Graph Similarity and Matching in Biological Networks

Surabhi S Nath

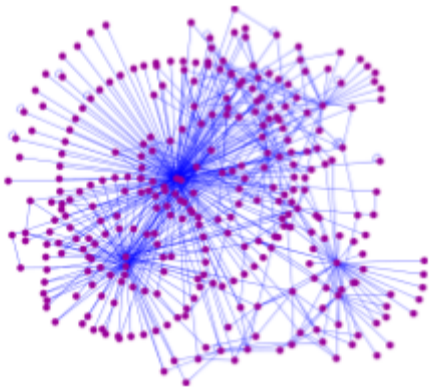
2016271

Utsav Rohilla

2015110

Motivation

Biological networks are one of the most complex systems. They exhibit relationships between between molecules, entities and represent structural properties. Studying these networks is crucial to understand life systems and processes. Graph based data structures can efficiently capture the dynamics of such networks. A variety of static and dynamic graph based methods have been applied for such studies.



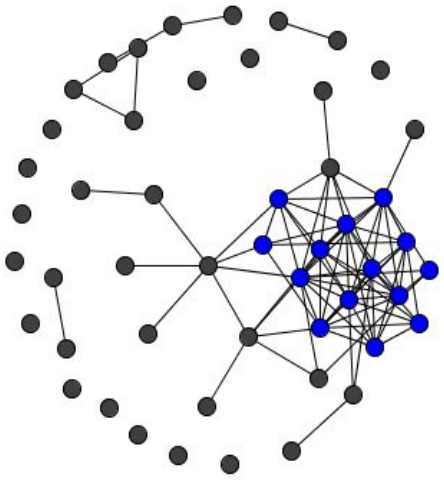
Literature Review

- Graph theory using hypertree and X-tree networks, network profiling combined with knowledge extraction was used to study biological networks PDF, PDF
- For detecting protein complexes, graphs embeddings have been used PDF
- Graph and path matching was applied on biological networks as early as in 2007 PDF
- Graph/subgraph matching and similarity has also been applied for studying relationships
- More recently, dynamic graphs have been used for temporal and structural analysis PDF motif finding PDF and protein complex identification PDF

Our project is inspired by the paper "Top-k Similar Graph Matching Using TraM in Biological Networks" link

Problem Statement

Given a graph database and a query graph, perform approximate graph matching to find similarity scores between the two graphs and substructures from the data graph that match the query graph. Return the best k subgraphs that from the data graph that match the query graph. Approximate graph matching algorithms are an efficient alternative to exact matching. Such a matching algorithm can serve applications in querying in PPI networks, genome data graphs and for searching, comparing, matching and retrieving relevant information embedded in graphs



Algorithm Design

The algorithm mentioned in the paper is nearly followed, with a few modifications. A distance value is obtained based on the conceptual likeness and similarity with respect to topology. A random walk score is calculated for every vertex which represents global structure. An associated reset probability is defined using beta.

$$p_{t+1} = (1 - \beta) * p_t + \beta * p_0$$

where p_t represents a vector whose i^{th} element is the probability of being at node v_i at time step t. The above update equation is repeatedly executed until convergence is achieved. We defined convergence as follows:

If the sum of the elements of the difference matrix $d = p_{t+1} - p_t <$ threshold, **converged**

Next, a beta signature is evaluated by obtaining the above random walk score for a range of beta values. Here, if we consider a very large number of beta values (approx. n), the complexity of the problem increases by a factor of n. To avoid this, we performed weighting of beta values. This results in a vector representation for each vertex.

From this, a beta similarity is obtained between 2 graphs. This score is calculated by taking

1 - root of sum of square differences between the vector representations of 2 vertices (may or may not belong to the same graph)

This score is then used to perform graph matching. Candidate subgraphs are generated and pruned. These are matched with the query graph and k related graphs are obtained. The algorithm is given below:

Algorithm 2: GraphMatch

Input: Data graph $D = (V_d, E_d)$ and query graph $Q = (V_q, E_q)$. Thresholds k, μ_v, μ_s and λ.
Output: Top-k matches of Q.

```
1 Initialize priority queue PQ as empty;
2 Calculate β-signature β(Q) for Q;
3 Compute radius r of Q;
4 for ∀v_d(v_d ∈ V_d) do
5     Compute δ(r, v_d);
6     if |Filter(δ(r, v_d), Q, μ_v, μ_s)| > |V_q| then
7         Top-k Match(Q, β(Q), D, β(χ(δ(r, v_d))), λ,
          PQ);
8 return All top k graphs g ∈ PQ ;
```

The **neighbourhood** of a vertex is defined as all the vertices is at most x distance (in this case, radius r of Query graph) from that vertex. The graph obtained by including the vertices in the neighbourhood along with corresponding edges is an induced subgraph for that vertex.

From the above algorithm we see that an **induced subgraph** is obtained from every node of the data graph and matched with the query graph. Since the data graph may be large, this could be algorithmically costly.

Algorithmic Complexity

Let *N* be the number of nodes in the graph (vertex set including both query graph and data graph) and *E* be the number of edges in the graph, K be the number of beta

values, CT be the number of iterations we do the matrix multiplication for the random walk probability calculation.

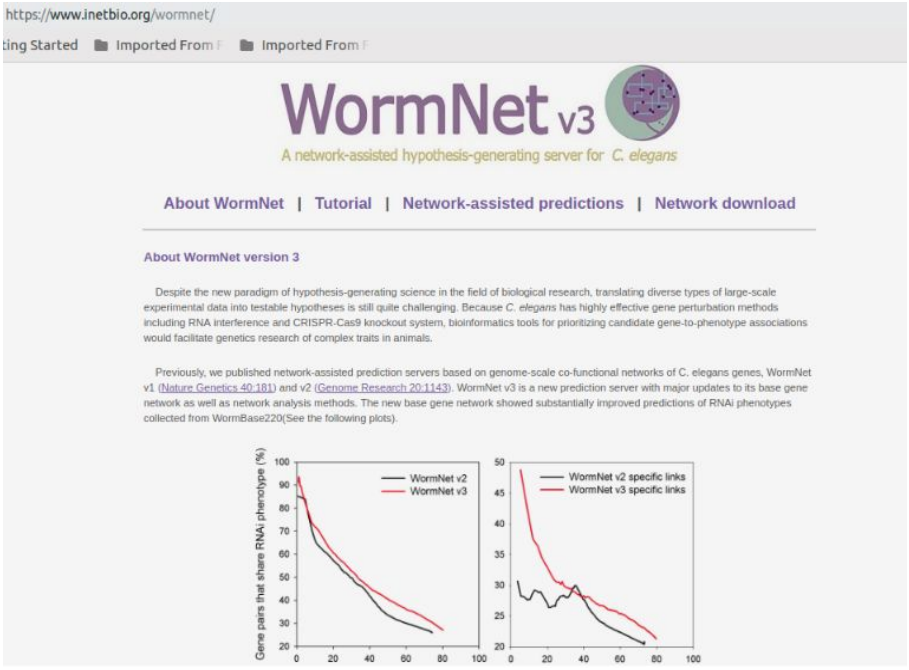
An upper bound of complexity of the code is given by **O(CT * N * N * K * log(N)).**

Complexity in Data Acquisition

Since the algorithmic complexity for the task is high - as seen above, the size of the input graphs needs to be selected carefully. However, it runs faster in practice because the graph is sparse and the complexity also includes the size of the induced subgraph that is matched with the data graph which is bounded by N but the actual value is much lesser.

A data graph size of around 5000 nodes takes nearly 3-5 minutes for evaluation.

Graphs for gene expression and protein-protein interaction in the form of egde list were obtained for worm (C.elegans), fly (D.melanogaster) and humans (H.sapiens) from WormNet (https://www.inetbio.org/wormnet/downloadnetwork.php). We mainly used graphs for Fly and Worm as they were smaller as compared to Human.



Findings

- While calculating the random walk probability for each node, the matrix values converges to some values in the matrix multiplication step.
- For a larger beta value, convergence is attained faster
- We can tune the parameters to get different results for the same data graph and query graph

Conclusions

We do an approximate matching between data graph and query graph for fly-worm and fly-fly:

- On running our code on the dataset of **fly and worm**, we get similar subgraphs of size 7-8 with sufficient similarity scores and parameters
- On running our code on the dataset of **fly and fly**, we get similar subgraphs of size 30-40 with sufficient similarity scores and parameters

These results reinforce that similarly between same species is more than similarly across species.

Code Explanation

The code consists of the following major functions explained above:

v2d: vector <vector <double> >

- `v2d randomWalk(v2d &dataSetGraph, double beta)`
- `v2d betaSignature(v2d dataSetGraph)`
- `v2d betaSimilarity(v2d sigA, v2d sigB)`
- `int getRadius(v2d &g)`
- `void getNeighbour(int u, v2d &g, int d, int D, set <int> &neighbours, bool vis[])`
- `set <int> deltaNeighbourHood(v2d &g, int v, int r)`
- `v2d inducedSubgraph(v2d &g, int v, int r)`
- `void filter(set <int> &Vs, v2d queryGraph, double muv, double mus)`

- `void kMatch(v2d queryGraph, v2d candidateGraph, double lambda, priority_queue <pair <int, set <int> > > &pq)`
- `vector <set <int> > graphMatch(v2d dataSetGraph, v2d queryGraph, double k, double muv, double mus, double lambda)`

Code running instructions

in.txt: Weighted edge list of the data set graph followed by a line of '*' and then edge list of the query graph.

out.txt: Output file created

Run using:
g++ -std=c++11 code.cpp
./a.out <in.txt >out.txt

Reproducibility

The code can be executed for any query and data graph and data can easily be acquired from the above mentioned source hence the work is reproducible. Code available at GITHUB.