Notification Number: F. NO. COE/ Ph.D./(Notification)/519/2022

Name of Scholar	**: Zubair Ali Ansari**
Name of Supervisor	**: Dr. Jahiruddin**
Name of Department	**: Computer Science**
Title	**: Efficient Algorithms for Subgraph Matching in Big Graphs**

Keywords: Graph searching, subgraph matching, subgraph isomorphism, subgraph matching algorithms, eccentricity, embedding, graph decomposition, large graphs

The remarkable expansion of large networks through numerous internet resources, along with their affordable accessibility, turned these large and complex networks into the demanding source of relevant information. Many large and complex networks such as social networks, citation networks, biological networks, etc., can be modeled by graphs. In a graph, network entities (objects) are represented by vertices, and interactions between them are represented by edges.

Depending on the nature of the networks, the resulting data graphs are very diverse and have many vertices and edges. Analyzing such large and complex data graphs to retrieve demanding information is an emerging research challenge and has potential applications useful to both industry and academia. However, in such a data graph crucial graph query primitive at the heart of many complex network analysis is to find efficiently and effectively all copies of the specified query graph. Finding all copies of the specified query graph into the associated data graph is a well-known subgraph matching problem. Because the subgraph matching problem is computationally expensive, finding all copies of the query graph in a data graph becomes more difficult when the data graph is diverse and large.

Researchers have presented many algorithms to deal with such subgraph matching problems that include TurboISO, QuickSI, VF2, Glasgow, and RI. However, most of them are inefficient when the size of the data graph is too large. Moreover, these algorithms show exponential behavior for some sets of queries and data graphs pairs.

To reduce the computational cost of the subgraph matching algorithm, it is crucial to select a pivot vertex (a vertex to start the matching process) of the query graph. Researchers have proposed different objective functions to determine the pivot vertex. However, to the best of our knowledge, none of the foregoing objective functions consider minimizing the size of candidate regions to select a pivot vertex. As the candidate region is a portion of the data graph that may have some isomorphic images (i.e., embeddings) of the query graph, to find such embeddings of a given query graph in the associated data graph, we need to explore all possible candidate regions of the data graph. In this regard, we have proposed an objective function to identify the pivot vertex in the query graph.

Using the objective function to locate the pivot vertex of the query graph, we have designed and implemented a subgraph matching algorithm SubISO to find n-embedding of the query graph. We have also compared the performance of SubISO with three popular state-of-the-art subgraph matching algorithms, namely TurboISO, QuickSI, and RI over three benchmark datasets -- Human, Yeast, and Hprd. We observed that SubISO performs significantly better in terms of execution time, and the number of embeddings found.

A study on some specific queries (aka straggler queries) revealed that the existing subgraph matching algorithms show exponential behavior to find their matches into the associated data graph. Recently, researchers have evaluated the exponential behavior of five existing subgraph matching algorithms-- GraphQL, SPath, QuickSI, TurboISO, and BoostISO on different pairs of query and data graphs. Their evaluations show that all contemporary algorithms have a couple of straggler queries, which are algorithm and data specific. Researchers have suggested many solutions like taking a certain time limit to find matches of straggler queries, and randomly try to solve straggler queries using other algorithms as straggler queries are algorithm specific. However, more focus is needed to deal with straggler queries. In this regard, we have proposed a solution of limiting the recursive calls in the SubgraphSearch( ) function of the SubISO algorithm. As a result, the SubISO algorithm finds many matches of identified straggler queries within a very short period.

To handle large graphs as data graphs, the subgraph matching algorithms lag in time efficiency. To deal with this issue, many researchers decomposed or compressed the data graph and then searched copies of the query graph in the resulting data graph. In this line, we have proposed a subgraph matching algorithm SubGlw that decomposes a data graph into several small-sized candidate subgraphs based on the broader sense of the divide-and-conquer problem-solving paradigm. As the candidate subgraph is a subgraph of the data graph that may contain several copies of the query graph, the decomposition of such data graph is carried out in such a way that the size and count of candidate subgraphs are optimal. After the decomposition, the proposed SubGlw algorithm finds all copies of the query graph in the candidate subgraph by exploring successively each candidate subgraph as the Glasgow algorithm does. The performance of SubGlw is empirically evaluated and compared with two state-of-the-art subgraph isomorphism solvers -- SubISO and Glasgow over three benchmark datasets -- Yeast, Human, and Hprd. The experimental findings reveal that SubGlw performs significantly better in terms of both embedding count and execution time. We have also presented an analysis for identifying saddle point, a timeout at which the SubGlw algorithm achieves maximum embeddings with the least execution time. Analysis on saddle point provides a better understanding for parameter settings in the proposed SubGlw algorithm.

Subgraph matching algorithms can be divided into three categories, namely search tree-based algorithms, constraint programming-based algorithms, and graph indexing-based algorithms. Search tree-based algorithms are very time-efficient, while constraint programming-based algorithms are remarkably effective in proportion to the number of solutions found. RI is one of the fastest search tree-based algorithms for solving subgraph matching problems. However, RI is not as effective as constraint programming-based algorithms. For a well-performing subgraph matching algorithm, time efficiency is crucial, but other performance measures such as the number of solutions found (i.e., effectiveness of the algorithm), index size, and the average number of recursive calls are also significant. To enhance RI's effectiveness without compromising its efficiency, we have proposed an enhanced version of RI termed as RI+, using candidate region-based decomposition and ordering. In this line, we have proposed several candidate region orderings that uses the structural properties of the candidate regions. We have compared the performance of RI+ with RI on two performance measures that include count-of-embeddings and search time over three benchmark datasets -- Human, Yeast, and Hprd. On comparative analysis, we observed that RI+ showed a significant improvement over both the performance measures. Moreover, on the empirical analysis, we observed that on changing the candidate region ordering in the proposed RI+ algorithm the search time of RI+ improves significantly.