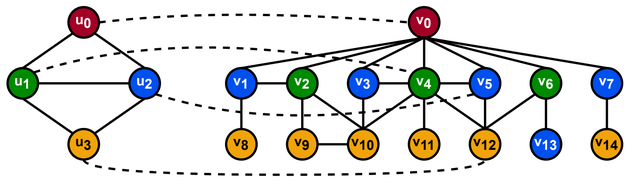
Graphlet-based Filtering for High-Performance Subgraph Isomorphism Search

Syed Ibtisam Tauhidi[1], Arindam Karmakar[2], Thai Son Mai[1], Hans Vandierendonck[1]

[1]Queen's University Belfast, [2]Tezpur University

Introduction

The `Subgraph Isomorphism (SI) search` problem involves finding embeddings of a *pattern* (or *query*) graph within a larger *data* graph.



An embedding of a pattern graph (left) in a data graph (right).

However, because of being NP-complete, no efficient polynomial-time algorithm for the problem is known. Nevertheless, researchers have proposed heuristics that leverage the properties of the pattern and data graphs to improve the runtime efficiency by pruning the search space.

Overview of heuristics - Filtering

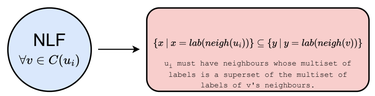
Many heuristics for the `SI` problem can be divided into three stages. In the first stage, i.e., *filtering*, for each pattern graph vertex, u_i, the candidate set of potential mapping data graph vertices, $C(u_i)$, is generated.

Two frequently used techniques for filtering are -

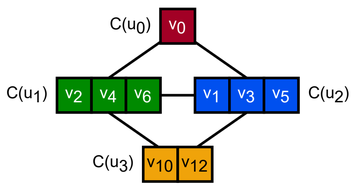
(a) **Label and Degree Filtering** (`LDF`)

Overview of `LDF` filtering.

(b) **Neighbourhood Label Filtering** (`NLF`)



Overview of `NLF` filtering.

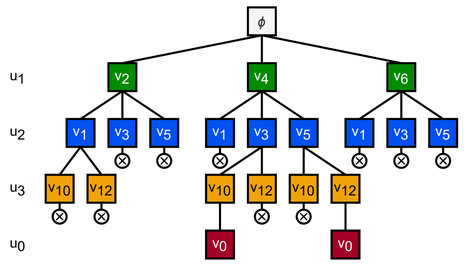


Candidate sets after `LDF` and `NLF` filtering.

Both `LDF` and `NLF` filtering techniques are applied in all heuristics in the `In-Memory Subgraph Matching` [3] codebase.

Overview of heuristics - Ordering and Searching

In the next stage, i.e., *ordering*, the order in which the vertices of the pattern graph are checked for successful mapping with the vertices in the candidate sets from the data graph is determined. The Highest Degree First (`HDF`) strategy is used by many heuristics, including Ullmann [4] and VF2 [1].

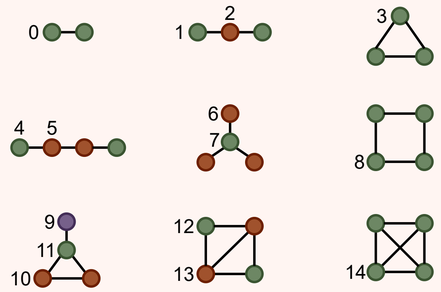


Search tree after `LDF` and `NLF` filtering and `HDF` ordering.

In the final stage, a backtracking search is performed to find the `SI` embeddings.

Graphlets and Orbits

Graphlets (or *motifs*) are commonly recurring *small* subgraphs in a large graph and are often used to characterise the internal structure of a graph. Orbits are groups of vertices of the graphlets with respect to their automorphism, defining the roles of the vertices within a graphlet.



Fifteen orbits in the nine connected graphlets of up to size four vertices (K_4).

We used the `ORbit Counting Algorithm (ORCA)` [2] to extract the orbit counts in linear time.

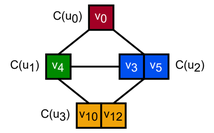
Our Proposed Approach

We propose Graphlet Filtering (`GLF`), to be applied after `LDF` and `NLF` filtering, based on the observation that for any pattern vertex, u_i, its orbit count for any particular orbit, o_j, must be smaller than or equal to the orbit count of o_j for each candidate data vertex in $C(u_i)$.

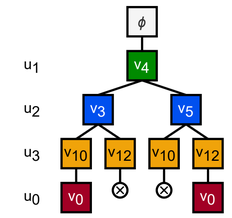
For example, the count of orbit 3 (i.e., vertex of a triangle graphlet) for u_3 is two, while v_1 belongs to only one triangle in the data graph. Therefore, further filtering of $\{v_1\} \notin C(u_2)$ using `GLF` filtering is obtained.

Running Example

A more restrictive filtering is observed on applying `GLF` filtering, in addition to `LDF` and `NLF` filtering. This results in enhanced pruning in the associated search tree.

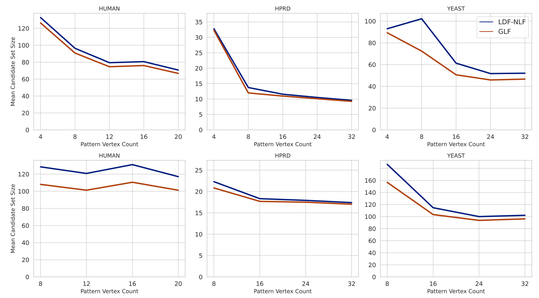


Candidate set after `GLF` filtering.



Search tree after additional `GLF` filtering.

Preliminary Results



Candidate sets sizes in dense (above) and sparse (below) pattern graphs.

Preliminary experiments with three datasets, i.e., `HUMAN`, `HPRD` and `YEAST`, show that `GLF` filtering is able to perform additional filtering of up to 12.93% in the `YEAST`, 10.64% in the `HUMAN` and 4.2% in the `HPRD` datasets. Across all datasets, filtering enhancement of 9.93% in the `SPARSE` and 8.49% in the `DENSE` pattern graphs can be observed.

Orbit or graphlet-based (`GLF`) filtering can be beneficial for more restrictive filtering, thereby pruning the search tree in the `SI` problem. Our study can potentially reduce the execution time of algorithms for pattern discovery and graph database query resolution in various domains.

References

[1] Luigi P Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento.
A (sub) graph isomorphism algorithm for matching large graphs.
IEEE transactions on pattern analysis and machine intelligence, 26(10):1367–1372, 2004.

[2] Tomaž Hočevar and Janez Demšar.
A combinatorial approach to graphlet counting.
Bioinformatics, 30(4):559–565, 2014.

[3] Shixuan Sun and Qiong Luo.
In-memory subgraph matching: An in-depth study.
In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1083–1098, 2020.

[4] Julian R Ullmann.
An algorithm for subgraph isomorphism.
Journal of the ACM (JACM), 23(1):31–42, 1976.