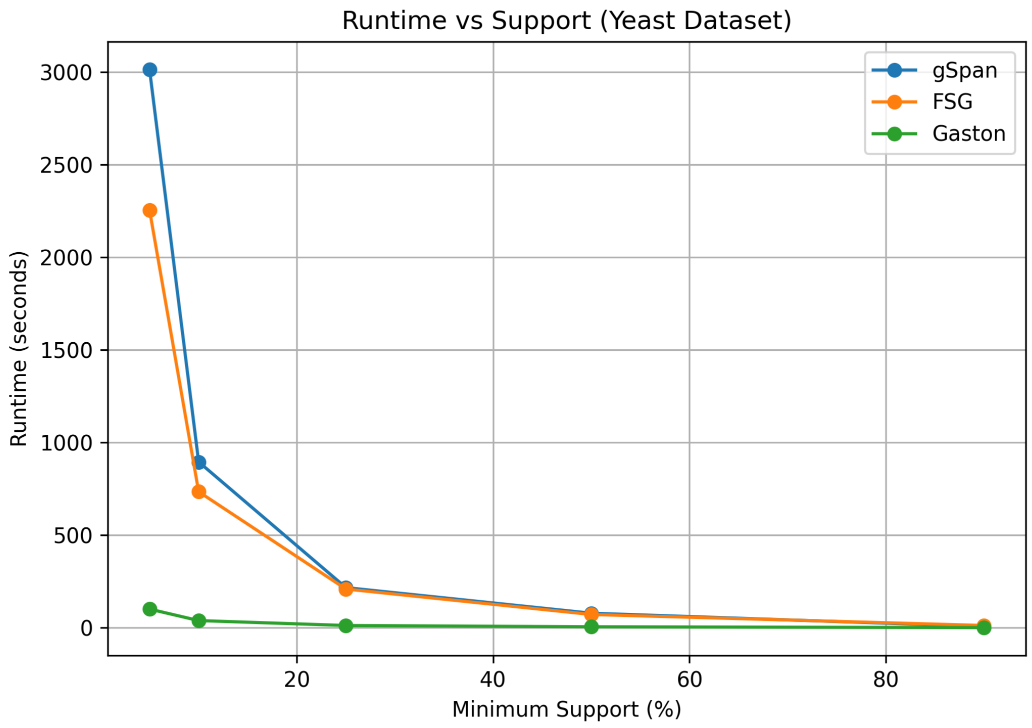

Figure 1: Runtime comparison of FSG, gSpan, and Gaston on the Yeast dataset at different minimum support thresholds.

5 Conclusion

This empirical study demonstrates that algorithmic performance in frequent subgraph mining is strongly influenced by both the mining strategy and the structural properties of the dataset. On the Yeast dataset, Gaston consistently outperforms FSG and gSpan due to its ability to exploit the dominance of path- and tree-like subgraphs. Contrary to common expectations, gSpan performs worse than FSG at low support due to the high overhead of DFS-code minimality checks in the presence of repeated labels and similar motifs. These results highlight the importance of choosing subgraph mining algorithms that are well-matched to the underlying data characteristics.