INTRODUCTION

- Graph mining is the process of extracting information from graph data.
- It is a special form of structured data mining.
- It has broad areas of applications from pattern discovery to pattern matching.
- It also expands over a large spectrum of fields from studying chemical compounds to social and biological networks.
- In this project, we have attempted to find the best possible graph mining algorithm for a specific use case and using existing code bases with modifications to analyze the application of the technique to the use case.

BACKGROUND RESEARCH

Graph Mining Techniques

- Frequent Subgraph Mining, Correlated Graph Pattern Mining, Optimal Graph Mining, Approximate Graph Pattern Mining, Graph Pattern Summarization, Graph Classification, Graph Clustering, Graph Indexing, Graph Searching, Graph Kernels, Link Mining, Web Structure Mining, Workflow Mining, Biological Network Mining.
- Graph Mining Technique Selection
  - Since we are trying to find similar graphs any technique that checks for exact match was eliminated. Approximate graph pattern matching allows difference in topology up to a specified threshold limit and thus perfectly suits our use case.
  - Graph clustering performs the task of grouping similar graphs together using different matching techniques. Our idea was to use approximate graph mining technique as the underlying technique for graph clustering, which was defined as the future scope.

Graph Clustering

- Groups similar graphs together.
- Can use any graph matching algorithm to find similar graphs.
- Approximate graph pattern matching
  - Unlike most of the algorithms it does not look for exact match but mines for similar graphs.
  - Depending upon the implementation one can decide the percentage of similarity for match.

- NeMa: Network matching algorithm.
- Compares a query (smaller) graph with data (larger or equal) graph. Returns more than one possible solution. First being the best.
- Assigns cost to each node which reflects difference between the two graphs.
- Cost calculation combines label matching and neighbor proximity.

Use Case

- Consider the graphs in figure 3 and 4, NeMa gives the mapping (1-1, 2-2, 3-3, 4-4) and cost 0.0.
- Now consider the program for Fibonacci series (prog1(main())) and (prog2(main()) & fibonaccii() using iteration.
- With previous code resulting in two graphs corresponding to each method, main() matching costed 0.034 & fibonaccii() costed 1.98.
- However, with the new code which gave a single graph, new graphs of prog1 and prog2 costed only 0.0186.
- We tested some sample codes with small variations to find similar results.
- However, we need to enter the vertex of graph2 with respect to which you want the alignment to be carried out.
- Consider figure 3, when such a situation arises and seed point for graph 2 is 3, NeMa implementation will match vertices 3 and 4 correctly but will fail to match nodes 1 and 2 although they have a corresponding vertex.
- Also, if there are multiple assignment nodes NeMa matches the assig vertex of graph 2 to the first assign vertex of graph 1.

Contributions

- Code for graph comparison (NeMa).
- Library to convert programming language to graphs.
- Converts a class into numerous graphs, one for each method in the class. As an example, code to graph conversion.

Changes

- Wrote code to utilize the library and obtain graph in the required format.
- Modified the code converting programming language to graphs to generate a single graph for every program.
- Consider the code in the above example.

Existing Code

public static void main(String[] args) {
    int i = 8;
    while (i > 0) {
        print(i);
        i--;
    }
}

public void print(int i) {
    System.out.print(i);
}

CONCLUSIONS

- NeMa can match multiple vertices of graph 2 with a single vertex of graph 1. i.e. vertices 1, 2 and 3 of graph 2 can all be matched with vertex 1 of graph 1.
- The current implementation of NeMa does not appear to be consider the topology for matching. The neighborhood cost calculation should be reevaluated.
- We were able to modify the current code to graph conversion program to connect all methods in the same class.
- Therefore, we can infer that NeMa algorithm may be used for matching graphs by making certain modifications such as not matching multiple vertices of one graph with a single vertex of the other graph.

Future Work

- Code to graph conversion can be extended to include multiple classes.
- Currently we are matching only a pair of graphs at a time. The big picture is to able to cluster similar graphs together in a large database of graphs.
- As shown in the figure, graphs can be matched and clustered at different depths.

REFERENCES