COL761: Assignment 1 - Part 3

Abhay Sharma 2012CS50272
Akash Dabaniya 2021CS10111
Mridul Singh 2018CS50412

February 4, 2026

1 Task 3: Graph Indexing

Problem Overview

The objective of this task is to perform efficient subgraph isomorphism queries on a large graph database. A naïve approach (checking isomorphism against every graph) is computationally prohibitive ($O(|D| \cdot NP)$).

To solve this, we implemented a **feature-based indexing strategy**[3]. We mine discriminative subgraph fragments to create binary feature vectors for the database graphs. At query time, we filter out non-promising candidates using the inclusion-exclusion principle, significantly reducing the candidate set size $|C_q|$ passed to the expensive verification step.

2 Methodology

2.1 Step 1: Discriminative Subgraph Mining

Instead of using random subgraphs or simply the most frequent ones, we developed a pipeline to identify features that maximize information gain.

1. Mining with Gaston

[2]We utilized the Gaston algorithm for Frequent Subgraph Mining (FSM).

- **Why** Based on our experiments in Part 2, Gaston consistently outperformed gSpan and FSG in runtime.
- **Support Threshold:** We set a low minimum support (approx. 2%) to capture a wide variety of potential features, ensuring we didn't miss rare but discriminative patterns.

2. Feature Selection Metric (Variance Maximization)

A feature is only useful for indexing if it can distinguish between graphs. A feature present in 100% of graphs or 0% of graphs provides zero pruning power.

We scored each mined subgraph f based on the variance of a Bernoulli trial:

$$Score(f) = P(f) \times (1 - P(f))$$

Where $P(f) = \frac{\text{support}(f)}{|D|}$.

This metric favors subgraphs with support close to **50%**. These features theoretically split the database in half, maximizing the pruning power of each bit in the index.

3. Redundancy Pruning via Jaccard Similarity

Simply picking top-scoring features often results in selecting variations of the same substructure. To ensure orthogonality in our feature space, we implemented a redundancy check:

1. For every new candidate feature, we compared its transaction ID set (T_{new}) with the transaction ID sets of already selected features ($T_{selected}$).

2. We calculated the Jaccard Similarity:

$$J(T_{new}, T_{selected}) = \frac{|T_{new} \cap T_{selected}|}{|T_{new} \cup T_{selected}|}$$

3. If $J > 0.90$, the new feature was deemed redundant (co-occurring too frequently with an existing feature) and was discarded.

2.2 Step 2: Index Construction

For the database D and query q, we constructed binary feature vectors V_D and V_q of length $K = 50$ (where K is the number of selected features).

$$V[i] = \begin{cases} 1 & \text{if feature } f_i \subseteq G \\ 0 & \text{otherwise} \end{cases}$$

This mapping was performed using 'NetworkX' for subgraph isomorphism checks.

2.3 Step 3: Candidate Generation (Filtering)

We generated the candidate set C_q using the necessary condition for subgraph isomorphism:

> *If query q is a subgraph of g, then every feature present in q must also be present in g.*

Using bitwise operations, we retained a graph g_i in the candidate set only if:

$$(V_{g_i} \text{ AND } V_q) == V_q$$

This operation is extremely fast and effectively prunes the search space.

3 Conclusion

By combining the speed of Gaston for mining, a variance-based scoring metric for feature selection, and Jaccard-based pruning for diversity, our approach constructs a compact and highly discriminative index. This minimizes the size of the candidate set $|C_q|$, thereby optimizing the competitive score defined as $S_q = |R_q|/|C_q|$.

Disclaimer

Large Language Models were used during this assignment for clarifying the workflow of graph indexing pipelines and for assistance with LaTeX syntax. The implementation logic and experimental conclusions are our own work.

References

[1] X. Yan and J. Han, "gSpan: Graph-Based Substructure Pattern Mining," *ICDM*, 2002.

[2] S. Nijssen and J. Kok, "The Gaston Tool for Frequent Subgraph Mining," *Electronic Notes in Theoretical Computer Science*, 2005.

[3] X. Yan, P. S. Yu, and J. Han, "Graph Indexing: A Frequent Structure-based Approach," *SIGMOD*, 2004.