Interactive Visual Pattern Search on Graph Data via Graph Representation Learning

(Comprehensive Summary)

Suvi Varshney *

I. Objective

The paper aims to support interactive graph pattern search by providing a visual analytics framework using graph representational learning. They utilize existing *Graph Neural Networks (GNNs)* for subgraph creation, decision, and node alignment, essentially converting them to high dimensional fixed length vectors. This allows their methods to scale the search as the graph scales. In *GNNs*, they propose a novel, **NeuroAlign** method for node alignment. The visual implementation of their system, **GraphQ**, is able to provide query and result modification through visual interaction.

II. Contributions

The outline of the contributions are:

1) They come up with a visual analytics system for human-in-the-loop, example-based graph pattern search using graph representational learning.
2) Node-alignment is done through a novel approach, **NeuroAlign**, which they claim is 10-100 x faster than the baseline combination algorithms.
3) The framework is implemented is called **GraphQ**, which provides an interactive query display supporting modification based on visual results.

III. VIS Designs / Techniques Used

GhaphQ consists of 5 major parts:

1) Query editing panel which represents the subparts of the query and their interrelation to each other using a graph. Here we can modify the query.
2) Query result panel displaying an overview of all the sub-graphs which are the result of a query and the user-selected sub-graph in detail.
3) A statistics and filtering panel helps the user to visualize the statistics of the query results using scatterplots, bar-graphs, and tabular format.
4) A toggle switch panel to enable/disable fuzzy matching and to highlight/unhighlight the matches.
5) A popup window to compare the structure of the query and the structure of the matched graphs visually.

IV. ML Methods Used

The graph matching problem is solved by two *GNN* frameworks, **NeuroMatch** and the novel **NeuroAlign**. **NeuroMatch** decomposes the query and the graph into several subregions for an effective search. **NeroAlign** is used to improve node alignment, by working on fixed-length embeddings. The output from the **NeuroAlign** is passed to a multi-layer preceptron which takes a pair of embeddings and returns a similarity score. This is also useful when the user wants to perform a fuzzy match in addition to an exact match.

V. Strengths

1) The authors encode the topology and nodes of a graph into fixed-length vectors. Then the graph search essentially is a high dimension vector comparison problem, much cheaper to solve.
2) To achieve an efficient node-alignment, they propose novel **NeuroAlign**, which is 10-100 x faster than combination algorithms and 19-29% more accurate than existing deep learning graph search algorithms.
3) **GraphQ** helps the user to visually interact with the query structure and the results and to modify them using visual assistance.

VI. Shortcomings

1) **NeuroAlign** currently isolates only isolates subgraphs isomorphic to the query in a single graph. However, in practice, we have multidimensional graphs, where more than one subgraph can match the query.
2) **NeuroMatch**, an integral part of **GraphQ**, is often slow in large, densely connected graphs. This problem amplifies when it has to search all the subgraphs even when they are not an exact match.
3) The algorithm is only available for undirected graph since the current backbone of *GNN* do not support directed graphs.

VII. Future Work

The authors plan to extend the application to index-based search in graph databases, searching on the index embeddings to achieve sub-linear time performance. Further, this implementation could also be extended to highly dense social media networks and 3D point clouds.

VIII. Question for Discussion

As we have seen in many visualization projects focusing on interactive graph searching algorithms, they focus on the path taken to discover relevant nodes. Will such a visualization be helpful? Will it help to debug complex queries?

*MS in CS at UC Davis, ID: 920256895, suvarshney@ucdavis.edu