Origin: Non-Rigid Network Alignment

Si Zhang

Hanghang Tong

Jiejun Xu

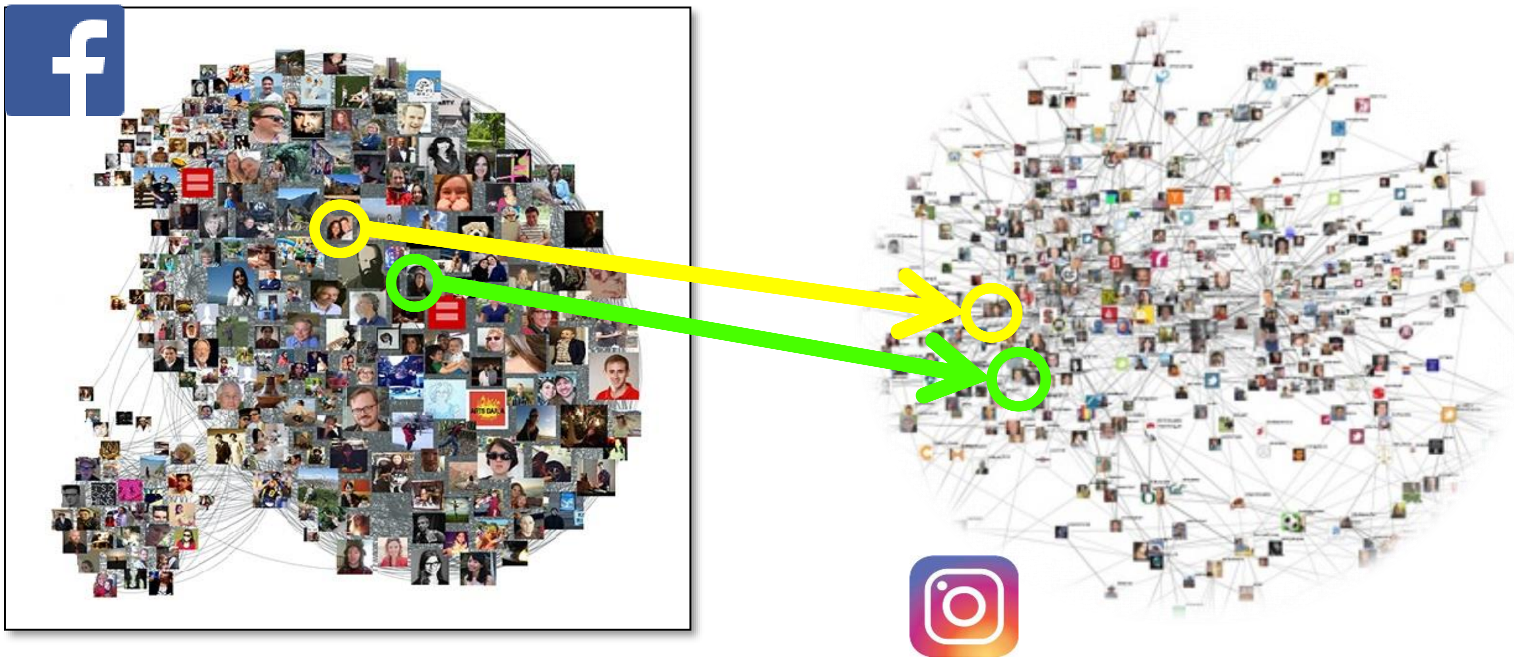
Yifan Hu

Ross Maciejewski

ILLINOIS

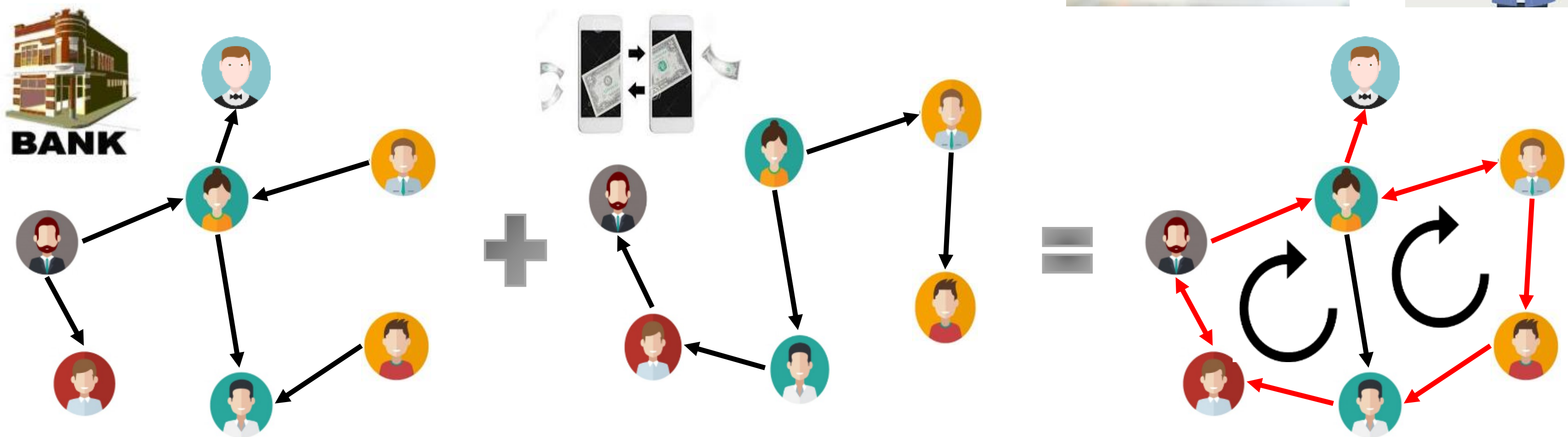
Network Alignment

- To find node correspondence across networks.



Network Alignment: Applications

- Fraud detection

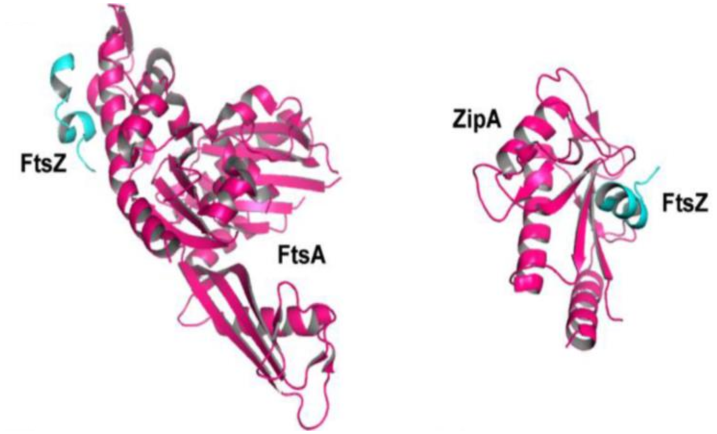


- **Unsuspicious patterns become suspicious!**

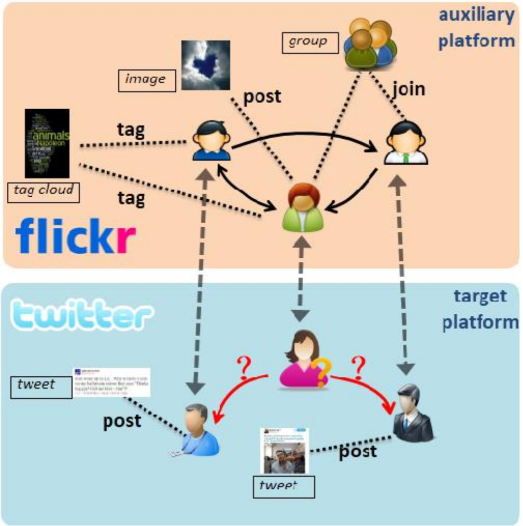
Network Alignment: Applications

- Other applications

Drug design
[Kazemi et al. 2016]



Friend recommendation
[Yan et al. 2013]



[1] Kazemi, Ehsan, et al. "PROPER: global protein interaction network alignment through percolation matching." *BMC bioinformatics* 2016
[2] Yan, Ming, et al. "Friend transfer: Cold-start friend recommendation with cross-platform transfer learning of social knowledge." *2013 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2013.

Existing Methods

- Graph matching based methods
  - Koopsmans-Beckmann's quadratic assignment problem (KB-QAP)

$$\begin{aligned} \max \quad & \mathrm{Tr}(\boldsymbol{S}^T \boldsymbol{A}_1 \boldsymbol{S} \boldsymbol{A}_2) + \mathrm{Tr}(\boldsymbol{H}^T \boldsymbol{S}) \\ s.t. \quad & \text{constraints on } \boldsymbol{S} \end{aligned}$$

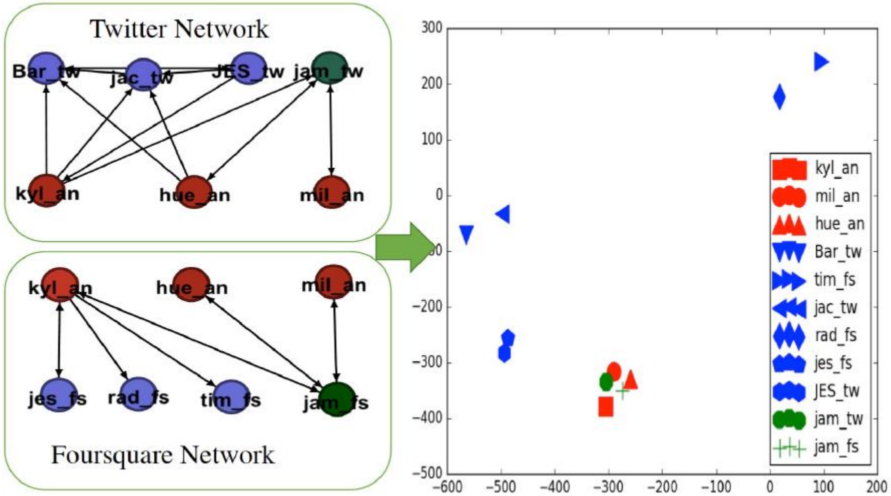
- Choices on constraints
  - $\boldsymbol{S}$ is a permutation matrix (exact constraint)
  - $\boldsymbol{S}$ is a doubly stochastic matrix (stochastic relaxation)

$$\boldsymbol{S} \in [0,1]^{n_1 \times n_2}, \boldsymbol{S}\mathbf{1}_{n_2} \leq \mathbf{1}_{n_1}, \boldsymbol{S}^T \mathbf{1}_{n_1} = \mathbf{1}_{n_2}$$

  - $\boldsymbol{S}$ is an orthogonal matrix (spectral relaxation)

Existing Methods (con't)

- Embedding based methods
  - Learn representations of nodes in different networks
  - Infer alignment by similarities among embedding vectors

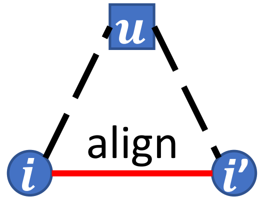


(showcase from Liu et al. 2016)

[1] Liu, Li, et al. "Aligning Users across Social Networks Using Network Embedding." *IJCAI*. 2016.

Limitation #1: Representation Power

- Koopsmans-Beckmann's QAP

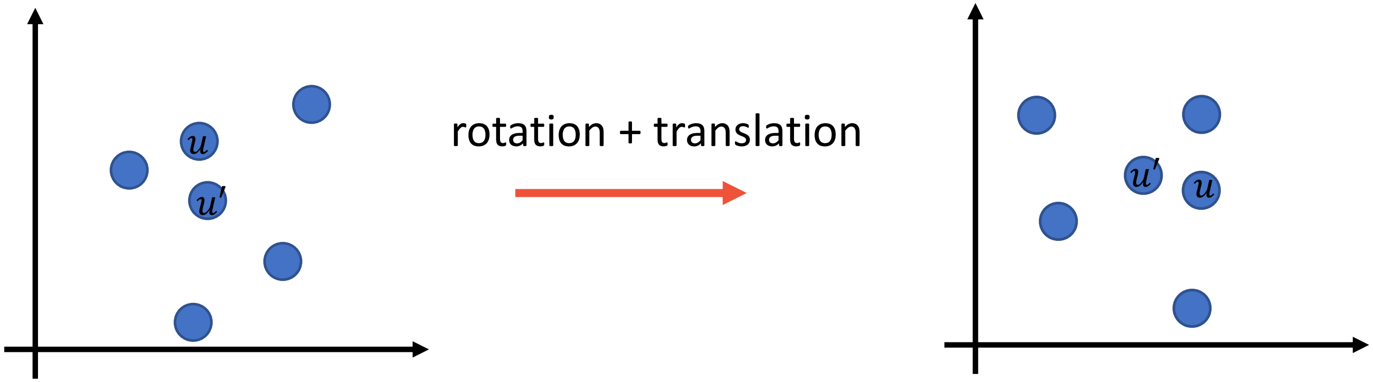


$$\max \ \ \mathrm{Tr}(\boldsymbol{S}^T \boldsymbol{A}_1 \boldsymbol{S} \boldsymbol{A}_2) = \sum_{u,k} (\boldsymbol{S}^T \boldsymbol{A}_1)_{uk} (\boldsymbol{A}_2 \boldsymbol{S}^T)_{uk}$$

  - Node-u feature vector: u-th row of linear transformations $\boldsymbol{S}^T \boldsymbol{A}_1$ & $\boldsymbol{A}_2 \boldsymbol{S}^T$
  - Inner product of feature vectors computed from $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$
  - Maximizing inner product similarities
- Limitations:
  - Linear transformation based on connections
  - High dimensions
- **Question:** How to learn better node representations?

Limitation #2: Representation Incomparability

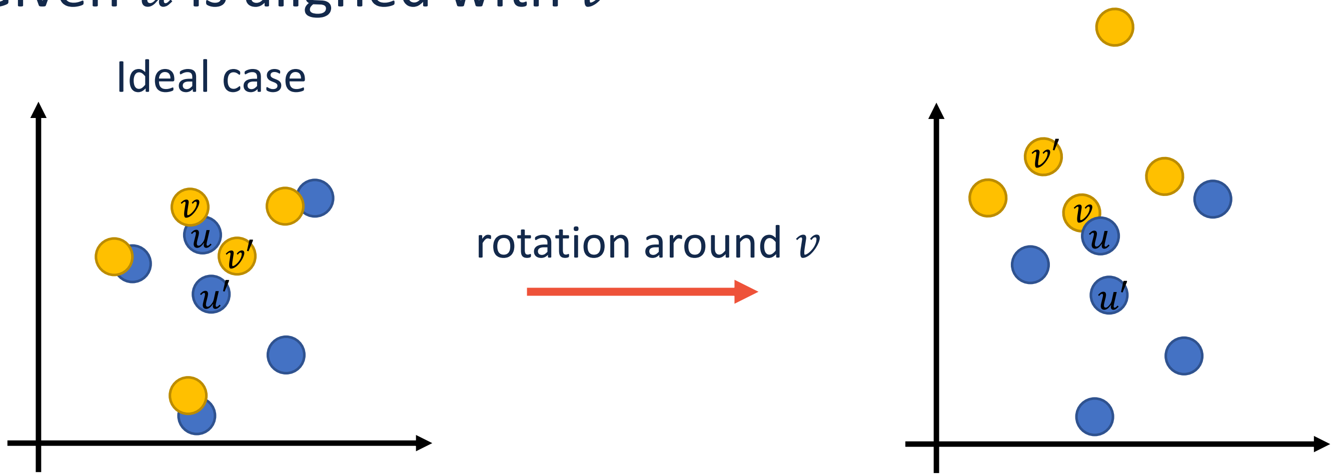
- Single network embedding



- Intra-network node similarities do not change
- Semantically rotation/translation invariance

Limitation #2 (con't)

- Multiple network embedding
  - Given u is aligned with v



- Inter-network node similarities totally changed!
- **Question:** How to address the representation incomparability?

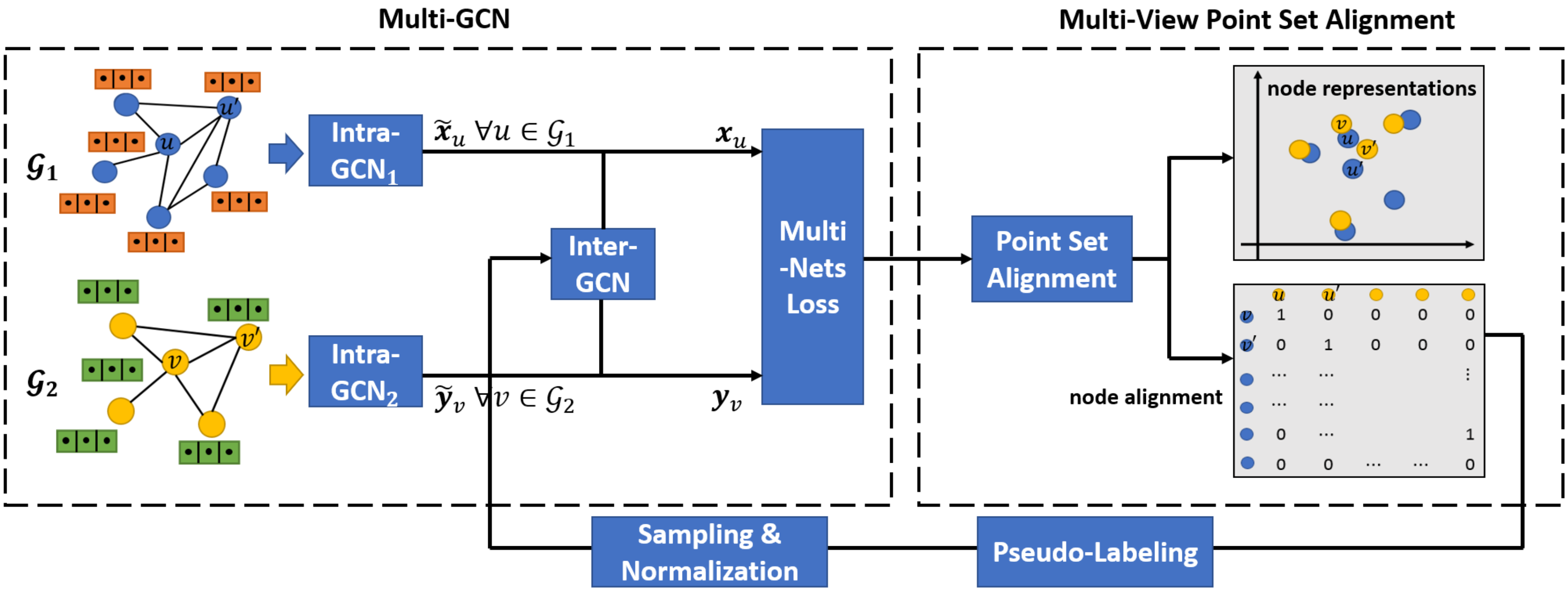
Prob. Def.: Non-Rigid Network Alignment

- **Input:**
  - (1) undirected networks $\mathcal{G}_1 = \{\mathcal{V}_1, \boldsymbol{A}_1, \boldsymbol{X}_1^0\}$ and $\mathcal{G}_2 = \{\mathcal{V}_2, \boldsymbol{A}_2, \boldsymbol{X}_2^0\}$;
  - (2) labeled aligned node pairs $\mathcal{L}^+ = \left\{\left(u_{l_i}, v_{l_i}\right) \mid i = 1, \cdots, L\right\}$;
  - (3) (optional) prior cross-network node similarity matrix $\boldsymbol{H}$.
- **Output:**
  - (1) alignment matrix $\boldsymbol{S}$;
  - (2) node representation matrices $\boldsymbol{Z}, \boldsymbol{Y}$ of $\mathcal{G}_1, \mathcal{G}_2$

Outline

- Motivations ✓
- **Model Overview**
- Q1: Multiple Network Representation Learning
- Q2: Non-Rigid Point Set Alignment
- Experiments
- Conclusions

Model Overview



Outline

- Motivations ✓
- Model Overview ✓
- **Q1: Multiple Network Representation Learning**
- Q2: Non-Rigid Point Set Alignment
- Experiments
- Conclusions

Single Network GCN

- Spatial-based GCN formulation (Intra-GCN)

$$\tilde{\boldsymbol{x}}_{\mathcal{N}_u}^{t} = \text{Aggregate}\left(\left\{\tilde{\boldsymbol{x}}_{u'}^{t-1}, \forall u' \in \mathcal{N}_u\right\}\right)$$
$$\tilde{\boldsymbol{x}}_{u}^{t} = \sigma\left(\left[\tilde{\boldsymbol{x}}_{u}^{t-1} || \tilde{\boldsymbol{x}}_{\mathcal{N}_u}\right]\boldsymbol{W}^{t}\right)$$

vector concatenation

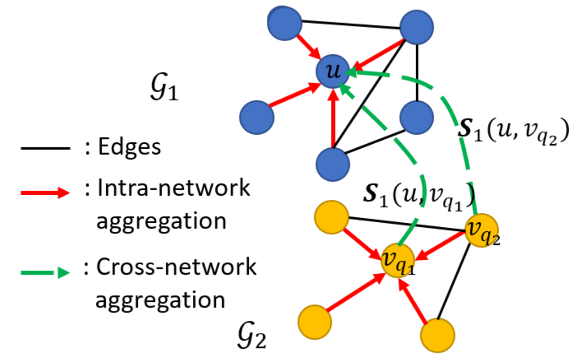
  - Aggregate hidden representations from neighborhood $\mathcal{N}_u$
  - Combine aggregated representation
  - Limitations: only aggregate within a single network
- **Question:** How to aggregate across different networks?

Multi-GCN: Formulation #1

- Cross-network aggregation via alignment

$$\widehat{\boldsymbol{x}}_u = \text{Aggregate}_{\text{cross}}(\widetilde{\boldsymbol{y}}_v) = \sum_{v \in \mathcal{V}_2} \boldsymbol{S}(u,v)\widetilde{\boldsymbol{y}}_v$$

$$\widehat{\boldsymbol{y}}_v = \text{Aggregate}_{\text{cross}}(\widetilde{\boldsymbol{x}}_u) = \sum_{u \in \mathcal{V}_1} \boldsymbol{S}(u,v)\widetilde{\boldsymbol{x}}_u$$



$$\boxed{\begin{aligned} \hat{\mathbf{x}}_u &= \text{AGGREGATE}_{\text{cross}}(\tilde{\mathbf{x}}_u) = \sum_{k=1}^{K} \mathbf{S}_1(u, v_{q_k})\tilde{\mathbf{y}}_{v_{q_k}} \\ \hat{\mathbf{y}}_v &= \text{AGGREGATE}_{\text{cross}}(\tilde{\mathbf{y}}_v) = \sum_{k=1}^{K} \mathbf{S}_2(u_{p_k}, v)\tilde{\mathbf{x}}_{u_{p_k}} \end{aligned}}$$

  - Sample on alignment $\boldsymbol{S}$ for aggregation localization and efficiency

- Cross-network combination

$$\boldsymbol{x}_u = \text{Combine}_{\text{cross}}(\widetilde{\boldsymbol{x}}_u, \widehat{\boldsymbol{x}}_u) = [\widetilde{\boldsymbol{x}}_u || \widehat{\boldsymbol{x}}_u]\boldsymbol{W}_{\text{cross}} + \boldsymbol{b}_1$$
$$\boldsymbol{y}_v = \text{Combine}_{\text{cross}}(\widetilde{\boldsymbol{y}}_v, \widehat{\boldsymbol{y}}_v) = [\widetilde{\boldsymbol{y}}_v || \widehat{\boldsymbol{y}}_v]\boldsymbol{W}_{\text{cross}} + \boldsymbol{b}_2$$

Multi-GCN: Formulation #2

- Multi-GCN loss function

Inter-network loss

$$\mathcal{J}_{\text{GCN}} = \boxed{\mathcal{J}_{\mathcal{G}_1}(\boldsymbol{X}) + \mathcal{J}_{\mathcal{G}_2}(\boldsymbol{Y})} + \lambda \mathcal{J}_{\text{cross}}(\boldsymbol{X}, \boldsymbol{Y})$$

Intra-network loss (e.g., SkipGram)

- Inter-network disagreement loss

$$\mathcal{J}_{\text{cross}}(\boldsymbol{X}, \boldsymbol{Y}) = \sum_{u \in \mathcal{V}_1} \left\| \boldsymbol{x}_u - \boxed{\sum_{k=1}^{K} \boldsymbol{S}_1(u, v_{q_k}) \boldsymbol{y}_{q_k}} \right\|_2^2 + \sum_{v \in \mathcal{V}_1} \left\| \boldsymbol{y}_v - \sum_{k=1}^{K} \boldsymbol{S}_2(u_{p_k}, v) \boldsymbol{x}_{u_{p_k}} \right\|_2^2$$

$$\boxed{\left[\sum_{k=1}^{K} \boldsymbol{S}_1(u, v_{q_k}) \tilde{\boldsymbol{y}}_{v_{q_k}} \,||\, \sum_{k'} (\boldsymbol{S}_1 \boldsymbol{S}_2^T)\left(u, u_{p_{k'}}\right) \tilde{\boldsymbol{x}}_{u_{p_{k'}}} \right] \boldsymbol{W}_{\text{cross}} + \boldsymbol{b}_2}$$

Cross-network

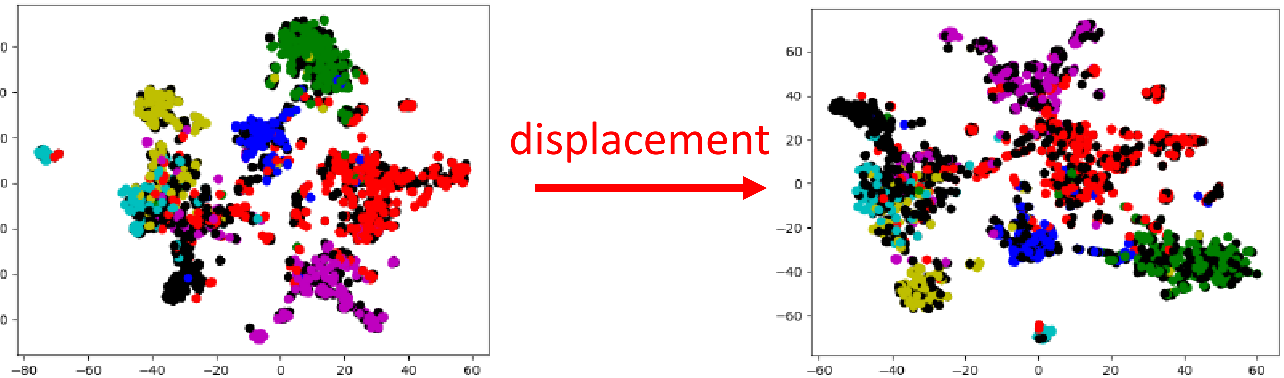
Within-network by alignment

Outline

- Motivations ✓
- Model Overview ✓
- Q1: Multiple Network Representation Learning ✓
- **Q2: Non-Rigid Point Set Alignment**
- Experiments
- Conclusions

Non-Rigid Point Set Alignment (NR-PSA)

- Goal: to address the representation incomparability
- Key ideas:
  - View node representation vectors as points in Euclidean space
  - Displace one point set towards another based on labeled alignment
  - Move coherently in two views (i.e., point view and node view)



point set of $\mathcal{G}_1$ point set of $\mathcal{G}_2$

NR-PSA: Formulation #1

- Intuition: to maximize labeled node-pair overlaps
- Given labeled node alignment $(u_{l_i}, v_{l_i}), l = 1, \cdots, L$

$$\min_{\boldsymbol{f}} \sum_{i=1}^{L} \left\| \boldsymbol{x}_{u_{l_i}} + \frac{1}{2} \boxed{\boldsymbol{f}\left(\boldsymbol{x}_{u_{l_i}}\right)} - \boldsymbol{y}_{v_{l_i}} \right\|_2^2 + \alpha \|\boldsymbol{f}\|_{\mathcal{H}}^2$$

vector-valued non-rigid displacement function

- Minimize vector distances after displacement
- Functional minimization problem
- $\|\boldsymbol{f}\|_{\mathcal{H}}^2$ is the RKHS norm for regularization

NR-PSA: Formulation #2

- Intuition: each point $x_{u_{l_i}}$ has two interpretations (views)
  - Representation vectors in the Euclidean space
  - Nodes of networks in the non-Euclidean graph space
- Divide $\mathcal{H}$ into two RKHS, i.e., $\mathcal{H} = \mathcal{H}_1 \oplus \mathcal{H}_2$, such that

$$\mathcal{H} = \{\boldsymbol{f} | \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}^1(\boldsymbol{x}) + \boldsymbol{f}^2(\boldsymbol{x}), \boldsymbol{f}^1 \in \mathcal{H}_1, \boldsymbol{f}^2 \in \mathcal{H}_2$$

- Re-write RKHS norm regularization into

$$\|\boldsymbol{f}\|_{\mathcal{H}}^2 = \min_{\substack{\boldsymbol{f}=\boldsymbol{f}^1+\boldsymbol{f}^2 \\ \boldsymbol{f}^1 \in \mathcal{H}_1 \\ \boldsymbol{f}^2 \in \mathcal{H}_2}} \alpha_1 \|\boldsymbol{f}^1\|_{\mathcal{H}_1}^2 + \alpha_2 \|\boldsymbol{f}^2\|_{\mathcal{H}_2}^2 + \mu \boxed{\sum_{j=1}^{n_1 - L} \left[\boldsymbol{f}^1\left(\boldsymbol{x}_{u_{r_j}}\right) - \boldsymbol{f}^2\left(\boldsymbol{x}_{u_{r_j}}\right)\right]^2}$$

displacement consistency in two views
on the unlabeled nodes

NR-PSA: Formulation #2 (con't)

- By representer theorem

$$\boldsymbol{f}(\boldsymbol{x}_u) = \boldsymbol{K}(u, \mathcal{I})\mathbf{T}$$

  - Matrix $\boldsymbol{K}$: kernel matrix computed by reproducing kernels in $\mathcal{H}_1, \mathcal{H}_2$
  - $\mathbf{T}$ is the matrix variable and $\mathcal{I} = \{u_{l_i} | i = 1, \cdots, L\}$

- Matrix-form objective function

$$\min_{\mathbf{T}} \ \mathcal{J}_{\mathrm{PSA}} = \sum_{i=1}^{L} \left\| \boldsymbol{x}_{u_{l_i}} + \frac{1}{2}\boldsymbol{K}(u_{l_i}, \mathcal{I})\mathbf{T} - \mathbf{y}_{\mathrm{v}_{l_i}} \right\|_2^2 + \alpha \mathrm{Tr}(\mathbf{T}^T \boldsymbol{K}_{\mathcal{I}} \mathbf{T})$$

  - $\boldsymbol{K}_{\mathcal{I}} = \boldsymbol{K}(\mathcal{I}, \mathcal{I})$
  - Details in the paper.

Origin: Algorithm

- Alternating between two stages
  - Stage #1: to learn node representations based on current alignment
  - Stage #2: to solve for the displacement function
- Stage #1: mini-batched SGD
- Stage #2: gradient descent

$$\frac{\partial \mathcal{J}_{\mathrm{PSA}}}{\partial \mathbf{T}} = \frac{1}{2} \boldsymbol{K}_{\mathcal{I}}^{T} \boldsymbol{K}_{\mathcal{I}} \mathbf{T} + \boldsymbol{K}_{\mathcal{I}}^{T} \big(\boldsymbol{X}(\mathcal{I},:) - \boldsymbol{Y}(\mathcal{I},:) \big) + 2\alpha \boldsymbol{K}_{\mathcal{I}} \mathbf{T}$$

  - Time complexity: sub-quadratic w.r.t # of nodes
- Outputs displaced node representations $\boldsymbol{Z}$ of $\mathcal{G}_1$

Outline

- Motivations ✓
- Model Overview ✓
- Q1: Multiple Network Representation Learning ✓
- Q2: Non-Rigid Point Set Alignment ✓
- **Experiments**
- Conclusions

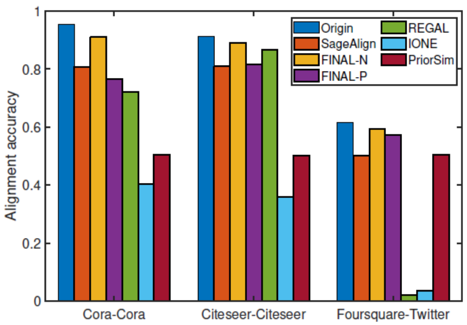
Experiment Setup

- Datasets:
  - Cora-1 & Cora-2 networks (nodes: 2,708 vs. 2,708)
  - Citeseer-1 & Citeseer-2 networks (nodes: 3,327 vs. 3,327)
  - Foursquare & Twitter networks (nodes: 5,313 vs. 5,120)
- Evaluation objectives:
  - Effectiveness: alignment accuracy
  - Efficiency: running time
- Comparison methods:

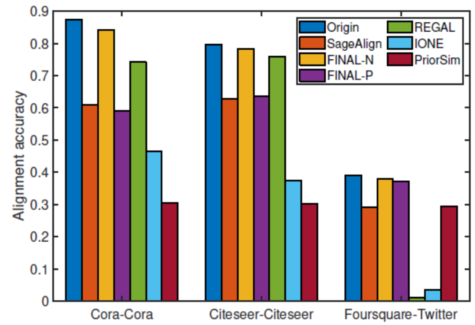
Methods	**Categories**
Origin	Representation based
SageAlign	Representation based
FINAL-N [1]	KB-QAP based
FINAL-P [1]	KB-QAP based
REGAL [2]	Representation based
IONE [3]	Representation based
PriorSim	Heuristics

[1] Zhang, Si, and Hanghang Tong. "Final: Fast attributed network alignment." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016.
[2] Heimann, Mark, et al. "Regal: Representation learning-based graph alignment." *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2018.
[3] Liu, Li, et al. "Aligning Users across Social Networks Using Network Embedding." *IJCAI*. 2016.

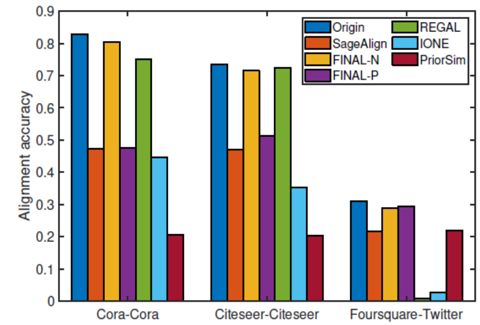
R1. Effectiveness



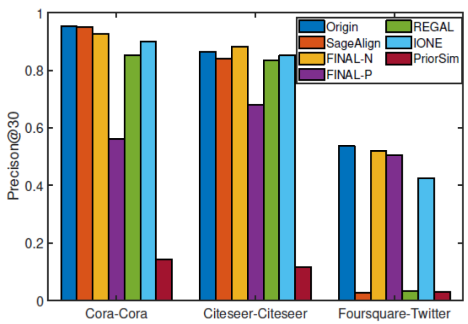
(a) 50% labeled alignment.



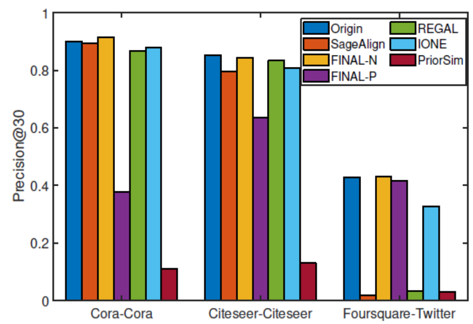
(b) 30% labeled alignment.



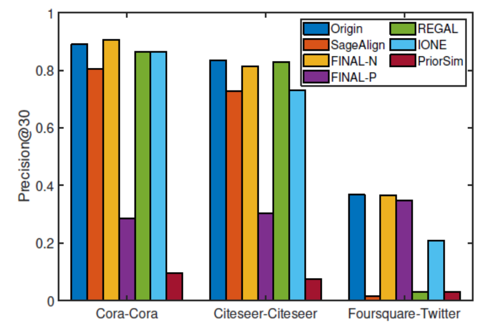
(c) 20% labeled alignment.



(a) 50% labeled alignment.



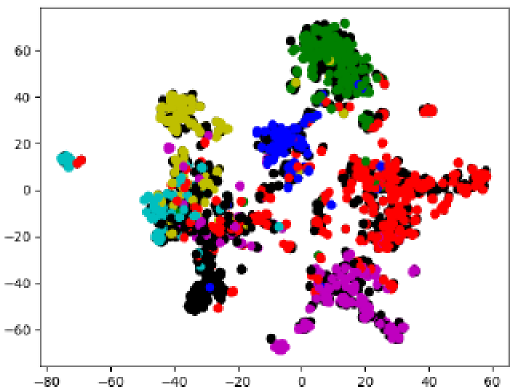
(b) 30% labeled alignment.



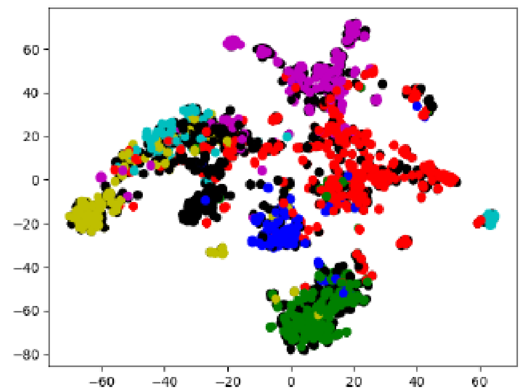
(c) 20% labeled alignment.

Observation: outperforms both QAP-based methods FINAL and other embedding-based methods.

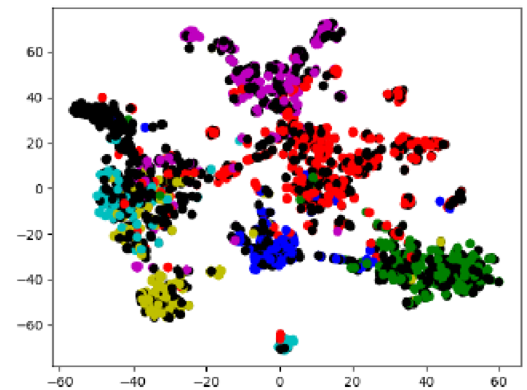
R2. Visualizations



(a) Cora-1 node representations (i.e. X).



(b) Displaced cora-1 representations (i.e. Z).

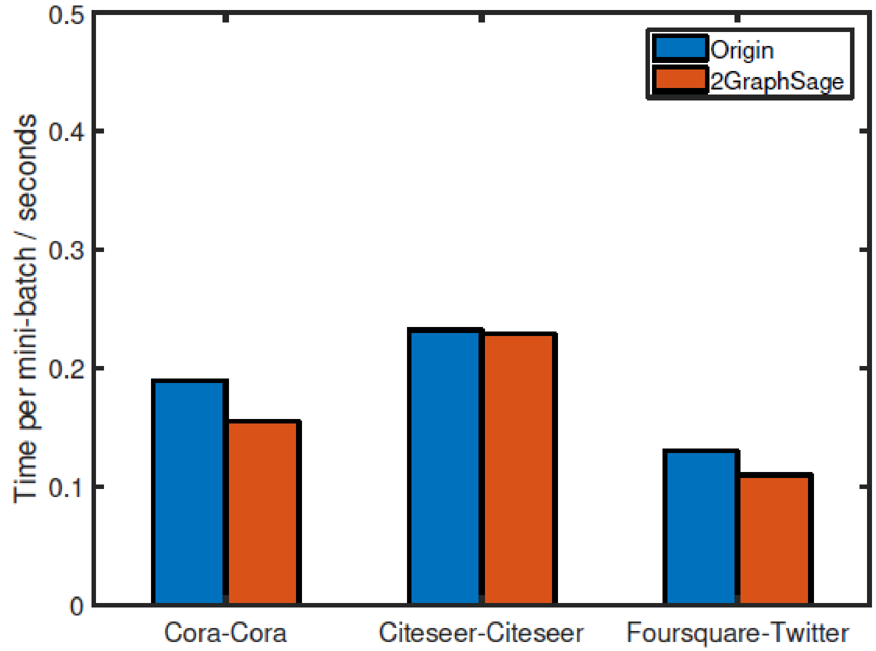


(c) Cora-2 node representations (i.e. Y).

Observations:

- Embeddings $\boldsymbol{X}, \boldsymbol{Y}$ of $\mathcal{G}_1, \mathcal{G}_2$ are misleading even with cross-network disagreement loss;
- Displaced embeddings $\boldsymbol{Z}, \boldsymbol{Y}$ are more accurate for alignment.

R3. Efficiency



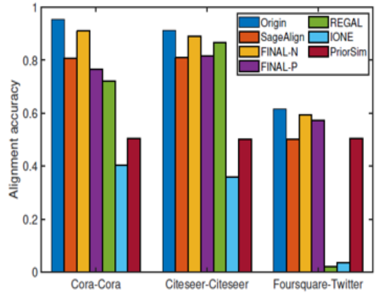
Observation: the extra computational cost for Inter-GCN is quite light.

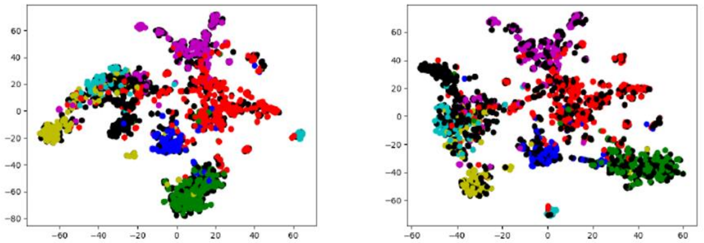
Outline

- Motivations ✓
- Model Overview ✓
- Q1: Multiple Network Representation Learning ✓
- Q2: Non-Rigid Point Set Alignment ✓
- Experiments ✓
- **Conclusions**

Conclusions

- Problem: Non-rigid network alignment
- Solutions (proposed Origin algorithm):
  - Multi-GCN: node representation learning across networks based on GCN
  - NR-PSA: non-rigid point-set alignment in two views
- Results:
  - Find more accurate node correspondence
  - Learn more meaningful node representations
  - Efficient compared to single-network counterpart
- More details in paper.





Thank You!