Comp790-166: Computational Biology

Lecture 17

March 28, 2023

Today

- Multimodal integration problems on graphs
  - Reconciling multiple graphs, or multiple relational definitions to learn common embedding for each node.
  - Graph Alignment for registering nodes to each other across multiple graphs

Review Questions

1. What are some of the possible limitations of concatenating features from multiple modalities?
2. How would we find a subspace representation for an individual modality (generally at a high level), (for example, a given $\mathbf{U}^m$?

Writing Down the Objective Function

The goal is to specify a $\mathbf{U}^m$ for each modality. The optimal graph embedding in k dimensions can written as,

$$\min_{\mathbf{U}^m \in \mathbb{R}^{N \times k}} \operatorname{tr}\left(\mathbf{U}^{m'} \mathbf{L}^m \mathbf{U}^m\right), \quad \text{s.t. } \mathbf{U}^{m\prime} \mathbf{U}^m = I$$

- It turns out the solution is the first k eigenvectors of the Graph Laplacian $\mathbf{L}^m$ by the Rayleigh–Ritz theorem

[1]

[1] Note that the $U^{m'}$ refers to the transpose of $\mathbf{U}^m$

Example from STRING

Using the STRING database, you can extract PPIs according to multiple relational definitions.

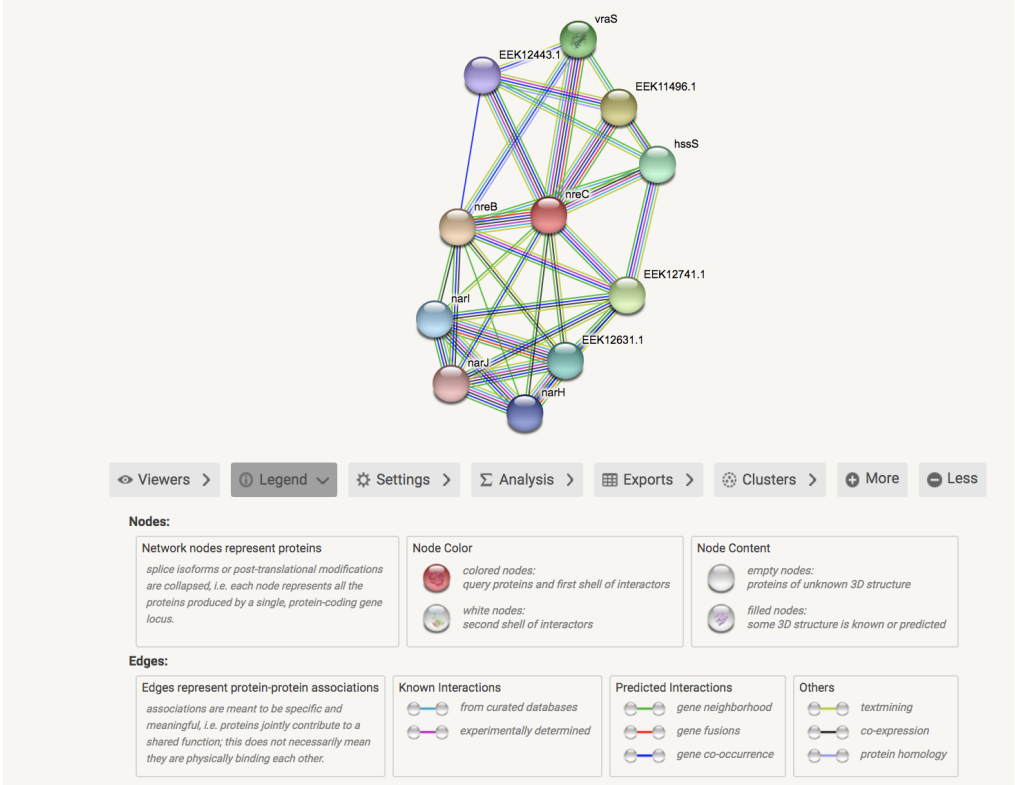


Figure: `https://string-db.org/`

Defining Consensus d-dimensional vectors for nodes

Given multiple relational definitions (e.g. multiple graphs) between a common set of nodes (features), define a consensus d-dimensional embedding vector for each node that aligns well with each individual graph (e.g. distinct relational definitions).

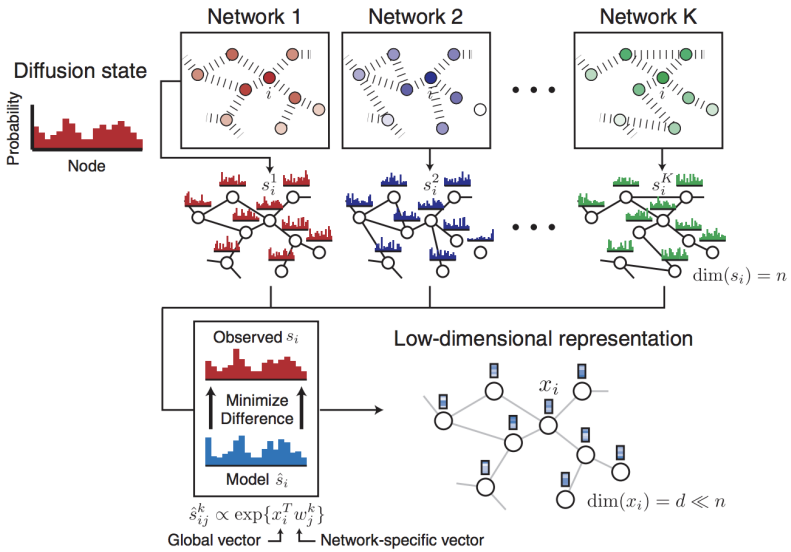


Figure: from Cho *et al.* Cell Systems. Each graph is representing a different relational definition between node (features).

Random Walk with Restart

- RWR is a way to account for both local and global ‘walk‘ information in the graph by giving your walker the chance to restart

But first, let's re-define the transition probability that a walker goes from node j to node i as,

$$B_{ij} = \frac{A_{ij}}{\sum_{i'} A_{i'j}}$$

RWR Formally Written

Given the transition matrix, B, the RWR from a node i is defined as,

$$s_i^{t+1} = (1 - p_r)Bs_i^t + p_r e_i$$

- p_r is the probability of restart
- e_i is an n-dimensional vector with $e_i(i) = 1$ and $e_i(j) = 0$ for $j \neq i$
- s_i^t is the vector of probabilities of each node being visited after t steps in the random walk, starting from node i

Clarifying What is Happening Here

$$s_i^{t+1} = (1 - p_r)Bs_i^t + p_r e_i$$

- The first term corresponds to following a random edge connected to the current node
- The second term corresponds to restarting from node i.
- At some point, this reaches a stationary distribution, s_i^{∞}, or fixed point
- When the diffusion states between two nodes are close, this implies they have similar positions in the graph with respect to other nodes.

Quantifying Topological Overlap Between a Node Pair

Each node is given two vector representations, $\mathbf{w}_i, \mathbf{x}_i \in \mathbb{R}^d$

- Let $\mathbf{w}_i$ refer to the context feature of a node (e.g. per relational definition)
- Let $\mathbf{x}_i$ refer to the node feature of node i (e.g. overall)

Define a new similarity measure between nodes i and j as,

$$\hat{s_{ij}} = \frac{\exp\{x_i^T w_j\}}{\sum_{j'} \exp\{x_i^T w_{j'}\}}$$

Unpacking

$$\hat{s_{ij}} = \frac{\exp\{x_i^T w_j\}}{\sum_{j'} \exp\{x_i^T w_{j'}\}}$$

- If $\mathbf{x}_i$ and $\mathbf{w}_j$ are close in direction and hence have a large inner product, then node j should be frequently visited in the random walk starting from node i.

Recap of what is happening

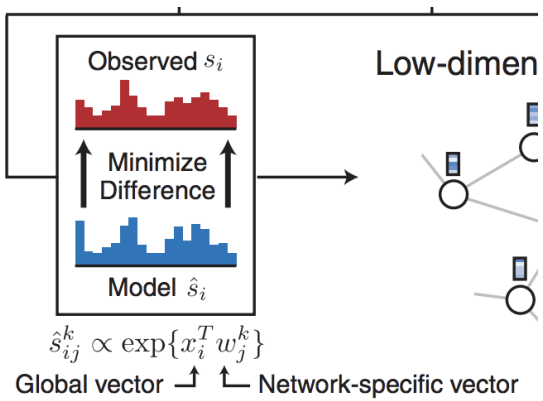


Figure: from Fig. 1. Given observed diffusion states from RWR, we should be able to find a global vector (**x**) and view-specific vector (**w**), such that a function of **x** and **w** gives a good diffusion state approximation.

Writing Out an Objective Function for Embeddings

To find optimal d-dimensional representations for each node, formulate an optimization problem that minimizes the notation between s (RWR diffusion state) and $\hat{s}$ ('approximation') as,

$$\underset{w,x}{\text{minimize}} C(s, \widehat{s}) = \frac{1}{n} \sum_{i=1}^{n} D_{KL}\left(s_i \| \widehat{s_i}\right)$$

Written out, given our definition of $\hat{s}$ gives the following (with $H(\cdot)$ denoting entropy),

$$C(s, \widehat{s}) = \frac{1}{n} \sum_{i=1}^{n} \left[-H\left(s_i\right) - \sum_{j=1}^{n} s_{ij} \left(x_i^T w_j - \log \left(\sum_{j'=1}^{n} \exp\left\{ x_i^T w_{j'} \right\} \right) \right) \right]$$

Integrating Heterogeneous Networks

You can do these RWRs on each individual network. At the same time, you can let x be fixed across all relational definitions.
Similar to what we have seen, yet adapted for modality k, we can write,

$$\widehat{s}_{ij}^{k} := \frac{\exp\left\{x_i^T w_j^k\right\}}{\sum_{j'} \exp\left\{x_i^T w_{j'}^k\right\}}$$

Writing the Objective Function Across All Modalities

Now, the objective function can be rewritten to take into account the recently-computed $\widehat{s}_{ij}^{k}s$, and sums over all modalities as,

$$\underset{w,x}{\text{minimize}} C(s,\widehat{s}) = \frac{1}{n}\sum_{k=1}^{k}\sum_{i=1}^{n} D_{KL}\left(s_i^k \| \hat{s}_i^k\right)$$

Implementation (the slow way)

To find the optimal ws and xs for each node, you could compute gradients, which turn out to be,

$$\nabla_{w_i^k} C(s, \widehat{s}) = \frac{1}{n} \sum_{j=1}^{n} \left(\widehat{s}_{ji}^k - s_{ji}^k \right) x_j$$

and

$$\nabla_{x_i} C(s, \widehat{s}) = \frac{1}{n} \sum_{k=1}^{K} \sum_{j=1}^{n} \left(\widehat{s}_{ij}^k - s_{ij}^k \right) w_j^k$$

An SVD Formulation: Setup

- Let S^k be the $N \times N$ diffusion state matrix for network k
- Also, let s_i^k be the ith column of this matrix, S^k

These matrices can be concatenated to form an $NK \times N$ matrix, S

Remember Truncated SVD?

The authors used truncated SVD as an alternative to estimating the w_i^ks and x_is as,

$$S = U\Sigma V$$

(Remember this implies that we will have some 0s on the diagonal (e.g. zeroed out singular values) of Σ)

- $\{w_i^k\} \rightarrow \Sigma^{1/2} U^T \rightarrow (d \times d) \times (d \times NK)$
- $\{x_i\} \rightarrow \Sigma^{1/2} V \rightarrow (d \times d) \times (d \times N)$

Using the Learned $\mathbf{x}_i$s as Feature Vectors

- After Mashup each node, i has an embedding, $\mathbf{x}_i$.
- Each protein has a known function, which we can try to predict.

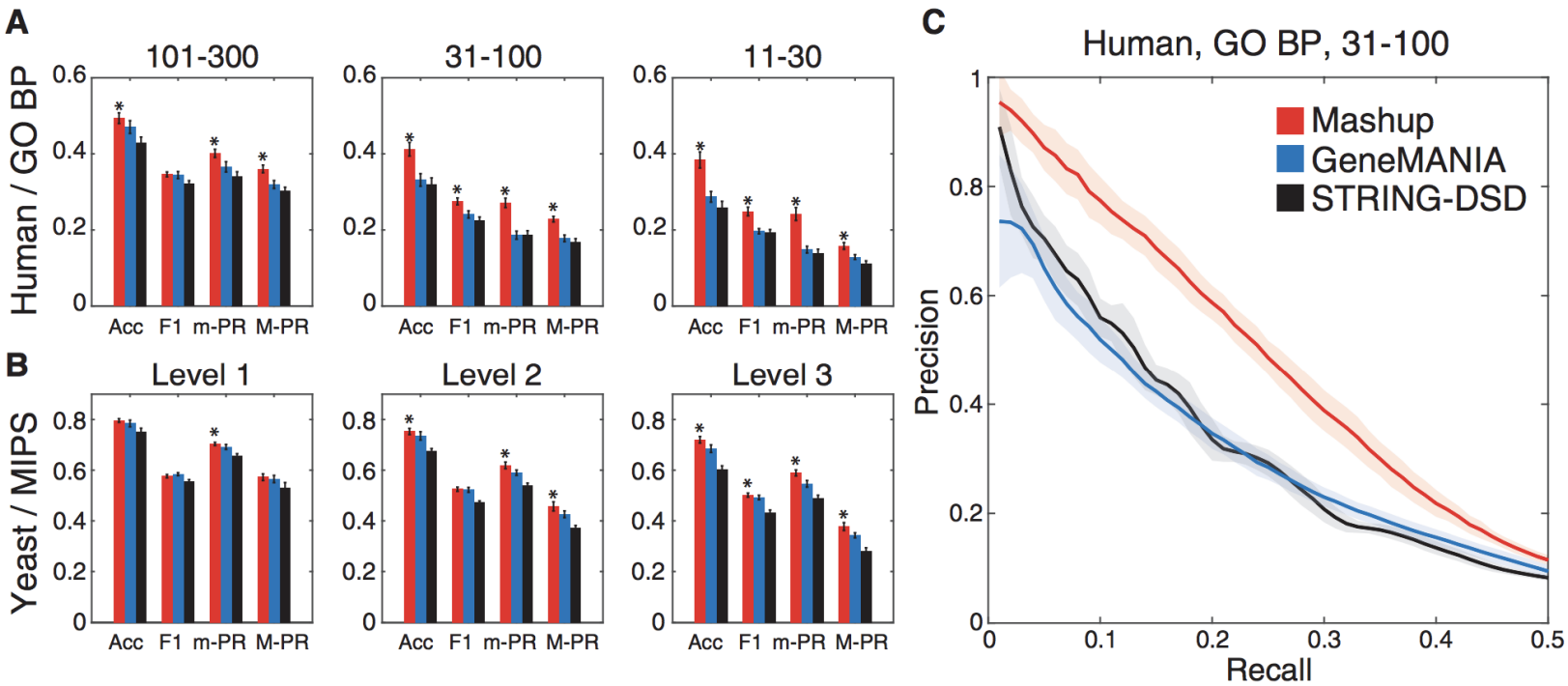


Figure: from Fig. 2. Performance is evaluated for multiple levels of annotation.

Similarly, Combining All Networks Leads to Better Protein Function Prediction

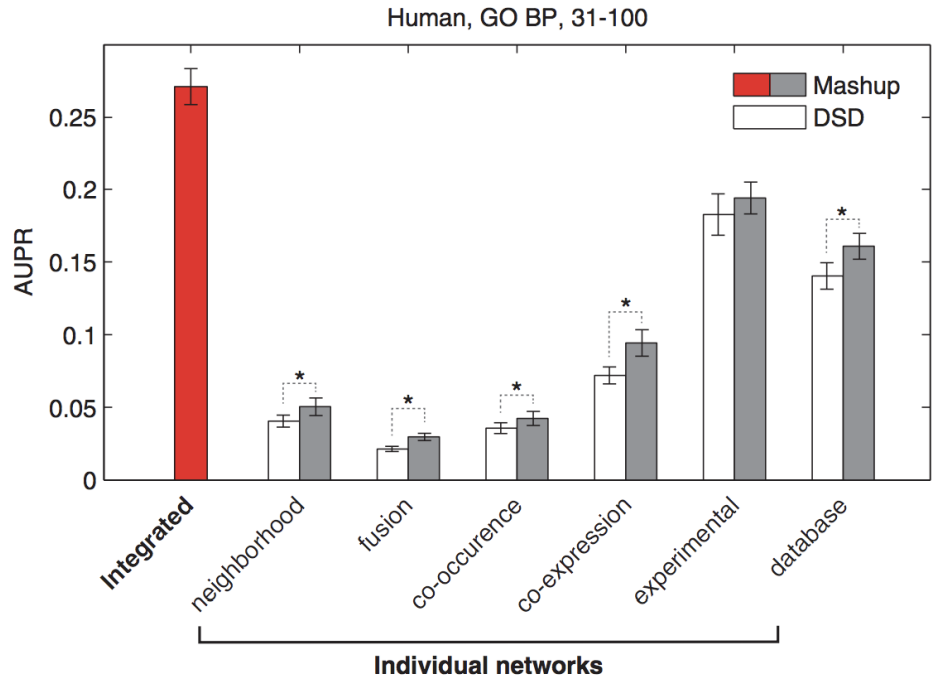


Figure: from Fig. 3. DSD is some single-graph diffusion method. Note that experimental is also a top performer.

Intuition about Parameters

The main parameters of interest is the restart probability, and the number of dimensions to keep.

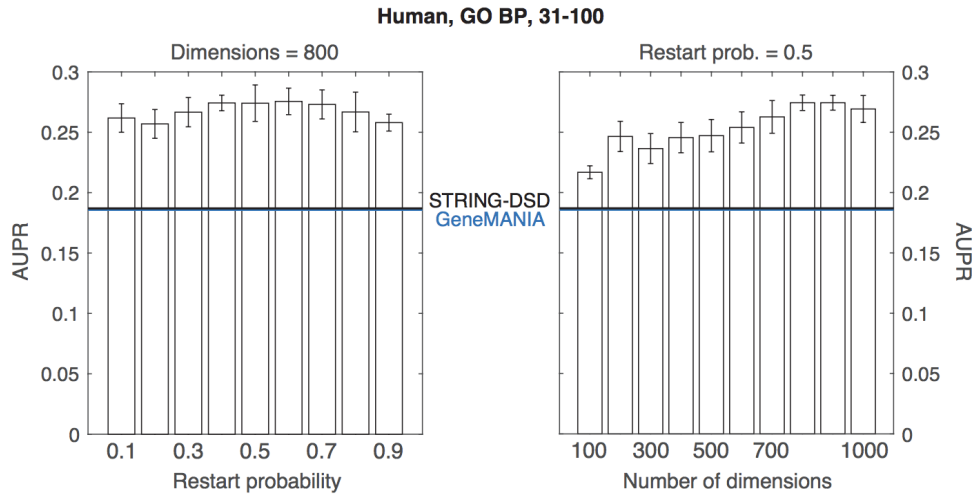


Figure: from Supp Fig. 4

*Btw, I recommend choosing your y-axis so that it is useful when making such a plot.

Mashup is More Robust to Noise in the Network

Here, noise was simulated by removing a subset of edges from the original graph.

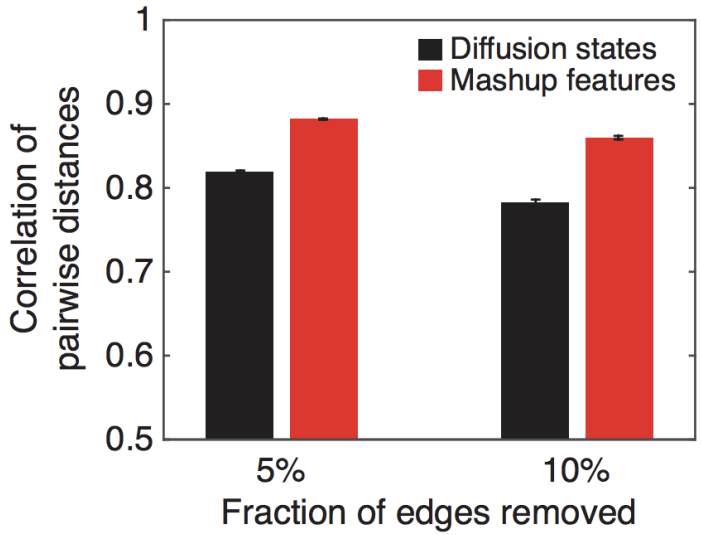


Figure: from Supp. Fig. 7. Edges were removed from the BioGrid physical interaction network. Similarities between nodes could be calculated based on diffusion state or mashup.

Recap and Switching Gears

- Context-specific relational information and general relational information are used to construct a joint representation for each node, based on accurate modeling of diffusion states.
- Vector representation can be used to better predict node functions.

Graph Alignment: What happens now if we have multiple graphs, but we don't know how to register the nodes, in the sense of which nodes align based both on connectivity patterns and node-level features?

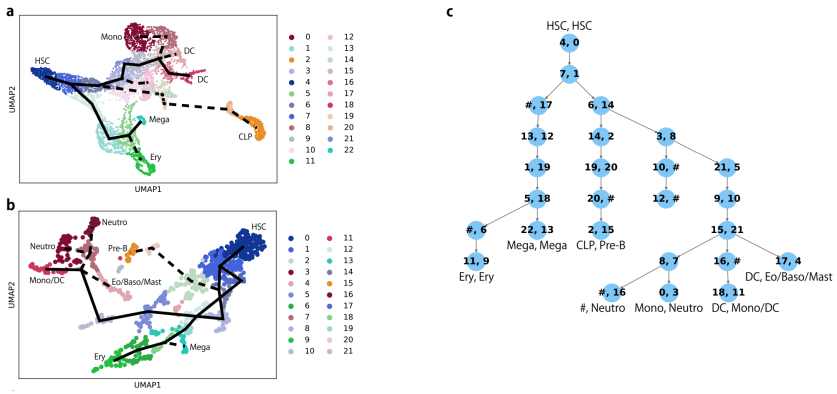


Figure: from Sugihara *et al.* Nature Communications, 2022

Problem

Here is a formal definition of the graph matching problem.

PROBLEM 1. *Given two graphs G_1 and G_2 with node-sets $\mathcal{V}_1$ and $\mathcal{V}_2$ and possibly node attributes $\mathcal{A}_1$ and $\mathcal{A}_2$ resp., devise an efficient* ***network alignment method*** *that aligns nodes by learning* ***directly comparable*** *node representations $\mathbf{Y}_1$ and $\mathbf{Y}_2$, from which a node mapping $\phi : \mathcal{V}_1 \rightarrow \mathcal{V}_2$ between the networks can be inferred.*

Figure: from from Heimann *et al.* CIKM 2018.

Node Attributes vs Connectivity

- Generally, attributes are measurements or features that we have about the nodes, unrelated to connectivity

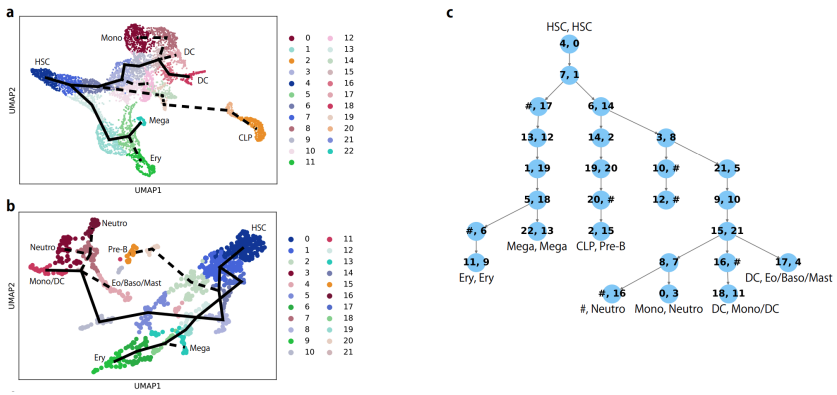


Figure: So, in this case, **connectivity** or edges are defined between cell-populations close in the psuedotermporal ordering. **Attributes** are features about the cells, such as gene or protein measurements.

Overview of Regal Method

Regal → Representation Based Graph Alignment.

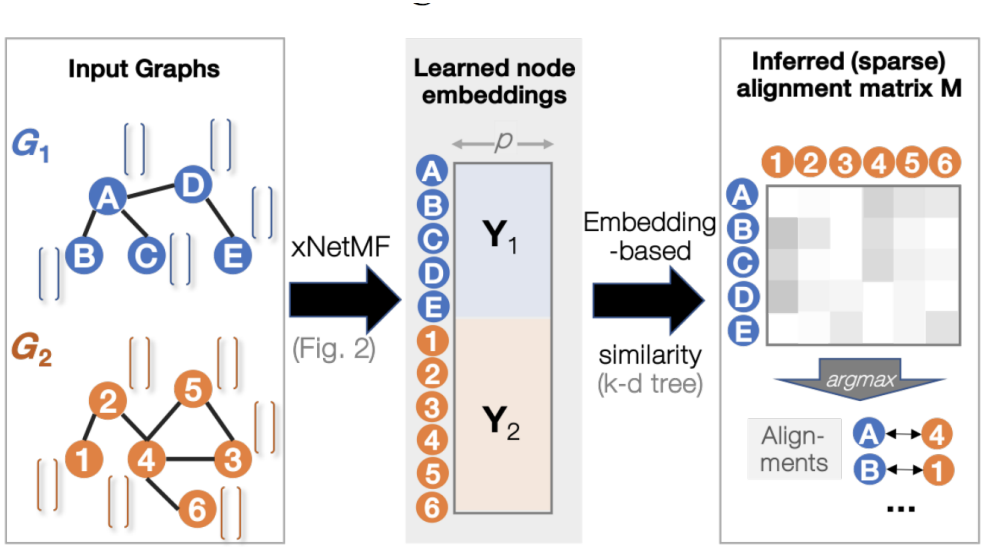


Figure: from from Heimann *et al.* CIKM 2018. Similar to Node2vec, the authors want to find a representation for each node

From our earlier example, we would get a mapping of cell-populations from one dataset to the other.

Defining Node Identity

- Previously (e.g. node2vec), we saw node representations were defined in terms of their neighbors through random walks.
- In this setting, of quantifying relationships between nodes between graphs, we cannot walk because we have two different graphs.
- The solution is to instead focus on nodes with similar structural roles (e.g. degree, degree of neighbors, etc).

Calculating Node Similarity Within and Between Graphs

Define distance between nodes u and v in terms of structure ($\mathbf{d}$), or attributes ($\mathbf{f}$) as,

$$\text{sim}(u, v) = \exp\left[-\gamma_s \cdot \|\mathbf{d}_u - \mathbf{d}_v\|_2^2 - \gamma_a \cdot \text{dist}\left(\mathbf{f}_u, \mathbf{f}_v\right)\right]$$

Each $\mathbf{d}_u$ is defined as,

$$\mathbf{d}_u = \sum_{k=1}^{K} \delta^{k-1} \mathbf{d}_u^k$$

Here, δ is a discount factor for greater hop distances.

Expensive Formulation

Given a factorization approach, write the $n \times n$ similarity matrix, $\mathbf{S}$ as,

$$\mathbf{S} \approx \mathbf{Y}\mathbf{Z}^T$$

Here, $\mathbf{Y}$ gives the node-to-embedding matrix (which is what we want).
Intuitively, we want the following to be as close as possible to 0,

$$||\mathbf{S} - \mathbf{Y}\mathbf{Z}^T||$$

($\mathbf{Z}$ and $\mathbf{Y}$ are both $n \times p$ matrices)

An Approximation with Landmarks

- The punchline is that $\mathbf{S}$ will be approximated with a low-rank matrix, $\tilde{\mathbf{S}}$, which is never explicitly computed!
- The solution is to choose $p << n$ 'landmark' nodes, chosen across both graphs (G_1, G_2).
- Compute similarity between each node and each landmark. This produces an $n \times p$ matrix, $\mathbf{C}$.
- Further, the $p \times p$ landmark-to-landmark submatrix can also be extracted ($\mathbf{W}$).

The Low-Rank Matrix, $\tilde{\mathbf{S}}$

Finally, the low-rank matrix $\tilde{\mathbf{S}}$ is given as,

$$\tilde{\mathbf{S}} = \mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^{T}$$

Here $\mathbf{W}^{\dagger}$ is computed as the pseudoinverse of $\mathbf{W}$

- The problem is that $\tilde{\mathbf{S}}$ is still an $n \times n$ matrix, and we still need to find the embedding matrix, $\mathbf{Y}$

Approximation for the Embedding Matrix, $\mathbf{Y}$

Theorem: Given graphs, $\mathcal{G}_1(\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2(\mathcal{V}_2, \mathcal{E}_2)$ with an $n \times n$ joint combined structural and attribute-based similarity matrix $\mathbf{S} \approx \mathbf{Y}\mathbf{Z}^T$, its node embedding matrix $\mathbf{Y}$ can be approximated as,

$$\tilde{Y} = CU\Sigma^{-1/2}. \tag{1}$$

Here, $\mathbf{C}$ is the $n \times p$ matrix of similarities between the n nodes and the p chosen landmarks and $\mathbf{W}^\dagger = \mathbf{U}\Sigma\mathbf{V}^T$ is the full rank SVD of the small $p \times p$ landmark similarity matrix matrix, $\mathbf{W}$

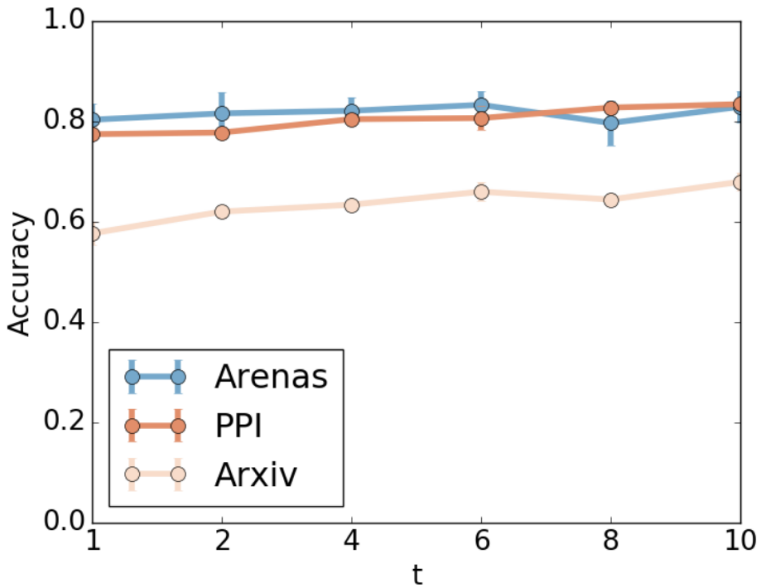
So to Summarize the Entire xNetMF

Algorithm 2 xNetMF $(G_1, G_2, p, K, \gamma_s, \gamma_a)$

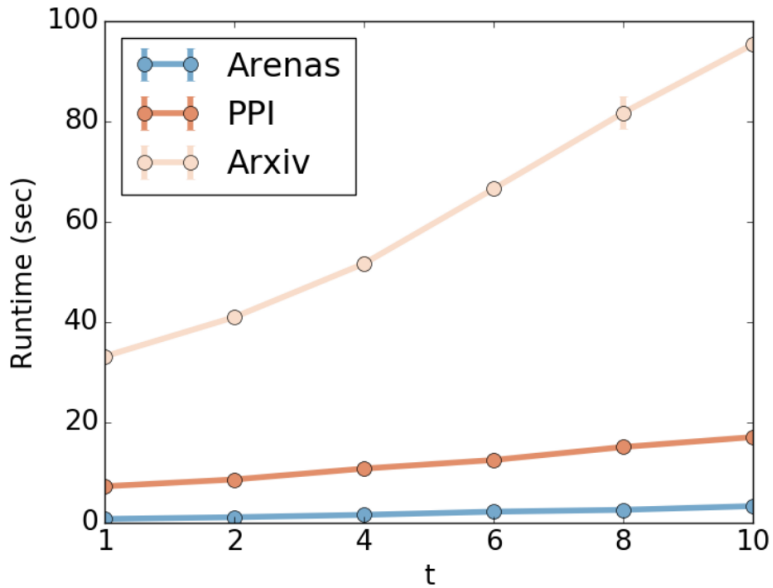
========== STEP 1. Node Identity Extraction ==========
for node u in $\mathcal{V}_1 \cup \mathcal{V}_2$ **do**
  for hop k up to K **do** ▷ counts of node degrees of k-hop neighbors of u
    $\mathbf{d}_u^k$ = CountDegreeDistributions($\mathcal{R}_u^k$) ▷ $1 \leq K \leq$ graph diameter
  end for
  $\mathbf{d}_u = \sum_{k=1}^{K} \delta^{k-1} \mathbf{d}_u^k$ ▷ discount factor $\delta \in (0, 1]$
end for
========== STEP 2. Efficient Similarity-based Representation ==========
========== STEP 2a. Reduced $n \times p$ Similarity Computation ==========
$\mathcal{L}$ = ChooseLandmarks(G_1, G_2,p) ▷ choose p nodes from G_1, G_2
for node u in $\mathcal{V}$ **do**
  for node v in $\mathcal{L}$ **do**
    $c_{uv} = e^{-\gamma_s \cdot ||\mathbf{d}_u - \mathbf{d}_v||_2^2 - \gamma_a \cdot \text{dist}(\mathbf{f}_u, \mathbf{f}_v)}$
  end for
end for ▷ Used in low-rank approx. of similarity graph (not constructed)
========== STEP 2b. From Similarity to Representation ==========
$\mathbf{W} = \mathbf{C}[\mathcal{L}, \mathcal{L}]$ ▷ Rows of $\mathbf{C}$ corresponding to landmark nodes
$[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = \text{SVD}(\mathbf{W}^\dagger)$
$\tilde{\mathbf{Y}} = \mathbf{C}\mathbf{U}\mathbf{\Sigma}^{-\frac{1}{2}}$ ▷ **Embedding**: *implicit* factorization of similarity graph
$\tilde{\mathbf{Y}} = Normalize(\tilde{\mathbf{Y}})$ ▷ **Postprocessing**: make embeddings have magnitude 1
$\tilde{\mathbf{Y}}_1, \tilde{\mathbf{Y}}_2 = \text{Split}(\tilde{\mathbf{Y}})$ ▷ **Separate** representations for nodes in G_1, G_2
return $\tilde{\mathbf{Y}}_1, \tilde{\mathbf{Y}}_2$

We Still Have Another Step (matching nodes between graphs)

But a question of interest is, how many landmarks do we need? These landmarks will be our effective embedding dimension.



(a) Accuracy w.r.t. # of landmarks



(b) Runtime w.r.t. # of landmarks

Figure: Here $p = t \log_2 n$

The Last Step: Distances Between Nodes in Different Graphs via Embeddings

Letting $\tilde{Y}_1$ and $\tilde{Y}_2$ be the embeddings for graphs 1 and 2 respectively, then pairwise node similarities between the graphs can be computed as,

$$\text{sim}_{emb}\left(\tilde{\mathbf{Y}}_1[u], \tilde{\mathbf{Y}}_2[v]\right) = e^{-\left\|\tilde{\mathbf{Y}}_1[u] - \tilde{\mathbf{Y}}_2[v]\right\|_2^2}$$

- Match nodes according to these similarity scores.
- **Soft Scoring Approach:** From kd-tree, find $\alpha << N$ top matches of nodes from the other graph.

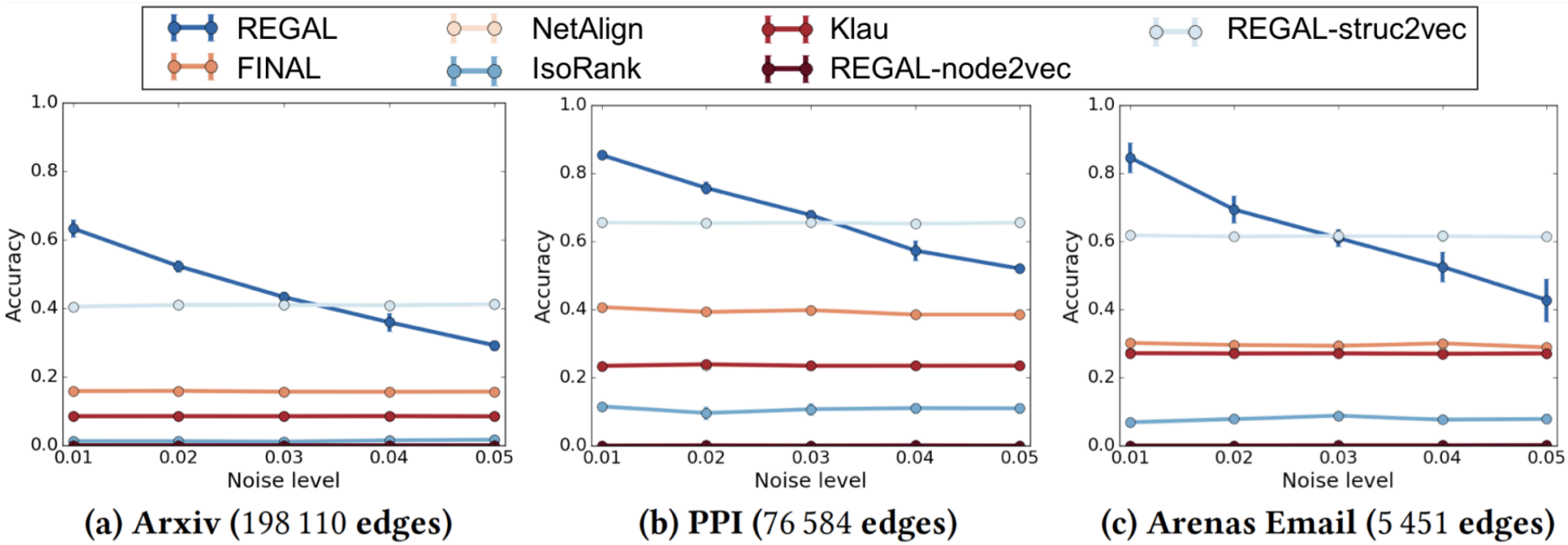
Experimental Evaluation

Given an adjacency matrix, $\mathbf{A}$, generate a random permutation matrix, $\mathbf{P}$ and generate a new network as,

$$\mathbf{A}' = \mathbf{PAP}^T$$

Further, remove edges from $\mathbf{A}'$ with probability p_s, and permute attributes with probability, p_a.

Results from Adding Noise



(a) **Arxiv** (198 110 **edges**) (b) **PPI** (76 584 **edges**) (c) **Arenas Email** (5 451 **edges**)

Figure: from Fig. 4, Replacing the embedding step with struct2vec produces better results when there is more noise.

REGAL Is More Ideal wrt Run-time

Dataset	Arxiv	PPI	Arenas
FINAL	4182 (180)	62.88 (32.20)	3.82 (1.41)
NetAlign	149.62 (282.03)	22.44 (0.61)	**1.89 (0.07)**
IsoRank	**17.04 (6.22)**	**6.14 (1.33)**	**0.73 (0.05)**
Klau	1291.00 (373)	476.54 (8.98)	43.04 (0.80)
REGAL-node2vec	709.04 (20.98)	139.56 (1.54)	15.05 (0.23)
REGAL-struc2vec	1975.37 (223.22)	441.35 (13.21)	74.07 (0.95)
REGAL	**86.80 (11.23)**	**18.27 (2.12)**	2.32 (0.31)

Figure: from Table 4. The methods that perform better in ‘noise‘ experiments also have higher run-time.

Conclusion

- REGAL computes embeddings via landmark points.
- The magic is in approximating the pairwise node similarity matrix through landmark points
- Performance is generally better than baselines for most ‘noise‘ levels.