Gene Ontology Consistent SubGraph Generation

Rakshith Bhyravabhotla, Rohith Nedunuri and Rudrani Angira

***Abstract*—This project aims to find the number of subgraphs in in a Directed Acyclic Graph(DAG). Intuitively, a consistent subgraph is constructed by selecting a subset of nodes such that the parent property holds and keeping all the edges that start from and end in the nodes from the subgraph. In this project, it is assumed that all the DAGs have a single root node. There are two different algorithms proposed in this project , one for a special case in which DAG is a tree and the other is for general graphs.The algorithm for the tree produces results in linear time complexity. The general algorithm produces results in exponential time for the worst case and linear time for the best case . The space complexity is linear in both the cases.**

***Index Terms*—Graph Algorithm , Tree Algorithm**

I. Tree Algorithm

A. Psuedo code for Tree

Algorithm 1 Tree algorithm

```
1: procedure TREE–ALGORITHM
2:     From the adjacency matrix given , calculate the number of children of each node.
3:     Data: Number of children for each Node.
4:     Result: Number of Consistent Sub Trees.
5:     if node is leaf then return 1
6:         for each child $i \in set$ do
7:             product = product * (1 + (subGraphs(child))
8:         end for
9:     end if
10:    return product
11: end procedure
```

B. Proof of Algorithm

Let $T(r)$ be the subtrees which include the node itself. $\forall$ r in T,

$$T(r) = \prod_{1}^{k} (1 + T(y_i)) \tag{1}$$

where $y_1, y_2, \cdots y_k$ are the children of r. $\forall y_i$ of r, we have a choice of $T(y_i)$ subtrees plus the empty tree itself. Hence,from each child there are

$$1 + T(y_i) \tag{2}$$

consistent subgraphs. If there are k children for the root node 1, total number of consistent subgraphs are

$$T(1) = \prod_{1}^{k} (1 + T(y_i)) \tag{3}$$

C. Analysis of Algorithm

The algorithm is a recursive function which runs for each node in the Tree.So, for a tree with V nodes the loop iterates for V times. So, the time complexity of the algorithm is $O(V)$.

We store the vertices and edges of the tree at any given time and nothing else. So, the space complexity of the graph is $\theta(V+E)$

II. Graph Algorithm

A. Pseudo code for Graph

B. Correctness of Graph Algorithm

Correctness of algorithm can be proved by Loop Invariant. The invariant in the proposed algorithm is the generation of only a consistent graph in every iteration.

1) **Initialization**:The input to the loop beginning at (5) above consists of only consistent sub graphs.
2) **Maintenance**: Inside the loop only the consistent sub graphs are generated through combinations.
3) **Termination**:When the algorithm terminates the list consists of only consistent sub graphs.

C. Complexity of Graph Algorithm

The complexity of the graph algorithm is approaching exponential in our design , as all the possible unique combinations of the graph are being calculated in every iteration.So the calculations in every iteration gradually increases from n^2 , n^4, n^8,and so on .

Prof Predrag Radivojac, SOIC , Indiana University
Mrinmoy Maity, AI, Algorithms

Algorithm 2 Graph algorithm

```
procedure GRAPH–ALGORITHM
    Implement an adjacency list for all the nodes of a graph storing all the child nodes and parent nodes for each node.
    Sort the adjacency list in ascending order of the number of dependents or child nodes for each node.
    for each node $\in sortedList$ do
        push the parent nodes subsequently in a stack till root node is reached
        pop all the elements and store it as a set
    end for
    After step 3 a list of consistent sub graphs is obtained.
    while No new combination is generated do
        Generate the $\binom{n}{2}$ combinations of the elements in list obtained in 3 and perform union operation on the existing list
    end while
end procedure
```

III. Results for the tree data set

The above algorithm has been tried on both the tree data sets given i.e. $tree\_25$ and $tree\_100$ The following are the the output: For $tree\_25$, the number of consistent subtrees are 89960 For $tree\_100$, the number of consistent subtrees are $9,223,372,036,854,775,807$ There are several other test cases on which the algorithm was tested and is proved correct.

IV. Results for the graph data set

The above Algorithm has been tested on the mini example given and several other smaller examples and are proved to be correct. It has been run on $mini\_mfo$ and $mini\_mfo2$ The program is still running without producing an out of memory exception and currently, the number of sub DAGs produced are as follows. $178,373$ on $mini\_mfo2$ and $10,057,178$ on $mini\_mfo$

V. Team Contribution

The team has followed a peer review approach where work done by each person is reviewed by the other two. The work has been divided equally among all the team mates in all the aspects including design, analysis, implementation , testing and documentation. Several meetings and brainstorming sessions were held.

Acknowledgment

This research was supported in part by Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute, and in part by the Indiana METACyt Initiative. The Indiana METACyt Initiative at IU is also supported in part by Lilly Endowment, Inc.