Assignment 1: Data Mining (COL761)

Part 2: Frequent Subgraph Mining

February 2, 2026

1 Introduction

In this experiment, we analyzed the performance of three distinct frequent subgraph mining algorithms: **FSG** (Frequent Subgraph Discovery), **gSpan** (graph-based Substructure pattern mining), and **Gaston** (Graph/Sequence/Tree extractiON). The objective was to mine frequent substructures from the **Yeast** dataset at varying support thresholds and compare the runtime efficiency of each approach.

2 Experimental Setup

- **Dataset:** The Yeast dataset (Anti-cancer screen data), consisting of approximately 64,110 chemical compound graphs.
- **Support Thresholds:** 5%, 10%, 25%, 50%, and 95%.
- **Environment:** The algorithms were executed on a Linux-based Virtual Machine.
- **Metrics:** Execution time (in seconds) was recorded for each algorithm at each threshold.

3 Results

The execution times for the three algorithms are summarized in Table 1 and visualized in Figure 1.

Table 1: Execution Time (seconds) vs. Support Threshold

Support (%)	**gSpan (s)**	**FSG (s)**	**Gaston (s)**
95	8.52	20.16	1.06
50	108.62	106.86	7.98
25	287.68	310.05	16.36
10	1219.84	1112.39	59.40
5	4693.25	3467.91	131.87

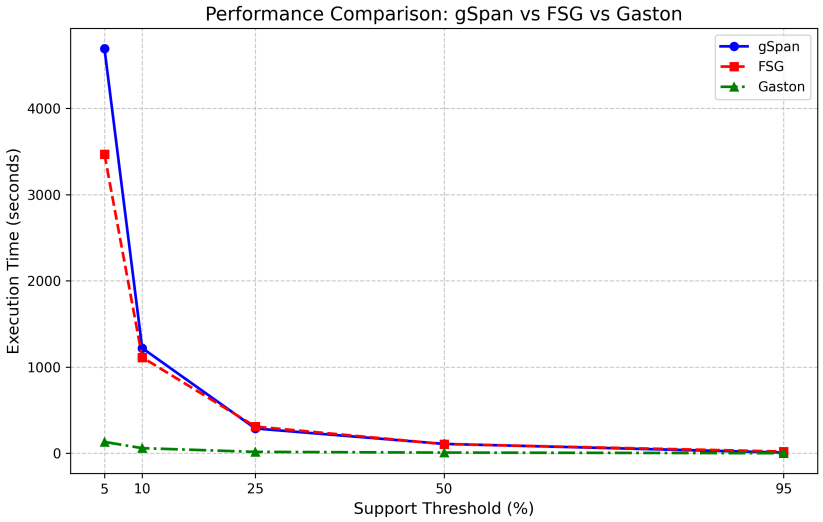

Figure 1: Performance Comparison of gSpan, FSG, and Gaston. Note the exponential growth of FSG and gSpan compared to the linear-like efficiency of Gaston.

4 Analysis of Trends and Growth Rates

4.1 Observed Trends

1. **Inverse Correlation with Support:** As expected, all algorithms show an increase in execution time as the support threshold decreases. Lowering the support results in a combinatorial explosion in the number of frequent subgraphs that must be identified.

2. **Performance Gap at Low Support:** At high support levels (95%), the difference between algorithms is negligible. However, at low support (5%), a massive performance gap emerges. **Gaston** completed the task in approx. 2 minutes, whereas **FSG** and **gSpan** required nearly an hour or more.

3. **The Winner: Gaston** consistently outperformed both gSpan and FSG by an order of magnitude, particularly in the dense search space of 5% support.

4.2 Algorithmic Comparison (Why is Gaston Faster?)

To understand the results, we must look at the underlying mechanisms of each algorithm as described in their respective literature.

4.2.1 1. FSG: The Apriori Bottleneck

FSG is an **Apriori-based** algorithm. It operates on a "generate-and-test" principle. To find a frequent subgraph of size $k+1$, FSG joins two frequent subgraphs of size k.

- **The Cost:** This candidate generation step is computationally expensive. It requires detecting all graph isomorphisms to ensure uniqueness and then verifying if these candidates exist in the database (subgraph isomorphism counting).

- **Result:** As seen in the plot, FSG's runtime grows steeply because the number of candidates explodes at lower supports.

4.2.2 2. gSpan: Pattern Growth

gSpan improves upon FSG by utilizing a **Pattern-Growth** approach (Depth-First Search). Instead of generating candidates and testing them, gSpan recursively extends a frequent graph by adding one edge at a time.

- **The Innovation:** gSpan introduces a canonical labeling system (DFS Code) to detect duplicate graphs without expensive isomorphism checks during generation. It avoids the "join" step of Apriori completely.
- **Observation:** While theoretically superior to FSG in general cases, our experiment showed gSpan performing similarly to (or slightly slower than) FSG at 5% support. This suggests that for this specific dataset and binary implementation, the overhead of maintaining DFS codes was comparable to FSG's candidate generation.

4.2.3 3. Gaston: The "Quickstart" Advantage

The dominant performance of Gaston can be attributed to the observations made by Nijssen and Kok in the Gaston paper.

- **The Principle:** Most frequent substructures in molecular data (like the Yeast dataset) are actually **paths** or **free trees**, rather than complex cyclic graphs.
- **Mechanism:** Gaston splits the mining process into three phases:
  1. *Path Mining:* Very fast ($O(n)$).
  2. *Tree Mining:* Moderately fast.
  3. *Graph Mining:* Expensive (only done when cycles are detected).
- **Why it won:** Since the Yeast dataset consists primarily of chemical structures that are often trees or simple cycles, Gaston handles the majority of the workload using its efficient path/tree solvers. It only switches to the expensive general graph mining algorithm for the small fraction of cyclic patterns. In contrast, gSpan and FSG treat every structure as a general graph from the very beginning, incurring unnecessary overhead.

5 Conclusion

The experiment confirms that while generic graph mining algorithms like gSpan and FSG are robust, domain-aware algorithms like **Gaston** offer superior performance for molecular datasets. By exploiting the structural properties of chemicals (high prevalence of paths and trees), Gaston avoids the combinatorial complexity that causes FSG and gSpan to degrade at low support thresholds.