Graphlet-based Filtering for High-Performance Subgraph Isomorphism Search

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

* *Queen's University Belfast, Belfast, Northern Ireland, UK*
† *Tezpur University, Tezpur, Assam, India*

ABSTRACT

The Subgraph Isomorphism (SI) search problem involves searching for the embeddings of a pattern graph in a given data graph. Several heuristic algorithms in the literature speed up the execution of the NP-complete problem by pruning the search space using various techniques. Many of these algorithms can be divided into three stages. In the first stage, filtering is performed for each pattern graph vertex to select a subset of the vertices from the data graph for potential mapping. In this study, we propose a technique to augment the filtering capabilities of existing algorithms using graphlet (orbit) counts of the pattern and data graphs vertices. Our proposed technique enhances the filtering capabilities in the current heuristic algorithms by up to 15.38%.
KEYWORDS: subgraph isomorphism; graphlet; orbit; filtering

1 Introduction

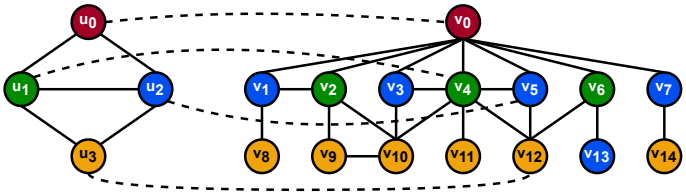

Fig. 1: Subgraph isomorphism mapping between a pattern graph (left) vertices mapped to the corresponding vertices in a data graph (right).

Graphs are widely used in computer science to represent various types of structured data from domains like astronomy, bioinformatics, social and computer networks, etc. Graph analytics deals with the analysis of such graph data structures. The Subgraph Isomorphism (SI) search problem is recurringly used in graph analytics. The SI problem involves the discovery of the embeddings of a *pattern* (or *query*) graph in a given *data* graph. Fig. 1

[1]E-mail: stauhidi01@qub.ac.uk

shows an embedding of a pattern graph in a data graph. The SI problem finds applications in various tasks like pattern discovery and graph database query resolution. However, because of being NP-complete, no polynomial-time algorithm for the SI search problem is known. However, multiple heuristics have been proposed in the literature that uses various characteristics of the pattern and data graphs to prune the exhaustive search space. We discuss the heuristics in more detail in Section 2.

2 Overview of the heuristic algorithms

Multiple heuristic algorithms for the SI problem can be divided into three stages, the first of which is the filtering stage. In this stage, for each vertex, u_i, of the pattern graph, the candidate sets of vertices from the data graph, $C(u_i)$, is generated. These sets contain the filtered set of vertices from the data graph that the particular pattern vertex may potentially be mapped to. Two techniques for filtering that are universally applied in many heuristics, including all available heuristics in the `IMSM` [SL20] codebase, are (a) Label and Degree Filtering (`LDF`) and (b) Neighbourhood Label Filtering (`NLF`).

The `LDF` filtering ensures that for each u_i, $C(u_i)$ contains only those vertices from the data graph that have (a) the same label as u_i and (b) degree equal to or greater than u_i. For the pattern and data graphs in Fig. 1, the candidate sets after applying `LDF` filtering are $C(u_0) = \{v_0\}$, $C(u_1) = \{v_2, v_4, v_6\}$, $C(u_2) = \{v_1, v_3, v_5\}$, and $C(u_3) = \{v_9, v_{10}, v_{12}\}$.

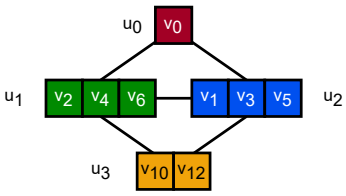

Fig. 2: Candidate sets after `LDF` and `NLF` filtering.

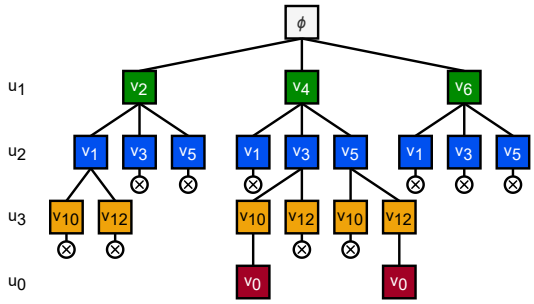

Fig. 3: Search tree after `LDF` and `NLF` filtering and `HDF` ordering.

The `NLF` filtering, applied after the `LDF` filtering, ensures that for each u_i, $C(u_i)$ is further constrained to contain only those data graph vertices whose neighbours have a multiset of labels that is a superset of the multiset of labels of the u_i's neighbours. There are no changes in the candidate sets, $C(u_0)$, $C(u_1)$, and $C(u_2)$, after `NLF` filtering, but $C(u_3)$ is updated to $C(u_3) = \{v_{10},\ v_{12}\}$. The candidate sets for the pattern graph vertices in Fig. 1 using its data graph after `LDF` and `NLF` filtering are shown in Fig. 2.

After filtering, the next stage determines 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. The Highest Degree First (`HDF`) strategy is used by many algorithms, including Ullmann [Ull76] and VF2 [CFSV04]. When two vertices have the same degree, we randomly pick one before the other. For the pattern graph in Fig. 1, we obtain $\{u_1, u_2, u_3, u_0\}$ as the search order using `HDF` ordering. In the third and final stage, a backtracking search is performed to find the SI embeddings. Fig. 3 shows the search tree to find the embeddings of the pattern graph in the data graph in Fig. 1 after `LDF` and `NLF` filtering, and `HDF` ordering.

3 Our Proposed Approach

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. The **ORbit Counting Algorithm (ORCA)** [HD14] is often used for the efficient extraction of orbit counts. The fifteen orbits obtained using the nine connected graphlets of size up to four are shown in Fig. 4.

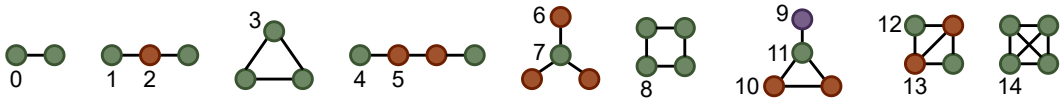

Fig. 4: Fifteen orbits in the nine connected graphlets of up to size four.

We propose a new stage of filtering, to be applied after LDF and NLF filtering, named Graphlet Filtering (GLF). The GLF filtering is 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 $C(u_i)$.

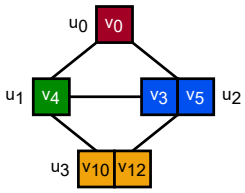

Fig. 5: Candidate set after GLF filtering.

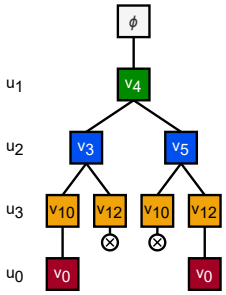

Fig. 6: Search tree after additional GLF filtering.

We observe that the count of orbit 3 (i.e., vertex of a triangle graphlet) for u_3 is two (i.e., u_3 belongs to two triangles in the pattern graph), while v_1 belongs to only one triangle in the data graph. Therefore, we obtain further filtering of $\{v_1\} \notin C(u_2)$ using GLF filtering. Similarly, $\{v_2,\ v_6\} \notin C(u_1)$ because u_1 has a count of one for orbit 13 (i.e., 3-degree vertex in 4-chordal-cycle graphlet), while v_2 and v_6 have a count of zero for orbit 13 (i.e., they do not belong to any 4-chordal-cycle graphlet). Fig. 5 shows the candidate sets for the pattern graph in Fig. 1 on applying GLF filtering after LDF and NLF filtering. Fig. 6 shows the search tree using HDF ordering after GLF filtering.

4 Preliminary Results

In our preliminary experiments with three datasets, i.e., HUMAN, HPRD and YEAST, we observe that GLF filtering is able to perform additional filtering of 12.93% in the YEAST, 10.64% in the HUMAN and 4.2% in the HPRD datasets. Across all datasets, we obtain a filtering enhancement of 9.93% in the SPARSE and 8.49% in the DENSE pattern graphs. In both SPARSE and DENSE pattern graphs, we get filtering enhancements up to 15.38% in different datasets. Fig. 7 shows the mean candidate sets size before and after GLF filtering.

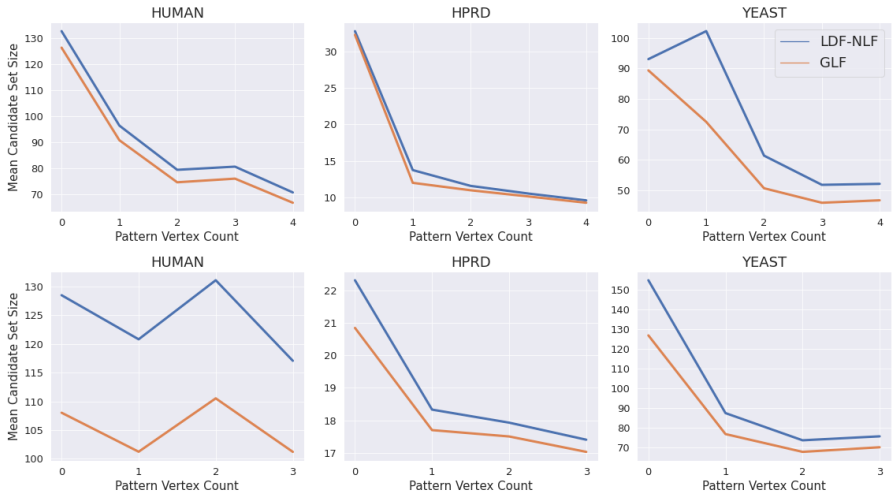

Fig. 7: Candidate sets sizes in dense (above) and sparse (below) pattern graphs

5 Conclusion

We observe in our preliminary studies that orbit or graphlet-based (`GLF`) filtering can be beneficial for more restrictive filtering, thereby pruning the search tree in the Subgraph Isomorphism problem. We tested the `GLF` filtering technique in combination with `LDF` and `NLF` filtering techniques. Multiple heuristic algorithms in the literature use various data structures and methods for filtering [SL20]. Future studies would analyse the effect of the `GLF` filtering technique in the runtime of the various heuristic algorithms. Our study has the potential to reduce the execution time of algorithms for pattern discovery and graph database query resolution.

References

[CFSV04] 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.

[HD14] Tomaž Hočevar and Janez Demšar. A combinatorial approach to graphlet counting. *Bioinformatics*, 30(4):559–565, 2014.

[SL20] 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.

[Ull76] Julian R Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, 23(1):31–42, 1976.