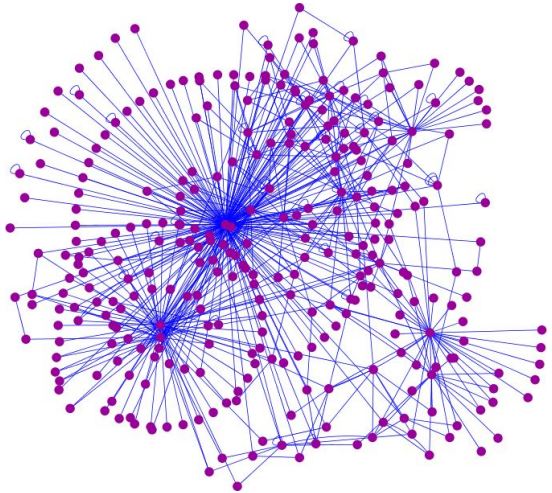
Application of Graph Algorithms for Studying Biological Networks



Surabhi S Nath
Utsav Rohilla

Introduction

- Biological networks are one of most complex systems
- They exhibit relationships between between molecules, entities and show structural properties
- Studying them is crucial to understand life
- 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 be 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

Problem Statement

- Similarities in graphs helps in studying gene-gene or protein-protein interactions
- Such networks present challenges for network analysis pertaining to their large network size which demand efficient algorithm
- We aim to perform approximate subgraph matching in biological domain for labelled and unlabelled graphs and also extend to dynamic graphs
- For similar graphs, the degree of similarity between two graphs is a key factor

Methodology

- A distance value is obtained based on the conceptual likeness and similarity wrt topology
- **Random Walk for Global Properties**
  - Graph similarity can be expressed as a function of result of random walks
  - $p^{t+1} = (1 - \beta) * p^t + \beta * p^0$, where p^t represents a vector whose i^{th} element is the probability of being at node v_i at time step t

Algorithm 1: Random Walk

Input: Graph $G = (V_g, E_g)$ and Restart Probability β.
Output: Random Walk Score $P_s(V_g)$.

1 Let $r_s(V_g)$ be the restart vector with all entries having value $\frac{1}{|V_g|}$;
2 Let A be the column normalized adjacency matrix defined by E;
3 Initialize $P_s(V_g) = r_s(V_g)$;
4 **while** $P_s(V_g)$ *has not converged* **do**
5 $\quad P_s(V_g) = (1-\beta) * A * P_s(V_g) + \beta * r_s(V_g))$;

- **Structural Similarity of Nodes**
  - It can be used to compare topological orientation and relative importance of the nodes.
  - Given nodes u_1, u_2 and u_3, it compares whether u_1 is more similar to u_2 or u_3
  - Evaluate Beta similarity as

$$\hat{\beta}(v_1, v_2) = 1 - \sqrt{\sum_{i=1}^{k} (a_i - b_i)^2},$$

where $\vec{\beta}(v_1) = (a_1, a_2, \ldots, a_k) \in \beta(G_1)$ *and* $\vec{\beta}(v_2) = (b_1, b_2, \ldots, b_k) \in \beta(G_2)$.

- Get a combined graph similarity, extend to find top-k graph matching using the TraM algorithm

- **TraM Algorithm**
  - Generate candidate subgraphs
  - Prune candidate subgraphs
  - Perform matching with query graph
  - Obtain k related graphs and derive conclusions

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 ;
```