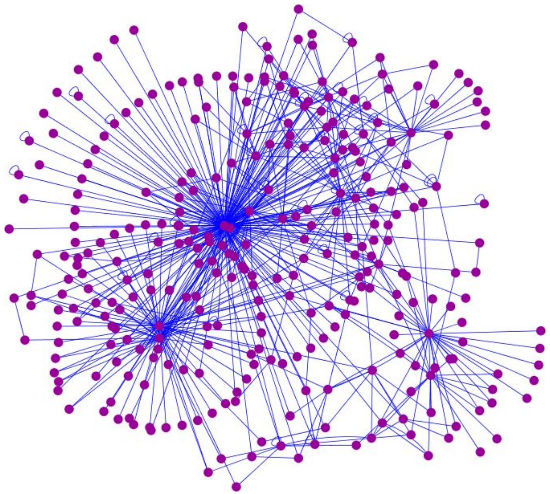
Top-k Similar Graph Matching in Biological Networks



Surabhi S Nath
Utsav Rohilla

Introduction

- Biological networks are one of most complex systems
- Relationships between between molecules, entities, structural properties
- Crucial to understand life
- Graph based data structures efficient in capturing dynamics of networks
- A variety of static, dynamic graph based methods have been applied

Problem Statement

- Graph matching important to retrieve information embedded in graphs
- Efficient alternative to exact graph matching - approximate graph matching
- Tram algorithm for large graph networks
- Given a graph database and multiple query graphs, find similar substructures from the data graph that match the query graph
- Applications include querying in protein networks - searching, comparing

TraM Algorithm

- Computes similarity between two labelled graphs
- Algorithm computes the topmost k similar graphs based on different similarity criteria. This algorithm's strength lies in the fact that multiple domain dependent similarity scores can be incorporated into an overall similarity calculation of nodes
- Conceptual likeness of two sets of nodes is captured using a domain-dependent similarity function called the σ-similarity function
- Random walk captures structural information of the graph and similarity function is called the β-similarity function

TraM Algorithm

- Before we get into the main algorithm, we need to get familiar with some definitions and helping algorithms

Algorithm 1: Random Walk

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

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

TraM Algorithm

Definition 2 (β-signature). *Let $G = (V_g, E_g)$ be a graph and $\mathbb{B}$ be a set of β values. Random walk score of a node is a function of the form $\omega : V_g \times \mathbb{B} \to [0, 1]$. An n-dimensional vector $(\omega(v, \beta_1), \omega(v, \beta_2), \ldots, \omega(v, \beta_n))$ is called a β-signature of node $v \in V_g$, denoted $\vec{\beta}(v)$ and the set $\beta(G) = \bigcup_{v \in V} \vec{\beta}(v)$ is called the β-signature of G.*

Definition 4 (β-similarity). *Let $G_1 = (V_{g_1}, E_{g_1})$ and $G_2 = (V_{g_2}, E_{g_2})$ be two graphs and $\beta(G_1)$ and $\beta(G_2)$ be their β-signatures. For any $v_1 \in V_{g_1}$ and $v_2 \in V_{g_2}$, their structural or β-similarity, denoted $\hat{\beta}(v_1, v_2)$, is defined by*

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

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

TraM Algorithm

Definition 5 (Graph similarity). *Let* $G_1 = (V_{g_1}, E_{g_1})$ *and* $G_2 = (V_{g_2}, E_{g_2})$ *be two graphs and* ϕ *be a mapping function. Then, the similarity* γ *between two graphs* G_1 *and* G_2 *under a mapping function* ϕ *is*

$$\gamma(G_1, G_2) = \sum_{\forall v_1, v_2 (v_1 \in V_{g_1}, \phi(v_2) \in V_{g_2})} \sigma(v_1, \phi(v_2)) \times \hat{\beta}(v_1, \phi(v_2)). \tag{2}$$

TraM Algorithm

Definition 6 (δ-neighborhood of nodes). *Let Q be a query graph, and r be its radius.*[1] *Let $D = (V_d, E_d)$ be any graph, and $v \xrightarrow{j} u$ represent j-hop reachability from node v to u. Then, δ-neighborhood of v is the set $\{u | v, u \in V_d \wedge v \xrightarrow{j} u \wedge j \leq r\}$ (including zero hop reachability, i.e., v itself), denoted $\delta(r, v)$.*

Definition 7 (Induced subgraphs). *Let $G = (V_g, E_g)$ be any graph, and $N \subseteq V_g$ be an arbitrary set of nodes. Then, $G' = (N, E')$ is called an induced subgraph, denoted $\chi(N) = G' = (N, E')$, such that whenever $v, v' \in N$ and $e = (v, v') \in E_g$, e is also in E', and nothing else is in E', i.e., G' is a subgraph of G, denoted $G' \sqsubseteq G$.*

TraM Algorithm

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.

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

TraM Algorithm

Algorithm 4: Filter

Input: Set of nodes $\delta(r, v_d)$ and Graph $Q = (V_q, E_q)$.
Thresholds μ_v and μ_s.

Output: Pruned $\delta(r, v_d)$.

if $|\delta(r, v_d)| > |V_q|$ **then**
    for *every* $v_n \in \delta(r, v_d)$ **do**
        Compute candidate subgraph as $\chi(\delta(r, v_d))$;
        Compute β-signature $\beta(\chi(\delta(r, v_d)))$ for $\chi(\delta(r, v_d))$;
        if $\forall v_q (v_q \in V_q (\max\{\sigma(v_n, v_q)\} < \mu_v$ *or* $\max\{\hat{\beta}(v_n, v_q)\} < \mu_s))$ **then**
            remove v_N from $\delta(r, v_d)$;

return $\delta(r, v_d)$;

TraM Algorithm

Algorithm 3: Top-k Match

Input: Query graph $Q = (V_q, E_q)$, $\beta(Q)$, candidate graph $C = (V_c, E_c)$, $\beta(C)$, threshold λ. Queue PQ

Output: Updated priority queue PQ.

for $i = 1$ *to Number of Nodes in time-stamp 1* **do**
- initialize $S^{(i,1)} = \emptyset$ and $sim(i, S^{(i,1)})^{(1)} = 0$;

for $t = 2$ *to* $2 \times |V_q|$ **do**
- **for** $r = 1$ *to Number of Nodes in time-stamp* t **do**
  - **if** t *is Even* **then**
    - **for** $p = 1$ *to Number of Nodes in time-stamp* $t-1$ **do**
      - **if** $(r \notin S^{(p,t-1)})$ *and* $((\exists i(i \in S^{(p,t-1)}$ *and* $r \xrightarrow{1} i$ *and* $i \in V_c)))$ *and* $(sim(p, S^{(p,t-1)})^{(t-1)} + \sigma(p,r) \times \hat{\beta}(p,r)) \geq MAX)$ **then**
        - $MAX = sim(p, S^{(p,t-1)})^{(t-1)} + \sigma(p,r) \times \hat{\beta}(p,r)$;
        - $k = p$;
        - $newMAX = true$;
    - **if** $newMAX = true$ **then**
      - $S^{(r,t)} = S^{(k,t-1)} \cup \{r\}$;
      - $sim(r, S^{(r,t)})^{(t)} = MAX$;
  - **if** t *is Odd or* $newMAX = false$ **then**
    - $S^{(r,t)} = S^{(r,t-1)}$;
    - $sim(r, S^{(r,t)})^{(t)} = sim(r, S^{(r,t-1)})^{(t-1)}$;
  - **if** $t = 2 \times |V_q|$ *and last* r **then**
    - $\gamma(Q,C) = sim(r, S^{(r,t)})^{(t)}$;

if $\gamma(Q,C) \geq \lambda$ **then**
- Add $\langle C, \gamma(Q,C) \rangle$ to queue PQ;

return PQ