Comparative Analysis of FSG, gSpan, and Gaston Algorithms

Team: PyMining [2025SIY7623, 2025SIY7622, 2022TT11351]

February 4, 2026

1 Performance Comparison

The empirical results (Figure 1) illustrate the runtime performance of FSG, gSpan, and Gaston across varying minimum support thresholds.

1.1 Observations

- **Exponential Growth:** All algorithms exhibit an increase in runtime as the support threshold decreases. This is due to the exponential explosion in the number of frequent fragments discovered at lower thresholds.
- **Algorithm Efficiency:** Gaston demonstrates superior performance, particularly at low support thresholds. While FSG (Apriori-based) and gSpan (DFS-based) show a sharp exponential curve, Gaston remains significantly faster.

2 Theoretical Justification

The performance gap between these techniques is explained by their underlying enumeration and pruning strategies:

1. **FSG (Level-wise):** Uses an Apriori-like join operation to generate candidates. The overhead of generating and then pruning candidates, combined with multiple subgraph isomorphism tests per candidate, makes it the least efficient on large datasets.
2. **gSpan (Pattern-growth):** Avoids candidate generation by using DFS codes and right-most extensions. This significantly reduces memory usage and processing time compared to FSG.
3. **Gaston (Refined Complexity):** Leverages the fact that paths and trees are easier to mine than cyclic graphs. By using specialized algorithms for simpler structures, Gaston minimizes the time spent on the NP-complete subgraph isomorphism problem.

3 Conclusion

Gaston is the preferred choice for this dataset due to its ability to handle low support thresholds without the exponential time penalty seen in Apriori-based methods. Its "Quickstart" approach provides the most scalable solution for molecular and graph-transactional data.

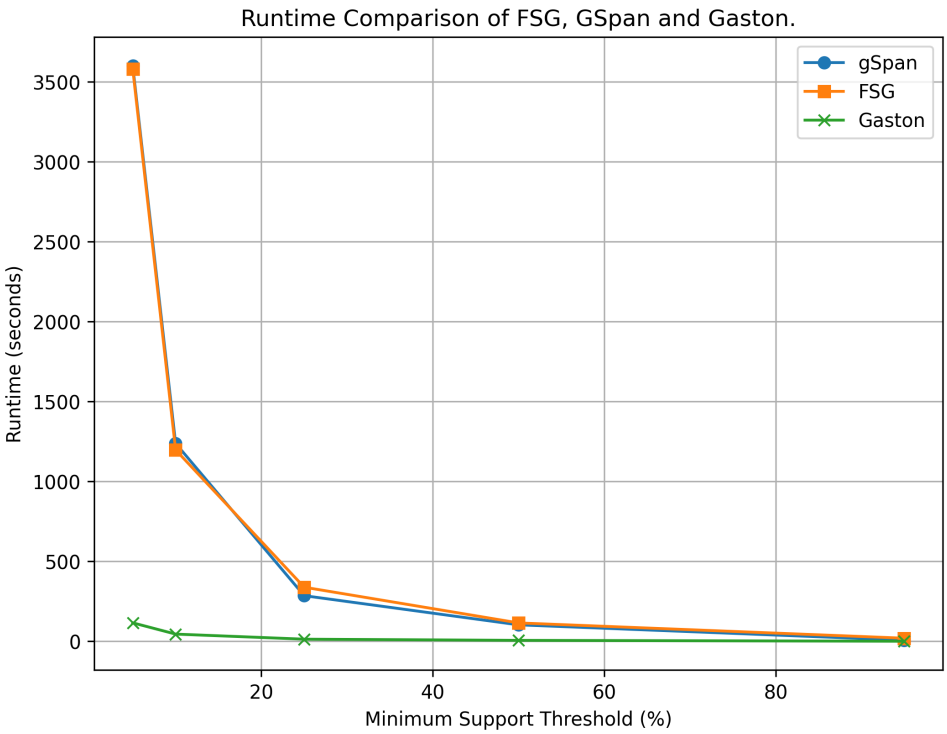

Figure 1: Runtime comparison of FSG, gSPan and Gaston across support thresholds.