Q3: Graph Indexing for Subgraph Search

Kushagar Garg (2022CS11088) & Arya C Nari (2022CS11129)

1 Problem Overview

Subgraph isomorphism is a classical NP-complete problem. Given a query graph q and a database graph g, deciding whether $q \subseteq g$ requires exponential time in the worst case. For a database of thousands of graphs, performing full subgraph isomorphism checks at query time is infeasible.

The goal of this assignment is to design an **indexing-based filtering mechanism** that significantly reduces the number of database graphs that must be checked using full subgraph isomorphism, while **guaranteeing correctness**. Formally, for each query graph q, we must produce a candidate set C_q such that:

$$R_q \subseteq C_q$$

where R_q is the true result set of graphs that contain q as a subgraph. The objective is to minimize $|C_q|$, thereby improving the score:

$$\text{score}_q = \frac{|R_q|}{|C_q|}$$

2 Indexing Strategy

2.1 Feature-Based Necessary Conditions

We adopt a **feature-based graph indexing approach**, where each graph is represented as a vector over a fixed set of subgraph features. Each feature represents a small labeled subgraph (e.g., an edge or a 2-path). A feature can only be present in a supergraph if it is present in the subgraph, making it a **necessary condition** for subgraph isomorphism.

If a query graph activates a feature that a database graph does not, then the database graph can be safely pruned.

3 Feature Design

3.1 Initial Feature Choices

We experimented with the following feature types:

- **Labeled edges** $(\text{node\_label}_u, \text{edge\_label}, \text{node\_label}_v)$
- **Labeled 2-paths** $(u \xrightarrow{e_1} v \xrightarrow{e_2} w)$ with node and edge labels

Edges provide universal coverage (every non-empty graph has edges), while 2-paths capture stronger structural constraints.

3.2 Frequency-Based Feature Selection

Not all features are useful. Very common features appear in almost all graphs and offer little pruning power, while extremely rare features rarely help queries.

We therefore applied **frequency windowing** over the database:

- Minimum frequency: 3 graphs
- Maximum frequency: 8% of database size

This idea is inspired by toxicophore selection strategies in mutagenicity research, where statistically uninformative substructures are discarded in favor of mid-frequency, high-information patterns.

4 Initial Failures and Debugging

4.1 Weak Pruning Despite Sparse Features

After implementing frequency filtering, we observed that database graphs had sparse feature vectors ($\approx$9 active features per graph), yet candidate sizes were still close to the full database ($\sim$4300 graphs).

To diagnose this, we measured feature activation in query graphs:

```
Avg active query features: 0.0
Min: 0, Max: 0
```

This revealed a critical issue: **most queries activated no features at all**. In such cases, the condition:

$$q_{\text{vec}} \leq g_{\text{vec}}$$

is trivially satisfied for all database graphs, resulting in no pruning.

4.2 Edge Fallback Attempt and Failure

We attempted to fix this by forcing the inclusion of edge features. However, the fallback failed because edges were still subjected to the same frequency upper bound as 2-paths. Since query graphs often contain *very common edges*, these edges were filtered out during feature mining.

As a result, queries still activated zero features.

5 Corrected Feature Strategy

5.1 Decoupled Feature Rules

We fixed the issue by **decoupling feature rules**:

- **2-path features**
  - Strict frequency window (rare and discriminative)
- **Edge features (fallback)**
  - No upper frequency bound
  - Only minimum frequency constraint

This guarantees that every non-empty query activates at least one feature, while still allowing strong pruning when more complex features are present.

6 Structural Necessary Conditions

In addition to feature vectors, we introduced **structural metadata constraints**, which are mathematically necessary for subgraph isomorphism:

- Number of nodes
- Number of edges
- Maximum node degree

A database graph is discarded if it violates any of these constraints relative to the query. These checks are computationally trivial and provide additional pruning for free.

7 Aggressive Optimization: Feature Multiplicity

7.1 Limitation of Binary Features

Binary (presence/absence) features are weak for small queries. For example, if a query contains three edges of a given type and a database graph contains only one, a binary representation would still allow the database graph to pass.

7.2 Count-Based Edge Features

To address this, we upgraded **edge features from binary to count-based**:

- Edge features store **multiplicity**
- 2-path features remain binary

The filtering condition remains:

$$q_{\text{vec}} \leq g_{\text{vec}}$$

but now operates over counts rather than booleans, significantly strengthening pruning while preserving correctness.

8 Final Candidate Filtering Algorithm

For each query–database pair:

1. Check structural metadata constraints
2. Check feature vector dominance ($q \leq g$)
3. Retain database graph if both pass

This pipeline guarantees:

- **No false negatives**
- Strong pruning for large queries
- Reasonable pruning even for small queries

9 Results and Observations

- Initial candidate sizes: ∼4300 (no pruning)
- After frequency filtering alone: negligible improvement
- After edge fallback: moderate improvement
- After multiplicity-aware features: **2–5× reduction in candidate size**

Some very small queries remain difficult to prune, which is a known theoretical limitation of necessary-condition indexing.

10 Discussion

This assignment highlights an important principle in graph indexing:

Necessary conditions alone cannot prune aggressively for very small queries.

However, careful feature design, fallback mechanisms, and count-aware constraints can significantly improve performance while preserving correctness.

The final solution balances:

- safety
- computational efficiency
- practical pruning power

11 References

1. Ullmann, J. R. *An Algorithm for Subgraph Isomorphism.* Journal of the ACM, 1976.
2. Cook, D. J., Holder, L. B. *Mining Graph Data.* Wiley-Interscience, 2007.
3. Helma, C. et al. *Derivation and Validation of Toxicophores for Mutagenicity Prediction.* Journal of Chemical Information and Modeling, 2004.
4. Yan, X., Han, J. *gSpan: Graph-Based Substructure Pattern Mining.* ICDM, 2002.