



# 组合优化问题的机器学习求解初探

Junchi Yan

Dept. of Computer Science & Engineering  
Artificial Intelligence Institute  
MoE Key Lab of AI  
[thinklab.sjtu.edu.cn](http://thinklab.sjtu.edu.cn)

严骏驰

上海交通大学计算机科学与工程系  
上海交通大学人工智能研究院  
人工智能教育部重点实验室  
[thinklab.sjtu.edu.cn](http://thinklab.sjtu.edu.cn)

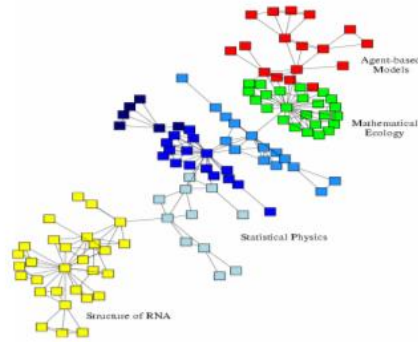
# Outline

- **Introduction**
- Background: Learning on Graph Matching
- Embedding approach for Deep Graph Matching
- Other techniques
- Outlook

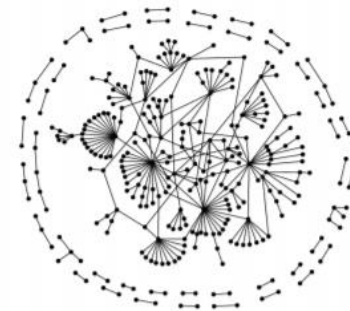
# Graphs Everywhere



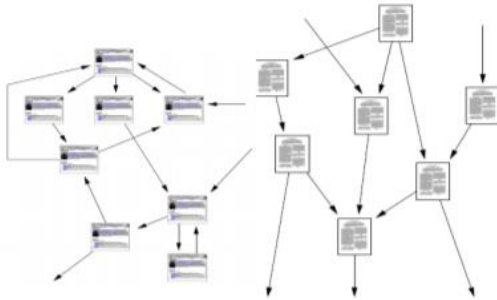
Social networks



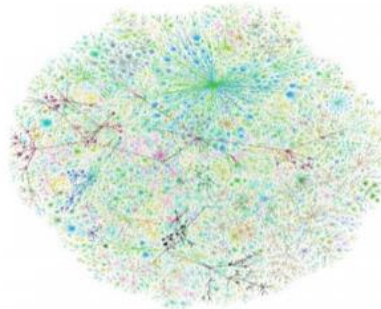
Economic networks



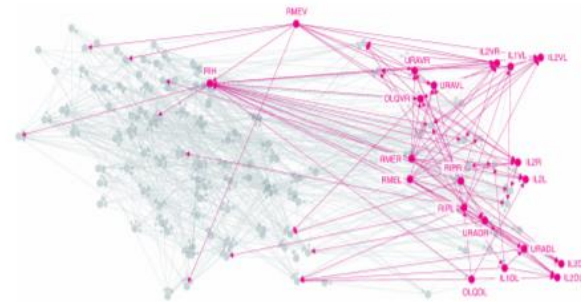
Communication graphs



Information networks:  
Web & citations



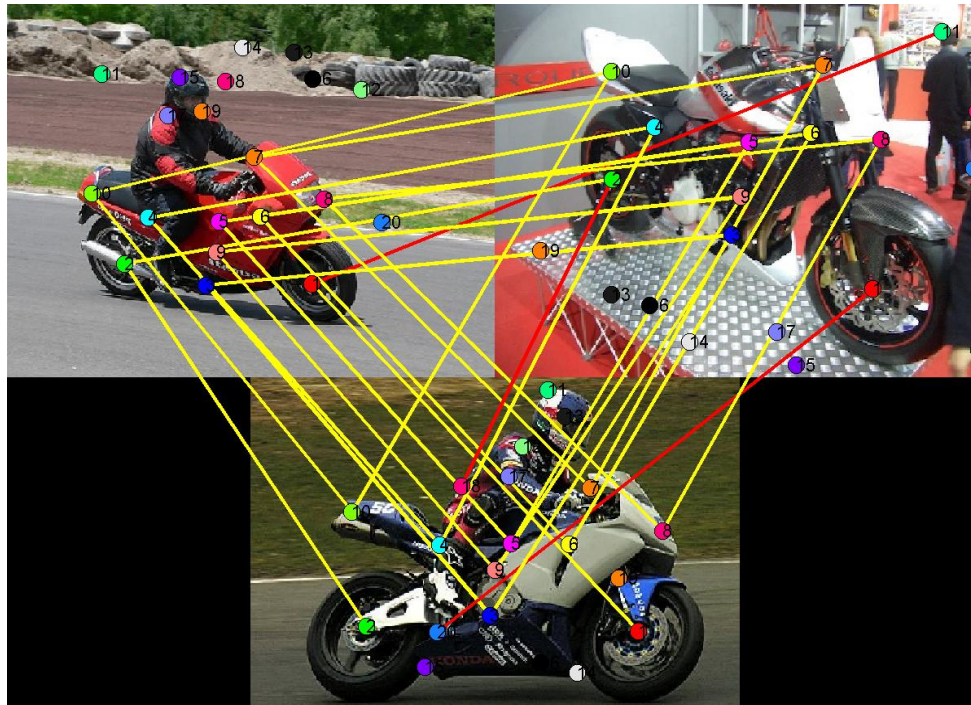
Internet



Networks of neurons

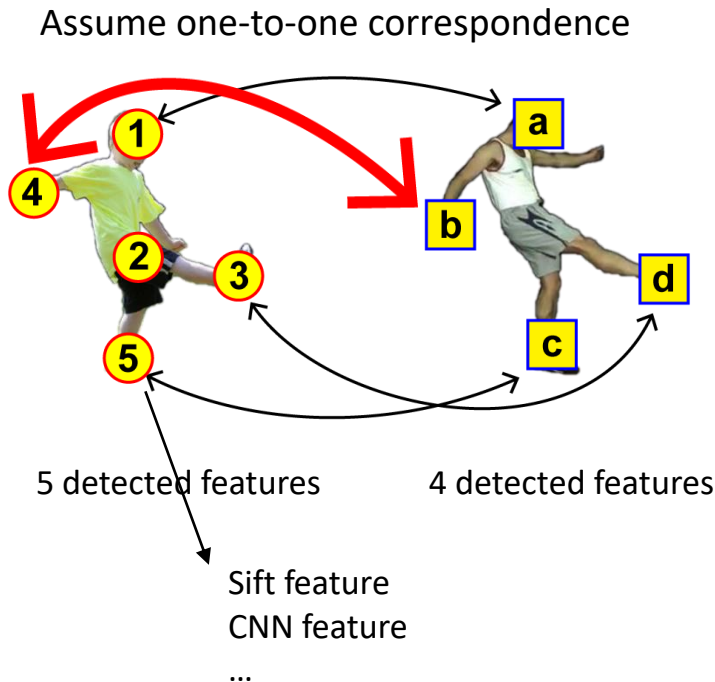
# What is Graph Matching

**Graph Matching** finds node correspondence among graphs.

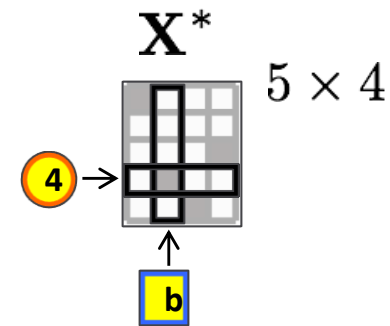


Yan, Cho, Zha, et al. "Multi-Graph Matching via Affinity Optimization with Graduated Consistency Regularization." IEEE T-PAMI 2016.

# Node correspondence by an assignment matrix



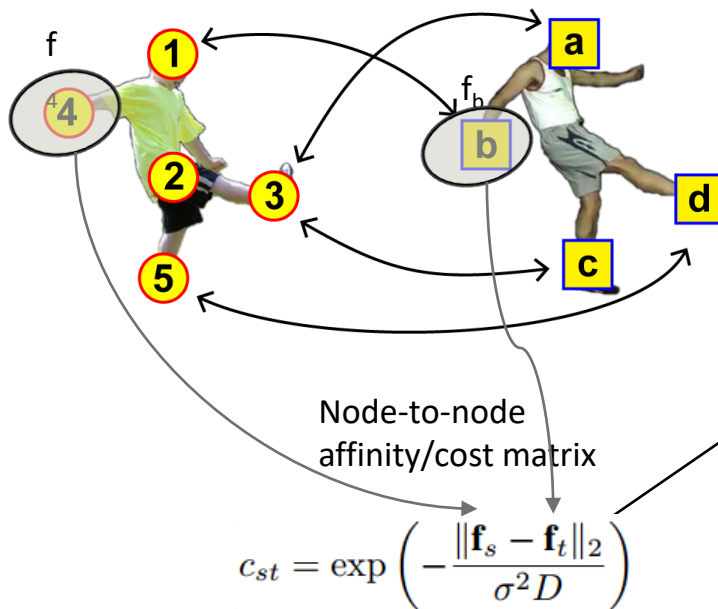
Solution: assignment matrix, i.e. a partial permutation matrix



$$\mathbf{X} \in \{0, 1\}^{5 \times 4}$$

$$\mathbf{X}\mathbf{1} \leq \mathbf{1}, \quad \mathbf{X}^T\mathbf{1} = \mathbf{1}$$

# Node-wise linear assignment problem



Linear Assignment Problem

$$\begin{aligned} \max_{\mathbf{X}} \quad & \text{tr} \left( \mathbf{X} \mathbf{K}_p^T \right) \\ \text{s. t.} \quad & \mathbf{X} \in \{0, 1\}^{5 \times 4} \\ & \mathbf{X} \mathbf{1} \leq \mathbf{1}, \quad \mathbf{X}^T \mathbf{1} = \mathbf{1} \end{aligned}$$

**Hungarian Algorithm**

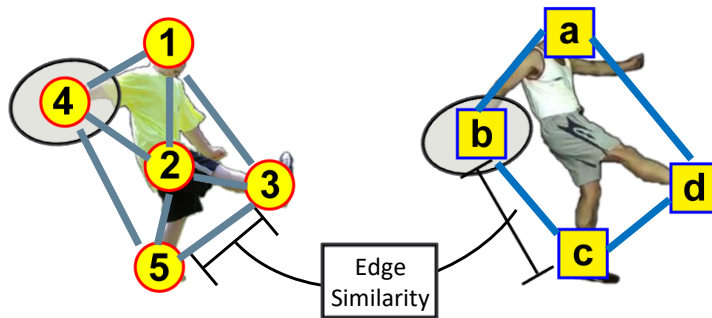
(Kuhn & Munkres, 1955)

Global optimality is ensured by

$O(n^3)$  time complexity where  $n$  is the number of nodes

# Edge-wise graph matching

Build graph by Delaunay Triangulation



1st-order Feature (eg. Local Texture)

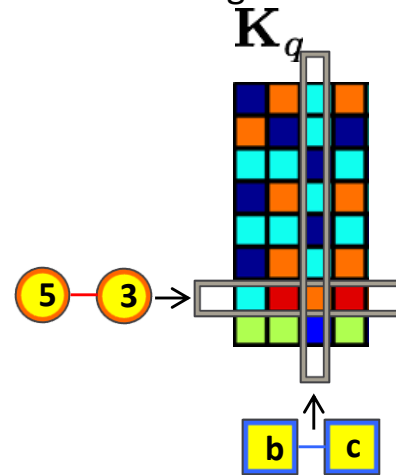
Feature Matching (linear)

+

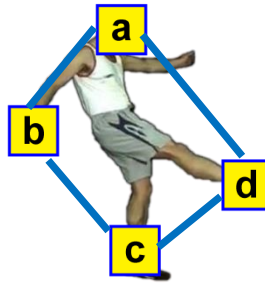
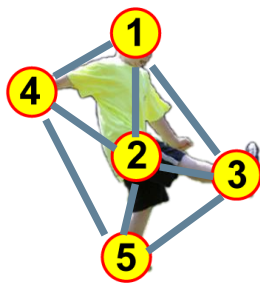
2nd-order Feature (eg, Edge Length)

Graph Matching (quadratic)

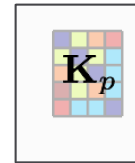
Edge Similarity



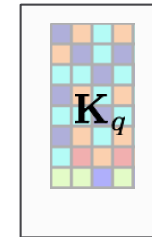
# Graph matching: a combinatorial optimization formulation



Node Similarity



Edge Similarity



Affinity  
maximization



Steven Gold



A. Rangarajan

$\max_{\mathbf{X}}$

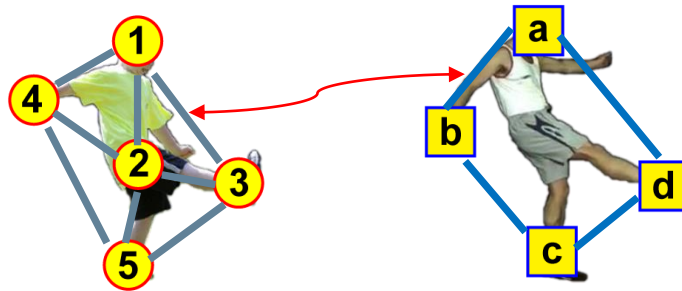
$$\sum_{i_1 i_2} x_{i_1 i_2} \kappa_{i_1 i_2}^p + \sum_{\substack{(i_1, j_1) \in E_1 \\ (i_2, j_2) \in E_2}} x_{i_1 i_2} x_{j_1 j_2} \kappa_{c(i_1, j_1) c(i_2, j_2)}^q$$

Node-Edge Index

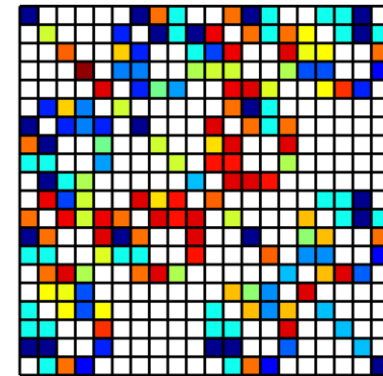
s. t.  $\mathbf{X} \in \{0, 1\}^{5 \times 4}, \quad \mathbf{X} \mathbf{1} \leq \mathbf{1}, \quad \mathbf{X}^T \mathbf{1} = \mathbf{1}$

S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *IEEE Transaction on PAMI*, 1996

# A more clean writing by an affinity matrix



Affinity matrix:  $\mathbf{K}$  (Leordeanu & Hebert, 2005)



Edge-to-edge relations  $\uparrow$   $20 \times 20$

$5 \cdot 4$



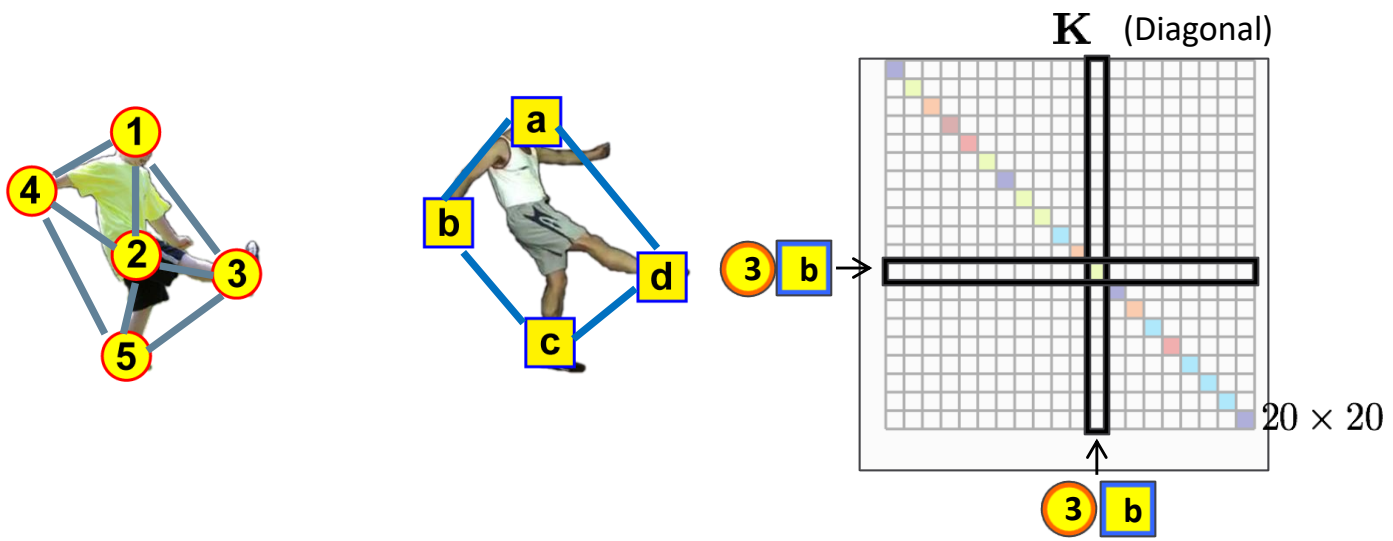
M. Leordeanu



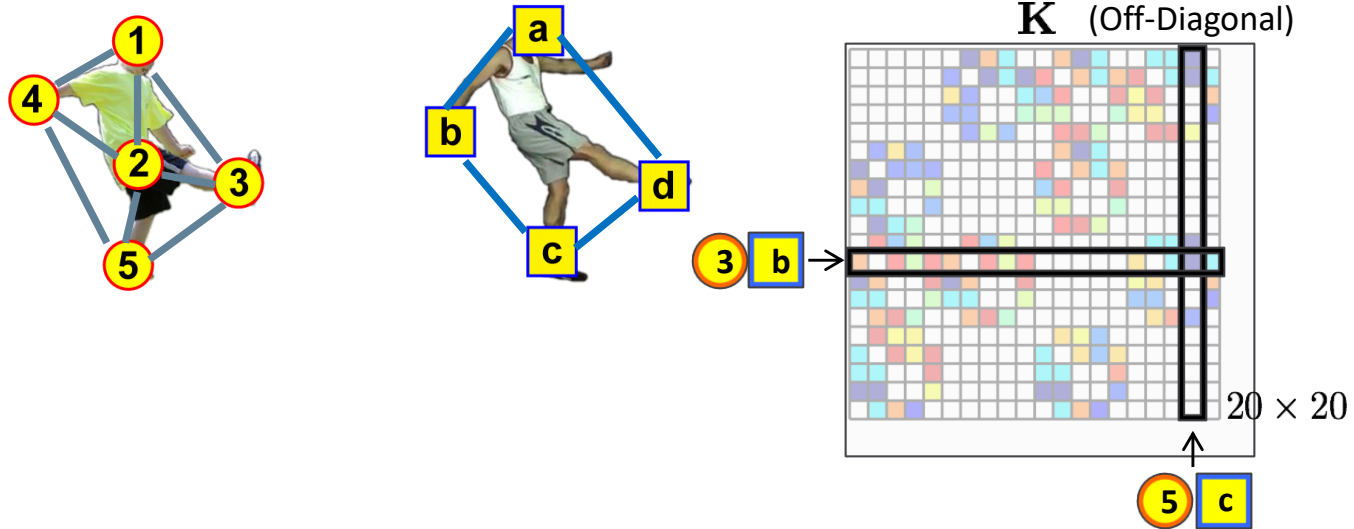
M. Hebert

M. Leordeanu and M. Hebert, “A spectral technique for correspondence problems using pairwise constraints,” in ICCV, 2005

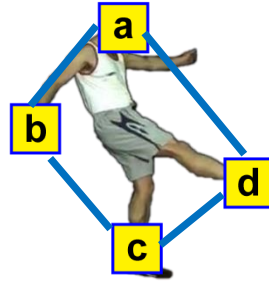
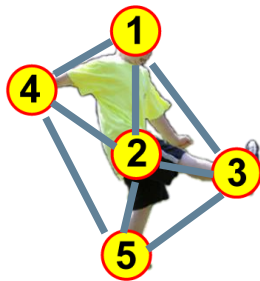
# Node-to-node affinity



# Edge-to-edge affinity



# Quadratic Assignment Problem



$$\begin{aligned} \max_{\mathbf{X}} \quad & \text{vec}(\mathbf{X})^T \mathbf{K} \text{vec}(\mathbf{X}) \\ \text{s. t.} \quad & \mathbf{X} \in \{0, 1\}^{5 \times 4} \\ & \mathbf{X}\mathbf{1} \leq \mathbf{1}, \quad \mathbf{X}^T\mathbf{1} = \mathbf{1} \end{aligned}$$

**NP-hard**

Branch-Bound

*Koopmans & Beckmann, 1955*

*Lawler, 1963*

*Loiola et al, 2007*

# Spectral Approximation

$$\max_{\mathbf{X}} \text{vec}(\mathbf{X})^T \mathbf{K} \text{vec}(\mathbf{X})$$

$$\text{s. t. } \mathbf{X} \in \{0, 1\}^{5 \times 4}$$

$$\mathbf{X} \mathbf{1} \leq \mathbf{1}, \mathbf{X}^T \mathbf{1} = \mathbf{1}$$

$$\|\text{vec}(\mathbf{X})\|_2^2 = 1$$

Faster

Not Tight

Not Discrete

*Spectral Method  
is Faster*



M. Leordeanu



M. Hebert



T. Cour



J. Shi

M. Leordeanu and M. Hebert, “A spectral technique for correspondence problems using pairwise constraints,” in ICCV, 2005

P. S. T. Cour and J. Shi, “Balanced graph matching,” in NIPS, 2006

# Double-stochastic Approximation

- S. Gold & Rangarajan, 1996
- Cho et al, 2010
- Leordeanu et al, 2009
- ...

$$\max_{\mathbf{X}} \text{vec}(\mathbf{X})^T \mathbf{K} \text{vec}(\mathbf{X})$$

**Not Convex**  
(K is Indefinite)

$$\text{s. t. } \mathbf{X} \in \{0, 1\}^{5 \times 4} \quad \mathbf{X} \in [0, 1]^{5 \times 4}$$
$$\mathbf{X}\mathbf{1} \leq \mathbf{1}, \quad \mathbf{X}^T\mathbf{1} = \mathbf{1}$$

**Slower**

**Not Discrete**

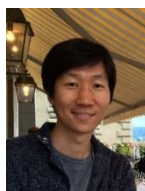
*Gradient Method  
is More Accurate*



S. Gold



A. Rangarajan



M. Cho



K. Lee



M. Leordeanu



M. Hebert



R. Sukthankar

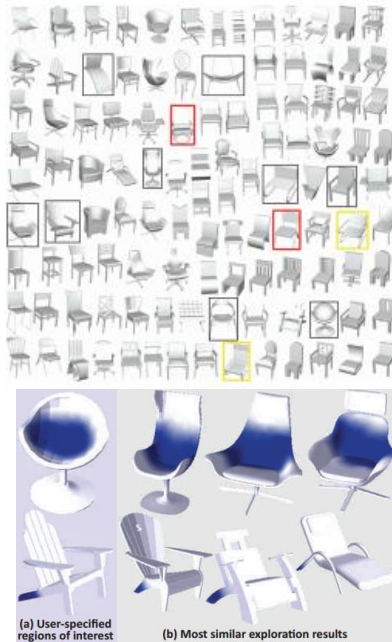
S. Gold and A. Rangarajan, “A graduated assignment algorithm for graph matching,” IEEE Transaction on PAMI, 1996

M. Cho, J. Lee, and K. M. Lee, “Reweighted random walks for graph matching,” in ECCV, 2010

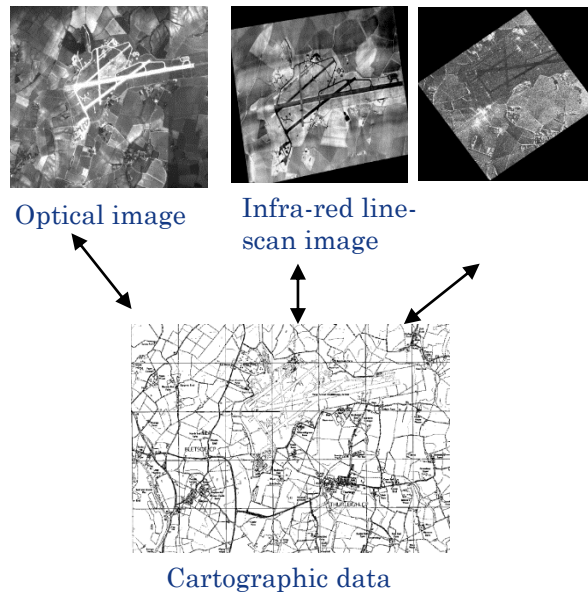
M. Leordeanu, M. Hebert, and R. Sukthankar, “An integer projected fixed point method for graph matching and map inference,” in NIPS, 2009

# Matching more than two graphs

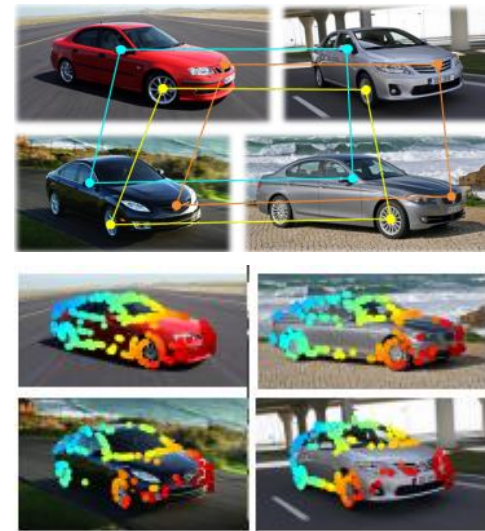
- More practical problem, with more information to use



Graphical object query and indexing, shape analysis  
SIGGRAPH'12



Info fusion PRL'97



3-D weak reconstruction  
ICCV'15

Exploring collections of 3D models using fuzzy correspondences, SIGGRAPH'12  
Multiple Graph Matching with Bayesian Inference, Pattern recognition letters'97  
Multi-Image Matching via Fast Alternating Minimization, ICCV'15

# Existing multiple GM methods

## Main categories

✓ Designate one of the graphs as the reference, and match all the others to the reference graph

- A. Sole-Ribalta, F. Serratosa, Models and algorithms for computing the common labelling of a set of attributed graphs, CVIU 2011

✓ Compute pairwise matchings, based on which improve overall accuracy

- D. Pachauriy, R. Kondorx, V. Singh, Solving the multi-way matching problem by permutation synchronization, in NIPS 2013
- Y. Chen, G. Leonidas, and Q. Huang. Matching partially similar objects via matrix completion. In ICML, 2014

✓ One-shot multiple feature set (**not graph**) matching

- Z. Zeng, T. H. Chan, K. Jia, and D. Xu. Finding correspondence from multiple images via sparse and low-rank decomposition. In ECCV, 2012
- X. Zhou, M. Zhu, and K. Daniilidis. Multi-image matching via fast alternating minimization. In ICCV, 2015



F. Serratosa



D. Pachauriy



Y. Chen



Z. Zeng



X. Zhou

# Composition based Affinity Optimization

Key idea: efficiently generate new candidate solutions by composition

$$\mathbf{x}_{ij}^{(t)} = \mathbf{x}_{ik}^{(t-1)} \mathbf{x}_{kj}^{(t-1)}$$

Find higher affinity score solution via composition

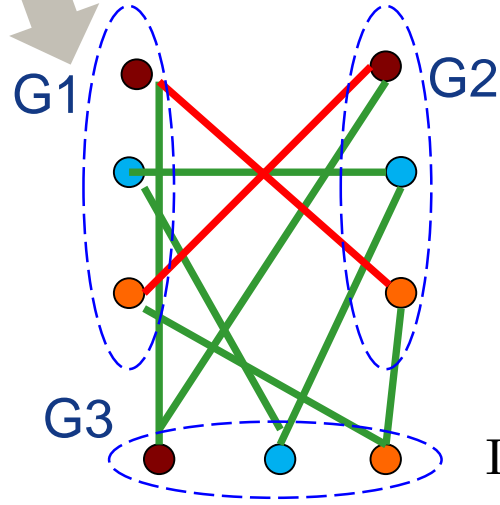
$$k^* = \arg \max_{k=1}^N J(\mathbf{x}_{ik}^{(t-1)} \mathbf{x}_{kj}^{(t-1)})$$



Mother



Father



Interpolating graph (son)



**Affinity**  $J$

$$\max_{\mathbf{X}} \text{vec}(\mathbf{X})^T \mathbf{K} \text{vec}(\mathbf{X})$$

$$\mathbf{X}^* = \arg \max_{\mathbf{X}} \sum_{i,j=1,2,\dots,N} \lambda_{ij} \text{vec}(\mathbf{X}_{ij})^T \mathbf{K}_{ij} \text{vec}(\mathbf{X}_{ij})$$

s.t.  $\mathbf{X}_{ij} \mathbf{1}_n = \mathbf{1}_n \quad \mathbf{1}_n^T \mathbf{X}_{ij} = \mathbf{1}_n^T \quad \mathbf{X}_{ij} = \mathbf{X}_{ji}^T \in \{0, 1\}^{n \times n}$

G1->G2 ✗  
 G1->G3->G2 ✓

# A simple & general algorithm

Still only local two graphs are involved in evaluation function  $J_{ij}$

Local noise

Modeling error

$$c_{st} = \exp\left(-\frac{\|\mathbf{f}_s - \mathbf{f}_t\|_2}{\sigma^2 D}\right)$$

Post-processing  
For enforcing overall consistency

**Proposition 1.** Alg.1 (CAO) is ensured to converge to a stationary configuration  $\mathbb{X}^*$  after a finite number of iterations.

---

## Algorithm 1 Composition based Affinity Optimization CAO

---

**Require:**  $\{\mathbf{K}_{ij}\}_{i=1, j=i+1}^{N-1, N}$ ,  $T$ ,  $\gamma \in (0, 1)$ ;

1: Perform pairwise matching to obtain initial  $\mathbb{X}^{(0)}$ ;

2: Calculate  $J^{(0)} = \sum_{i=1, j=i+1}^{N-1, N} \text{vec}(\mathbf{X}_{ij}^{(0)})^T \mathbf{K}_{ij} \text{vec}(\mathbf{X}_{ij}^{(0)})$ ;

3: **for**  $t = 1 : T$  **do**

4:   **for all**  $i = 1, 2, \dots, N-1; j = i+1, \dots, N$  **do**

5:     update  $\mathbf{X}_{ij}^{(t)} = \mathbf{X}_{ik}^{(t-1)} \mathbf{X}_{kj}^{(t-1)}$  by solving Eq.4;

6:   **end for**

7: **end for**

8: **if**  $C(\mathbb{X}^{(t)}) = 1$ , **return**  $\mathbb{X}_c^* = \mathbb{X}^{(t)}$ ;

9: **if**  $C(\mathbb{X}^{(t)}) < \gamma$  **then**

10:   Build the super graph  $\mathcal{G}_{sup}^a$  by pairwise affinity  $J(\mathbf{X}_{ij}^{(t)})$  and find a maximum span tree to generate  $\mathbb{X}_c^*$ ;

11: **else**

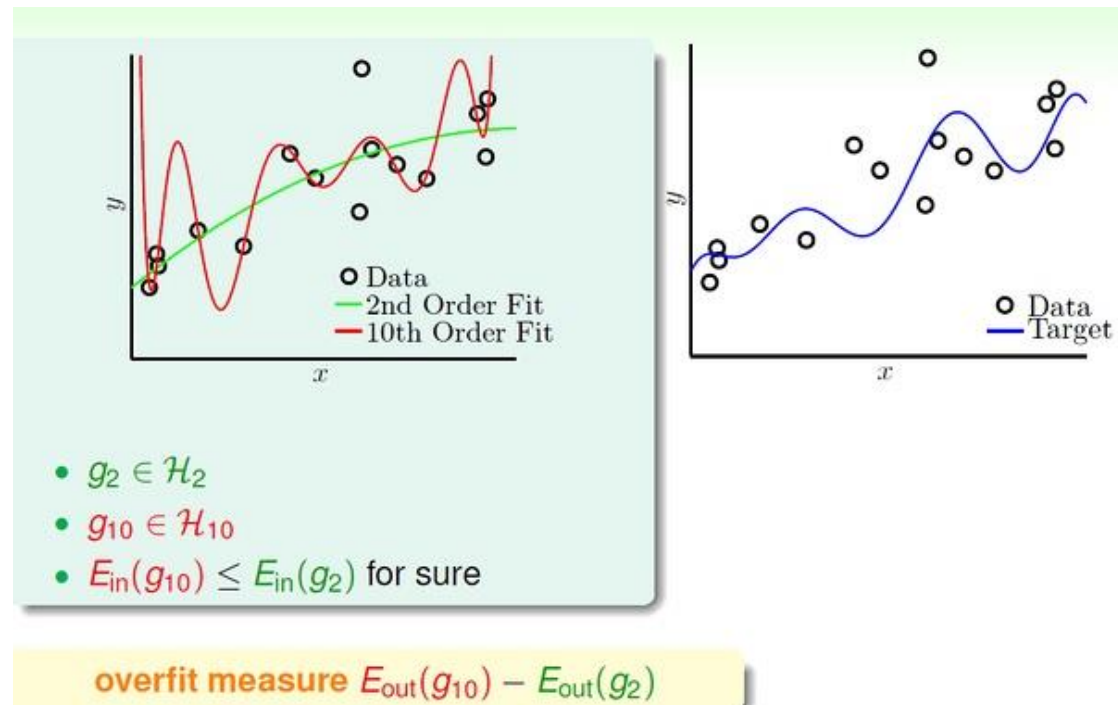
12:   If  $n \geq N$ , build the super graph  $\mathcal{G}_{sup}^c$  by  $C_p(\mathbf{X}_{ij}^{(t)}, \mathbb{X}^{(t)})$  and find a maximum span tree to generate  $\mathbb{X}_c^*$ ; Otherwise, perform the smoothing method [30] to obtain  $\mathbb{X}_c^*$ ;

13: **end if**

$$k^* = \arg \max_{k=1}^N J(\mathbf{X}_{ik}^{(t-1)} \mathbf{X}_{kj}^{(t-1)})$$

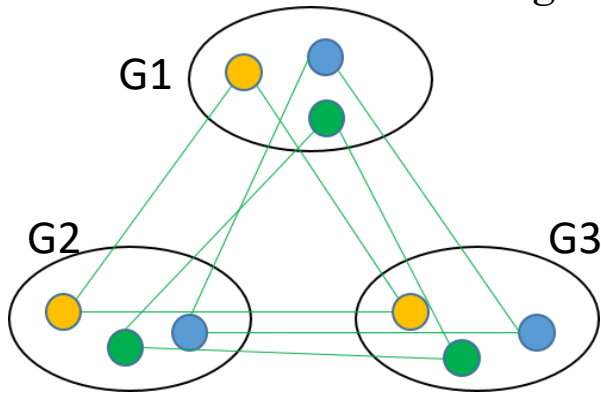
# Recall over fitting in machine learning

- Affinity score (noise, bias)  $\leftrightarrow$  Empirical term (noise, bias)
- Consistency score  $\leftrightarrow$  Regularization term

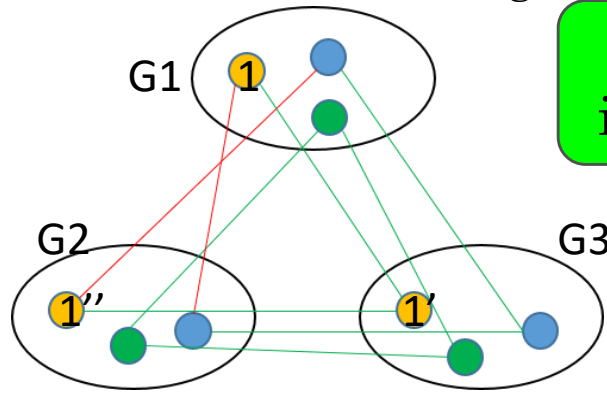


# Cycle consistency as regularizer

Consistent matchings

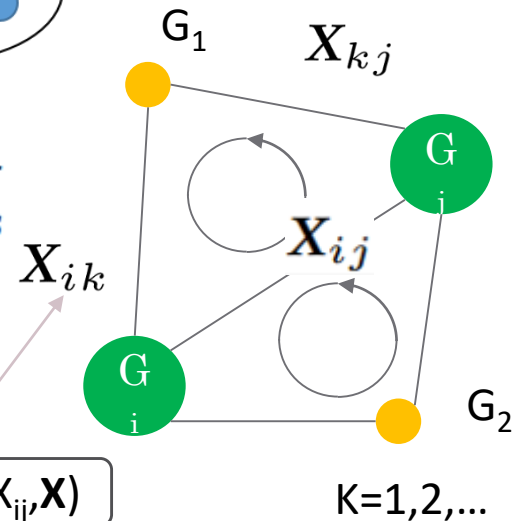


Inconsistent matchings



Consistency implies accuracy

**Definition 2.** Given graphs  $\{\mathcal{G}_k\}_{k=1}^N$  and matching configuration  $\mathbb{X}$ , for any pair  $\mathcal{G}_i$  and  $\mathcal{G}_j$ , the pairwise consistency is defined as  $C_p(\mathbf{X}_{ij}, \mathbb{X}) = 1 - \frac{\sum_{k=1}^N \|\mathbf{X}_{ij} - \mathbf{X}_{ik}\mathbf{X}_{kj}\|_F / 2}{nN} \in (0, 1]$ .



Decide the updating order of  $X_{ur}$  in ascending order of  $C_p(X_{ij}, \mathbf{X})$

$$C_p(\mathbf{X}_{ij}, \mathbb{X}) = 1 - \frac{\sum_{k=1}^N \|\mathbf{X}_{ij} - \mathbf{X}_{ik}\mathbf{X}_{kj}\|_F / 2}{nN} \in (0, 1]$$

# Gradually consistency-weighted CAO (TPAMI'16)

$$\mathbf{Y}_{ikj}^{(t-1)} = \mathbf{X}_{ik}^{(t-1)} \mathbf{X}_{kj}^{(t-1)}$$

$$k^* = \arg \max_{k=1}^N (1 - \lambda) \underbrace{J(\mathbf{Y}_{ikj}^{(t-1)})}_{\text{Affinity term}} + \lambda \underbrace{C_p(\mathbf{Y}_{ikj}^{(t-1)}, \mathbb{X}^{(t-1)})}_{\text{Consistency score}}$$

Affinity term

Consistency score

---

**Algorithm 2** Composition based Affinity Optimization via Graduated Consistency Regularization CAO-C

Consistency

**Require:**  $\{\mathbf{K}_{ij}\}_{i=1, j=i+1}^{N-1, N}$ ;  $T$ ,  $\lambda^{(T_0)} = \lambda^{(0)}$ ,  $\{\lambda^{(t)}\}_{t=1}^{T_0-1} = 0$ ,  $\gamma$ ;

1: Perform pairwise matching to obtain initial  $\mathbb{X}^{(0)}$ ;

2: Calculate  $J^{(0)} = \sum_{i=1, j=i+1}^{N-1, N} \text{vec}(\mathbf{X}_{ij}^{(0)})^T \mathbf{K}_{ij} \text{vec}(\mathbf{X}_{ij}^{(0)})$ ;

3: **for**  $t = 1 : T$  **do**

4:   **for all**  $i = 1, 2, \dots, N - 1; j = i + 1, \dots, N$  **do**

5:     update  $\mathbf{X}_{ij}^{(t)} = \mathbf{X}_{ik}^{(t-1)} \mathbf{X}_{kj}^{(t-1)}$  by solving Eq.7;

6:   **end for**

7:   **if**  $t > T_0$ ,  $\lambda^{(t)} = \min(1, \beta \lambda^{(t-1)})$ ;

Gradually added

8: **end for**

9: Perform the same post-processing as in Alg.1 (L9-13).

---

# Outline

- Introduction
- **Background: Learning on Graph Matching**
- Embedding approach for Deep Graph Matching
- Other techniques
- Outlook

# Learning Matching Function

A matching function with learnable weights:

$$g_{\omega}(\mathcal{G}_1, \mathcal{G}_2) = \underset{\pi}{\operatorname{argmax}} \langle \omega, \Phi(\mathcal{G}_1, \mathcal{G}_2, \pi) \rangle$$

learnable weights  
↑  
matching result      node/edge affinities

- Share weights for affinities in  $\Phi(\mathcal{G}_1, \mathcal{G}_2, \pi)$ ;
- $\mathcal{G}_1, \mathcal{G}_2$  are built on labeled images.

Loss function:

$$L = \frac{1}{N} \sum_{i,j} \Delta(g_{\omega}(\mathcal{G}_i, \mathcal{G}_j), y^{gt}) + \lambda \Omega(\omega)$$

empirical term      regularization

# Learning Graph Structure

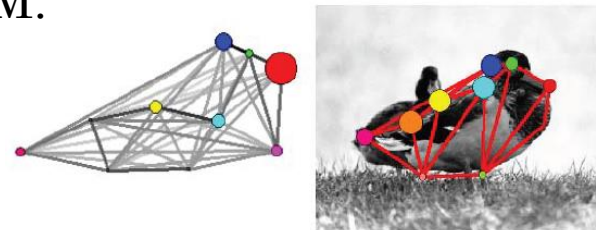
A **reference graph model** ( $\mathcal{G}$ ) is learned for each category of object.

Matching function

$$g_{\omega}(\mathcal{G}, \mathcal{G}') = \underset{\pi}{\operatorname{argmax}} \langle \omega, \Phi(\mathcal{G}, \mathcal{G}', \pi) \rangle$$

reference graph  
↑

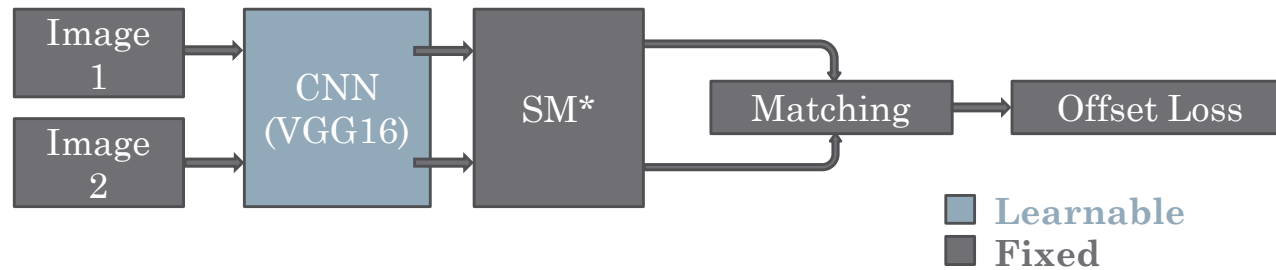
- **Multi-dimensional** weights for  $\Phi(\mathcal{G}, \mathcal{G}', \pi)$ ;
- $a_i, a_{ij}$  of reference graph are learnable;
- Input graph  $\mathcal{G}'$  is matched to learned **graph model**  $\mathcal{G}$ ;
- Graph learning is solved under SSVM.



(c) a learned graph model and its matching

# Learning Matching Features: Deep Learning of Graph Matching

Combination of CNN and graph matching.

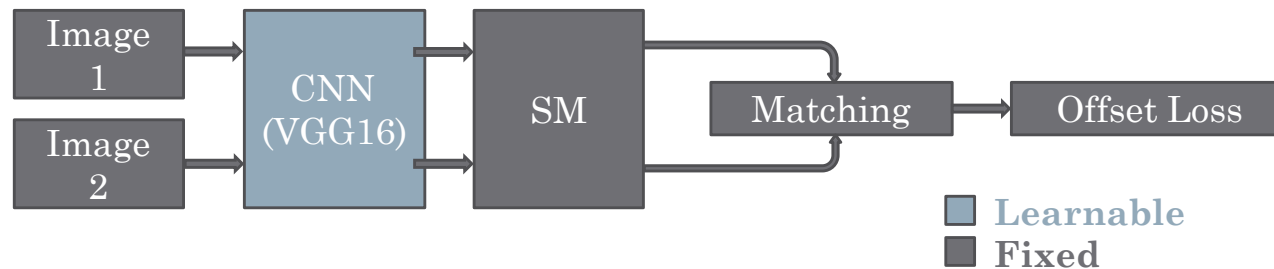


\* Leordeanu and Hebert, "A Spectral Technique for Correspondence Problems Using Pairwise Constraints." ICCV2005.

- **End-to-end** training with back propagation;
- **First formulation** of deep learning in graph matching.

Zanfir and Sminchisescu, "Deep Learning of Graph Matching." CVPR2018.

# Deep Learning of Graph Matching

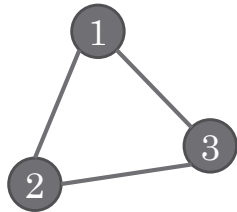


## Limitations

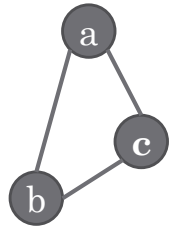
- Performance bottleneck (SM is not a sota GM solver);
- High computational cost;
- Offset loss failed to exploit the combinatorial nature of graph matching;
- Failed to model matching in deep learning (only CNN features are learned).

Zanfir and Sminchisescu, “Deep Learning of Graph Matching.” CVPR2018.

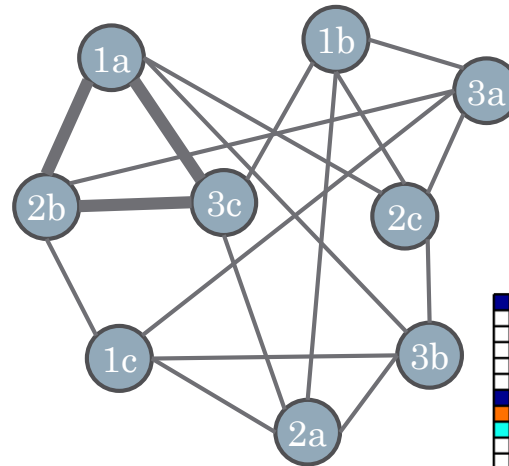
# Spectral Matching (SM)



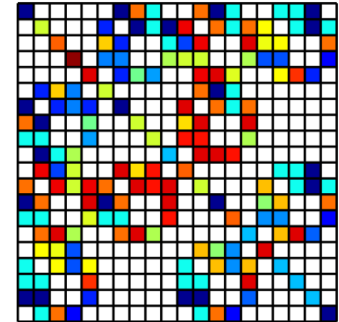
Graph 1



Graph 2



Association Graph



Leordeanu and Hebert, "A Spectral Technique for Correspondence Problems Using Pairwise Constraints." ICCV2005.

# Spectral Matching (SM)

Matching of two graphs:

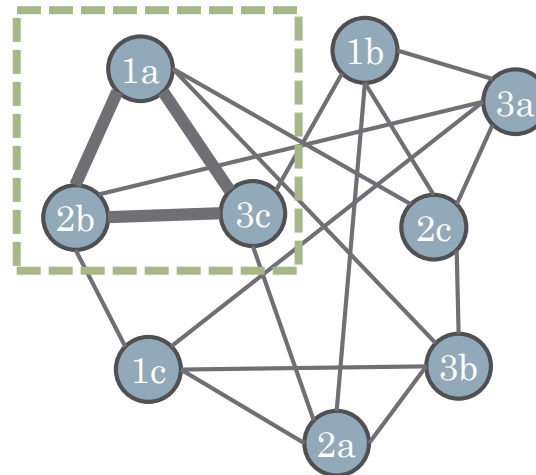
find **clustering** in association graph

(Still NP-hard)

Solved approximately via spectral method:

$$\max_x x^T K x, \quad s.t. \|x\|^2 = 1$$

Equivalent to finding maximum eigenvalue of  $K$ .

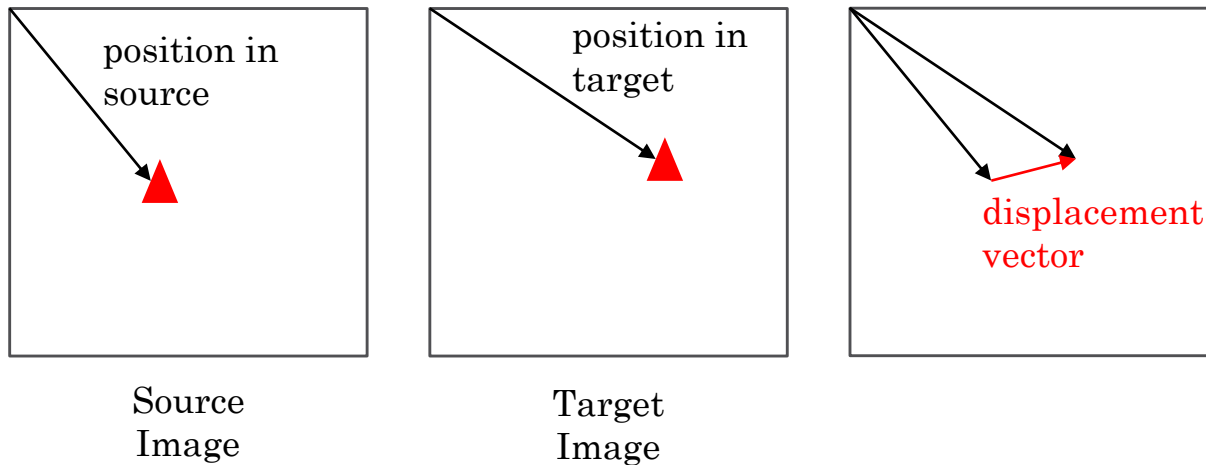


Association Graph

Leordeanu and Hebert, "A Spectral Technique for Correspondence Problems Using Pairwise Constraints." ICCV2005.

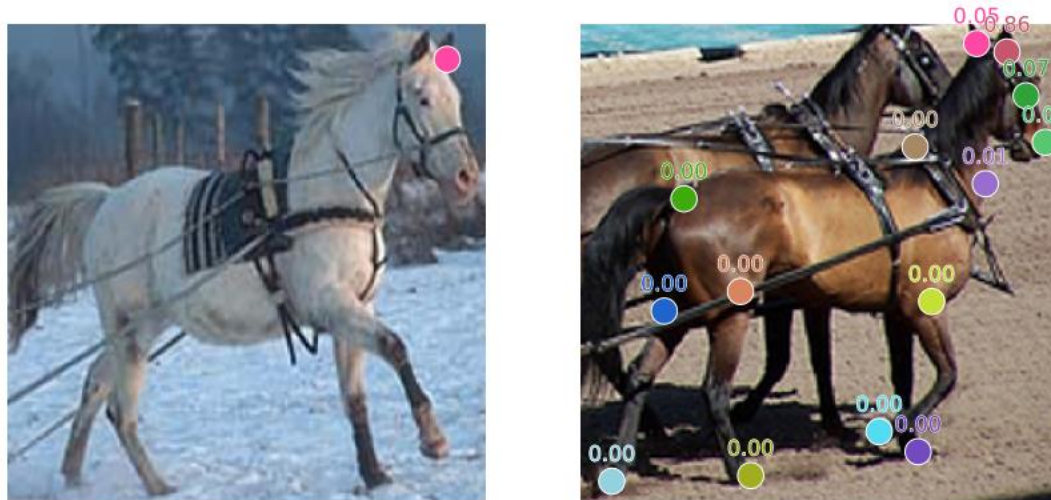
# Offset Loss (regression)

The offset loss adopted, is based on a metric named “**displacement vector**”



The predicted “displacement vector” is computed by weighted votes from all predictions.

# Permutation is **not** regression



$$L_{perm} = 5.139, \quad L_{off} = 0.070$$

Figure 1. Failure case of offset loss: source image (left) and target image (right) with matching candidates, where numbers denote the probability of predicted matching. The two corresponding nodes are colored in rose (only receives 0.05 probability by this poor prediction). Offset loss is computed by a weighted sum among all candidates, resulting in a misleading low loss 0.070. Our permutation loss, on the contrary, issues a reasonably high loss 5.139.

# Outline

- Introduction
- Background: Learning on Graph Matching
- **Embedding approach for Deep Graph Matching**
- Other techniques
- Outlook

# Learning Combinatorial Embedding Networks for Deep Graph Matching

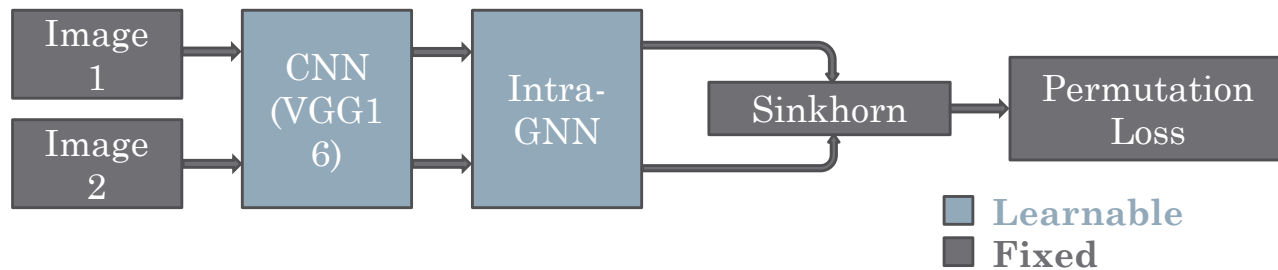
Runzhong Wang, Junchi Yan\* and Xiaokang Yang

ICCV'19 Oral/IEEE TPAMI 2020 <https://arxiv.org/abs/1904.00597>

Project website: [http://thinklab.sjtu.edu.cn/IPCA\\_GM.html](http://thinklab.sjtu.edu.cn/IPCA_GM.html)

Code: <https://github.com/Thinklab-SJTU/PCA-GM>

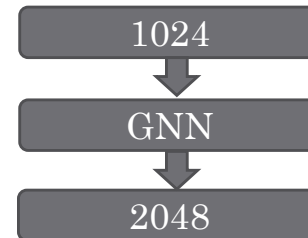
# PIA-GM: Permutation loss & Intra-graph Affinity



- Intra-GNN models graph affinity features;
- Sinkhorn layer solves correspondence efficiently;
- Permutation loss provides strong supervision for combinatorial problem.

CNN feature for each extracted  
feature point in image

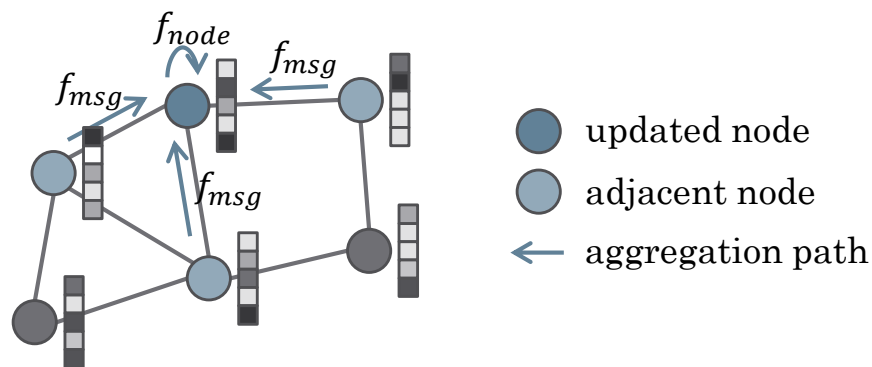
Output vector for each feature  
point



# Intra-GNN

Intra-GNN is mainly inspired by **Graph Convolutional Network (GCN)**

Feature aggregated from adjacent nodes.



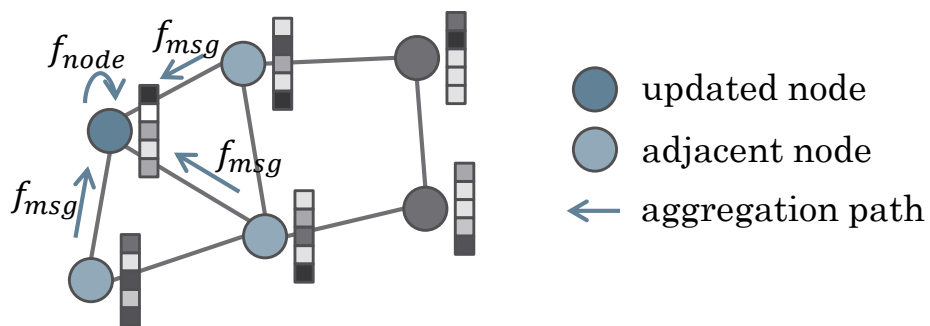
Graph structure  $\rightarrow$  Embedding vectors.

Kipf and Welling, "Semi-Supervised Classification with Graph Convolutional Networks." ICLR 2017.

# Intra-GNN

Intra-GNN is mainly inspired by **Graph Convolutional Network (GCN)**

Feature aggregated from adjacent nodes.



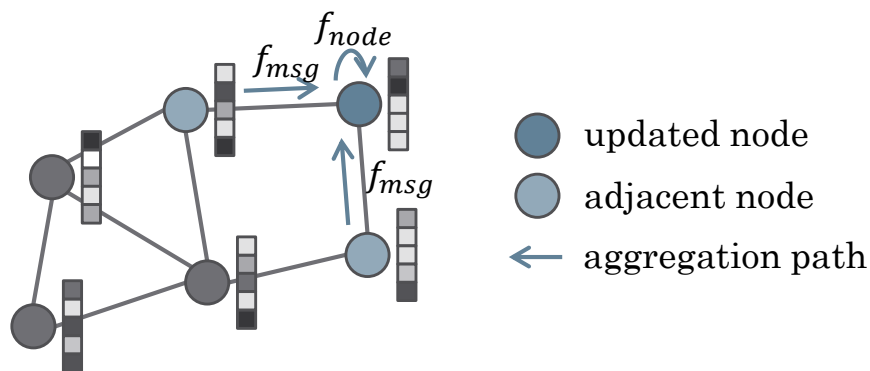
Iterate over all nodes.

Kipf and Welling, "Semi-Supervised Classification with Graph Convolutional Networks." ICLR 2017.

# Intra-GNN

Intra-GNN is mainly inspired by **Graph Convolutional Network (GCN)**

Feature aggregated from adjacent nodes.



$f_{node}$  and  $f_{msg}$  are neural networks and weight-sharing among all nodes.

```
// intra-graph aggregation Eq. (5, 6, 7)
for  $k \leftarrow \{1, 2, 3\}$  do
   $\{\mathbf{h}_{si}^{(k)}\} \leftarrow \text{GConv}_k(\mathbf{A}_s, \{\mathbf{h}_{si}^{(k-1)}\});$ 
```

Kipf and Welling, “Semi-Supervised Classification with Graph Convolutional Networks.” ICLR 2017.

# Distance Metric

The similarity (distance) between two vectors from the feature space of the output of GNN, is learned.

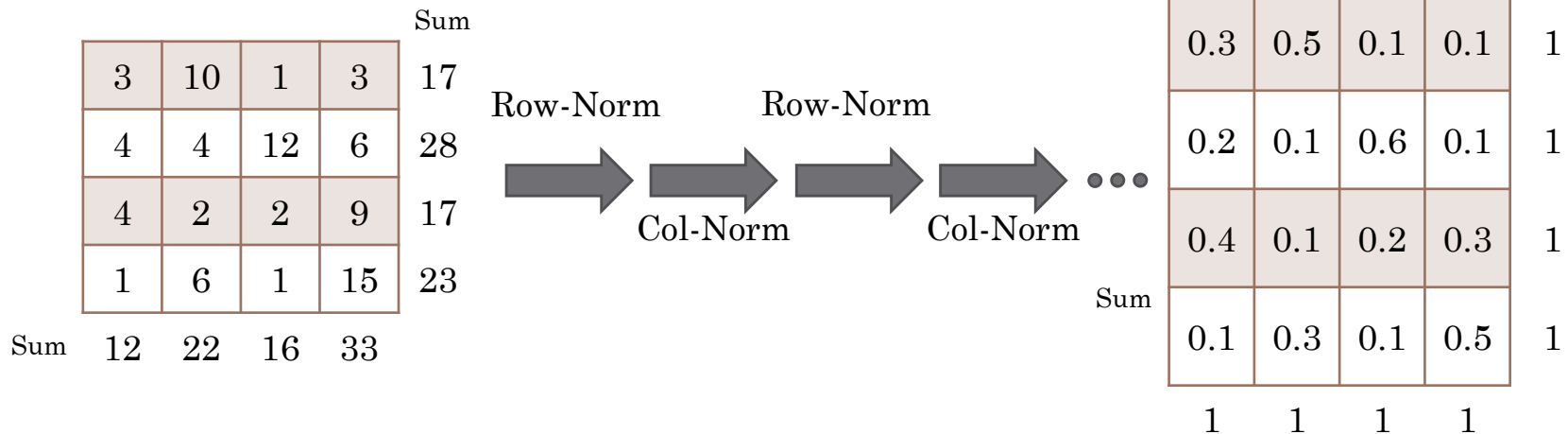
$$s(x_i, x_j) = \exp(x_i^T \mathbf{A} x_j)$$

Where  $\mathbf{A}$  is the matrix with learnable weights.

**Exponential function enforces** non-negativity of  $s$  (required by the later Sinkhorn step).

# Sinkhorn Algorithm

Non-negative matrix  $\rightarrow$  Doubly stochastic matrix

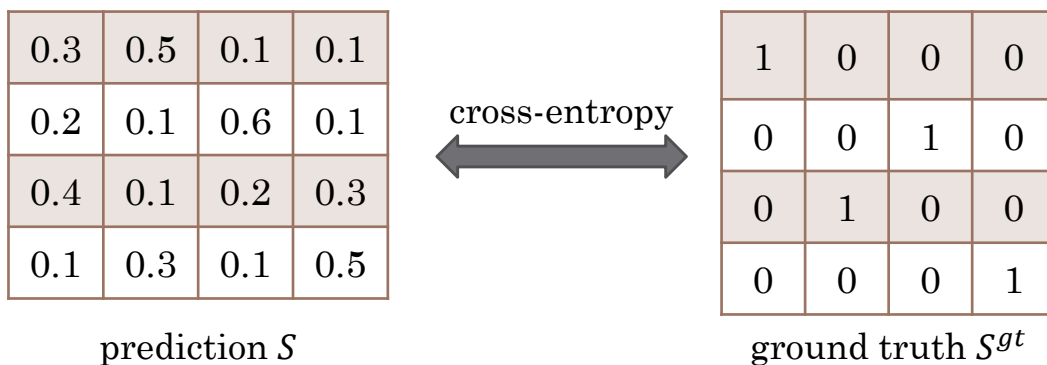


Sinkhorn solves Linear Assignment Problem (LAP) in  $O(knm)$ , and is capable for back propagation.

- $k$ : number of iterations;  $n \times m$ : matrix size

Sinkhorn and Knopp, “Concerning Nonnegative Matrices and Doubly Stochastic Matrices.”

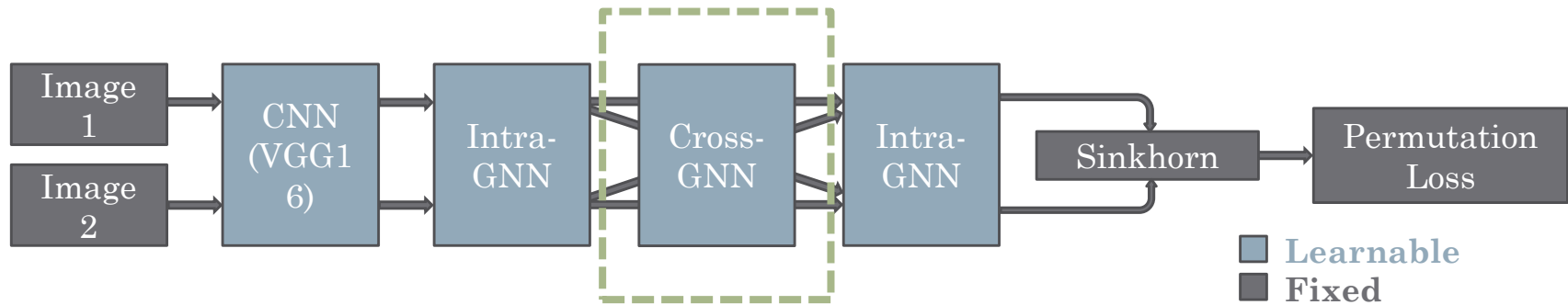
# Permutation Loss



$$L_{perm} = \frac{1}{N} \sum_{ij} [S_{ij}^{gt} \log S_{ij} + (1 - S_{ij}^{gt}) \log(1 - S_{ij})]$$

Permutation loss is the **inherent choice** of combinatorial problems like graph matching.

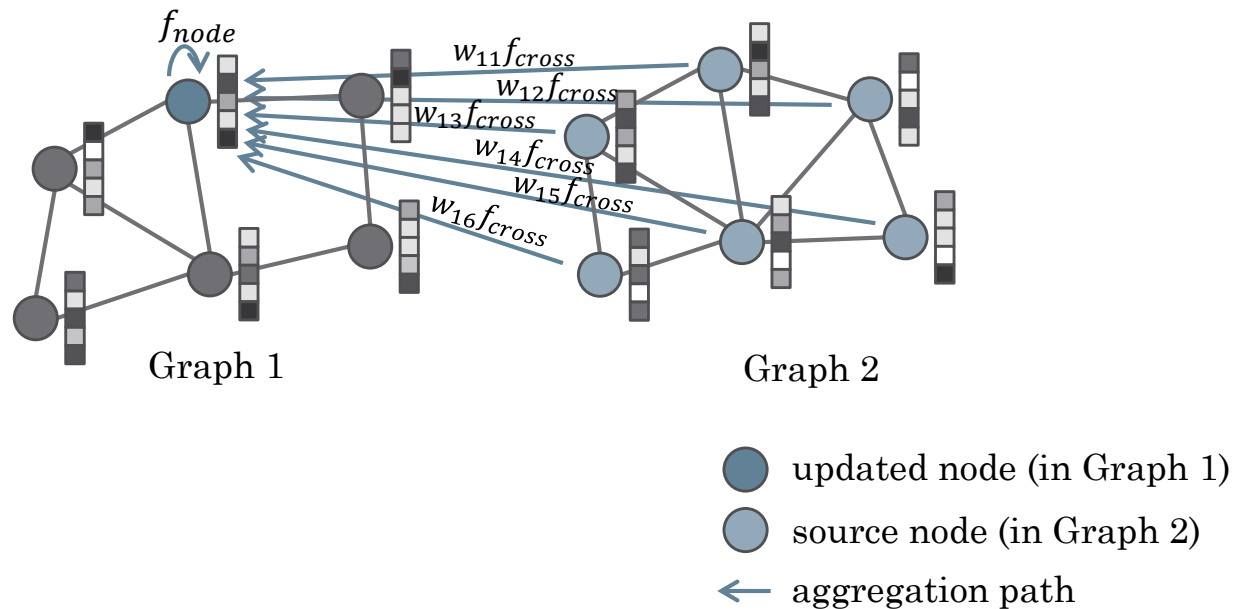
# PCA-GM: Permutation loss & Cross-graph Affinity



**Cross-graph GNN** is adopted to aggregate features across graphs.

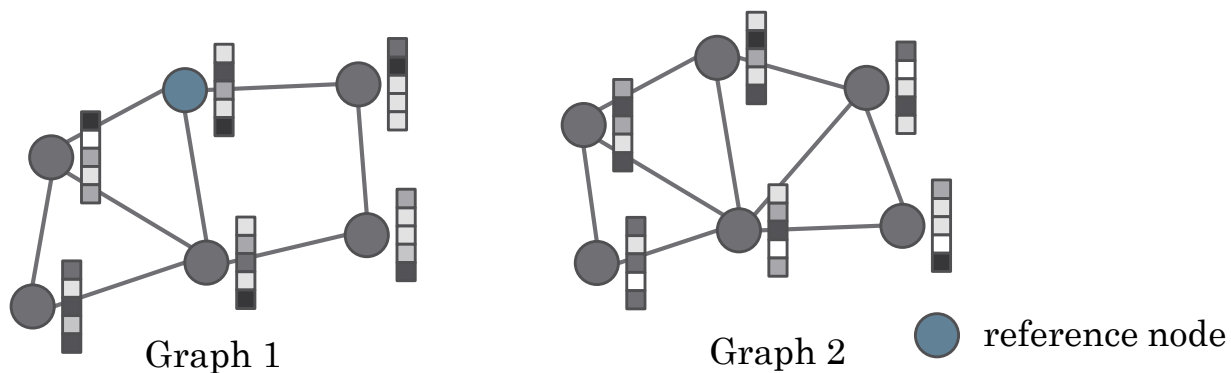
# Cross-GNN

Features are aggregated from nodes with similar features across graphs.



# Similarity Prediction

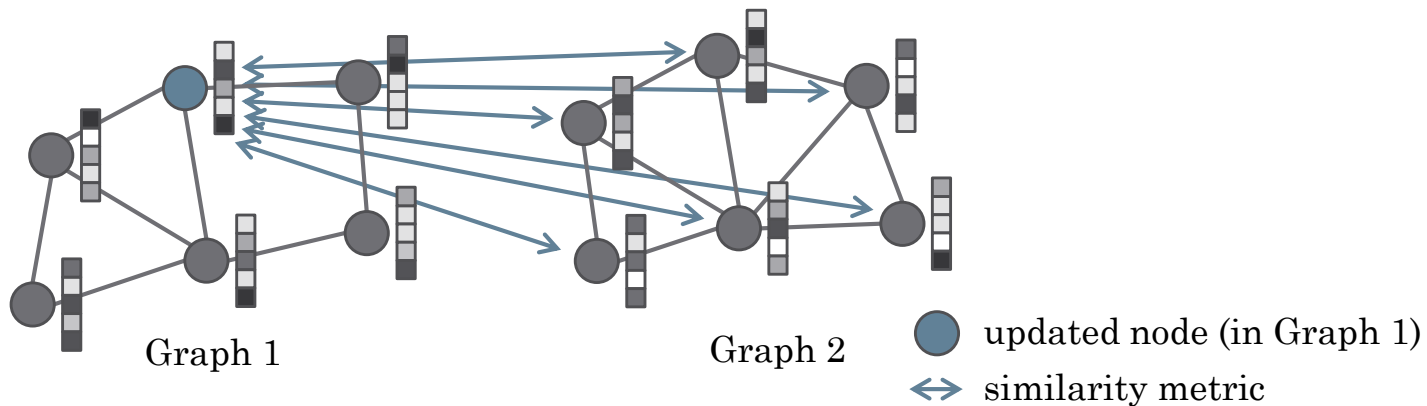
The cross-graph aggregation path is defined by cross-graph similarity, computed from feature vectors from shallower layers.



Given two graphs to be matched. Consider each node in Graph 1.

# Similarity Prediction

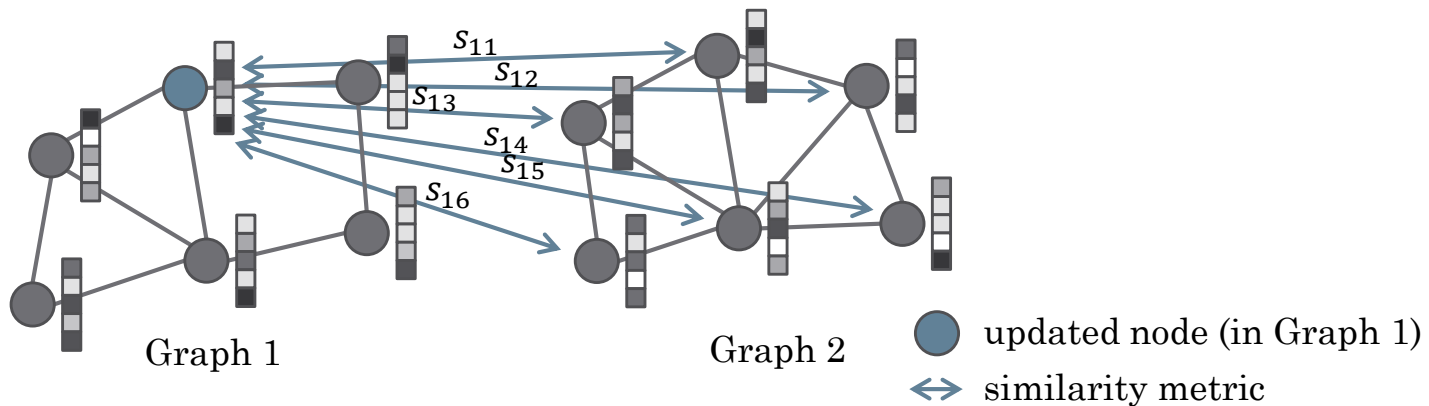
The cross-graph aggregation path is defined by cross-graph similarity, computed from feature vectors from shallower layers.



Evaluate the similarity between the reference node and all nodes in Graph 2.

# Similarity Prediction

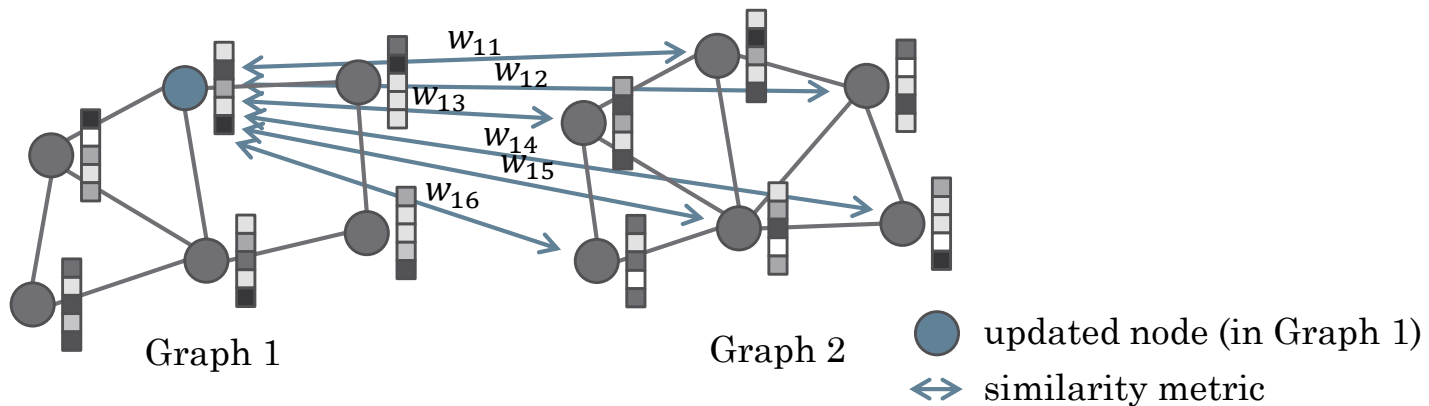
The cross-graph aggregation path is defined by cross-graph similarity, computed from feature vectors from shallower layers.



The similarity scores,  $s_{ij}$ , is stored into  $n \times m$  matrix.

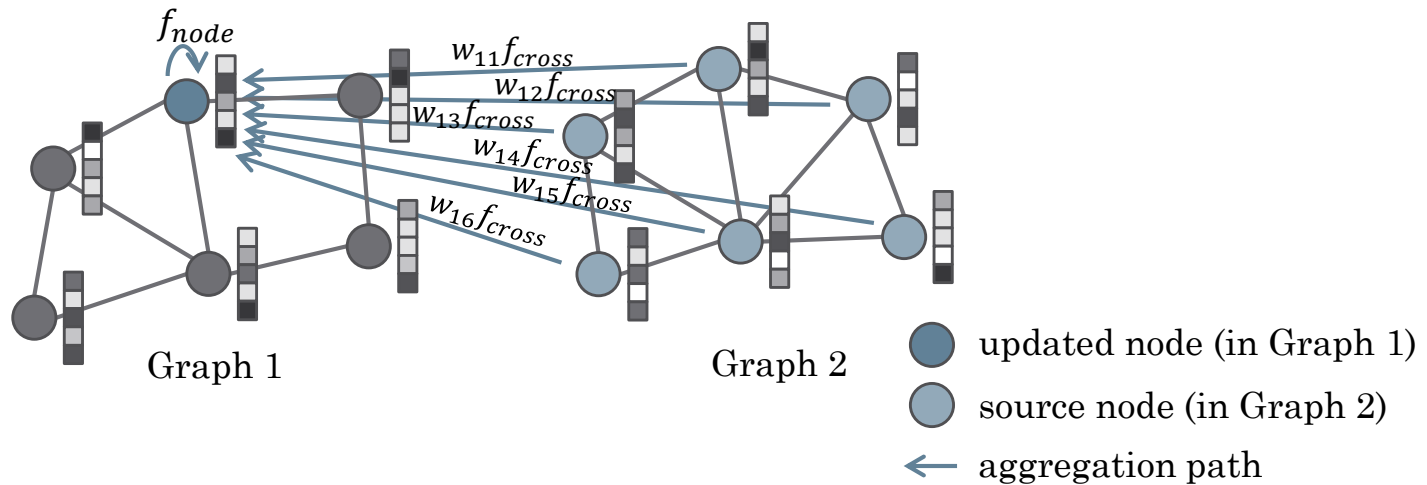
# Similarity Prediction

The cross-graph aggregation path is defined by cross-graph similarity, computed from feature vectors from shallower layers.



Similarity  $s_{ij}$  is transformed into weights  $w_{ij}$  by Sinkhorn.

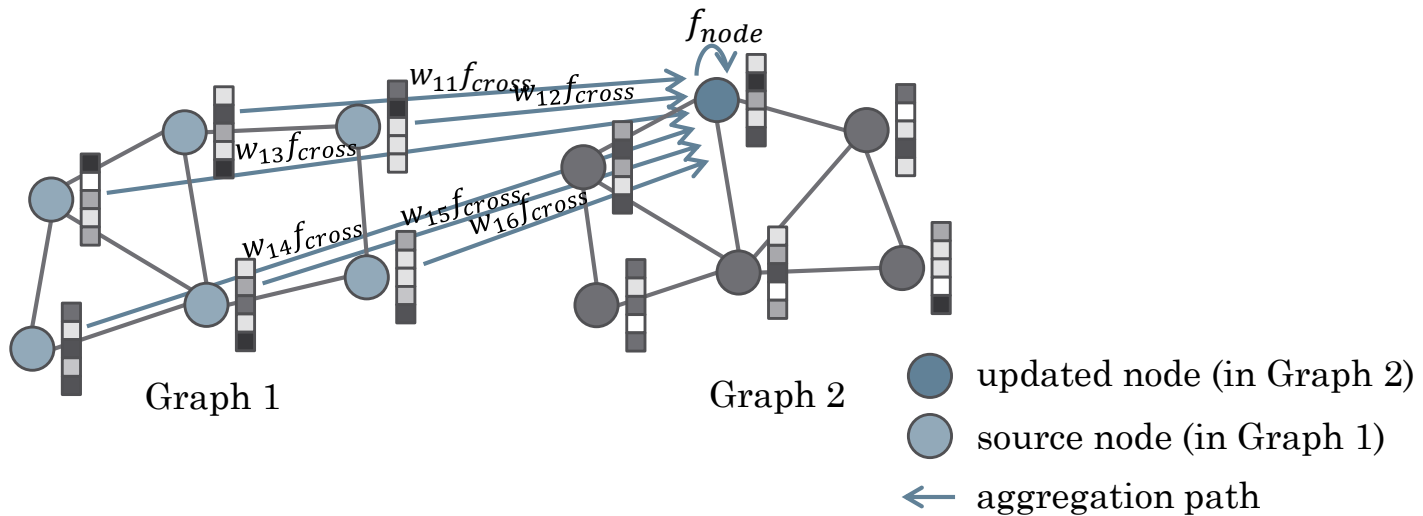
# Cross-graph Aggregation



Features are aggregated through the **weighted predicted path**, in the previous step.

Self-loop is reserved in cross-graph aggregation.

# Cross-graph Aggregation



// first intra-graph aggregation Eq. (5, 6, 7)

$\{\mathbf{h}_{si}^{(1)}\} \leftarrow \text{GConv}_1(\mathbf{A}_s, \{\mathbf{h}_{si}^{(0)}\});$

// similarity prediction Eq. (15, 18)

build  $\hat{\mathbf{M}}^{(0)}$  from  $\{\mathbf{h}_{1i}^{(1)}, \mathbf{h}_{2j}^{(1)}\}$  by Eq. (15);

$\hat{\mathbf{S}} \leftarrow \text{Sinkhorn}(\hat{\mathbf{M}}^{(0)});$

// cross-graph aggregation Eq. (10, 11, 12)

$\{\mathbf{h}_{1i}^{(2)}\} \leftarrow \text{CrossConv}(\hat{\mathbf{S}}, \{\mathbf{h}_{1i}^{(1)}\}, \{\mathbf{h}_{2j}^{(1)}\});$

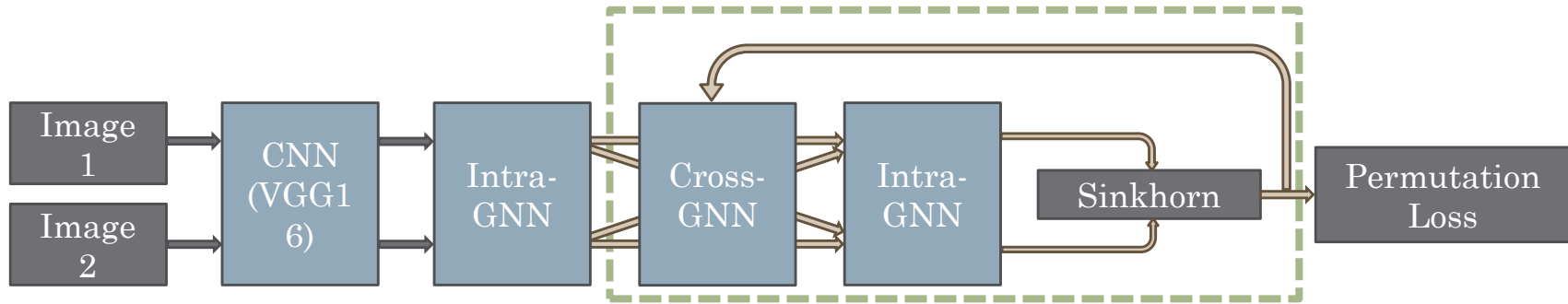
$\{\mathbf{h}_{2j}^{(2)}\} \leftarrow \text{CrossConv}(\hat{\mathbf{S}}^\top, \{\mathbf{h}_{2j}^{(1)}\}, \{\mathbf{h}_{1i}^{(1)}\});$

// second intra-graph aggregation Eq. (5, 6, 7)

$\{\mathbf{h}_{si}^{(3)}\} \leftarrow \text{GConv}_2(\mathbf{A}_s, \{\mathbf{h}_{si}^{(2)}\});$

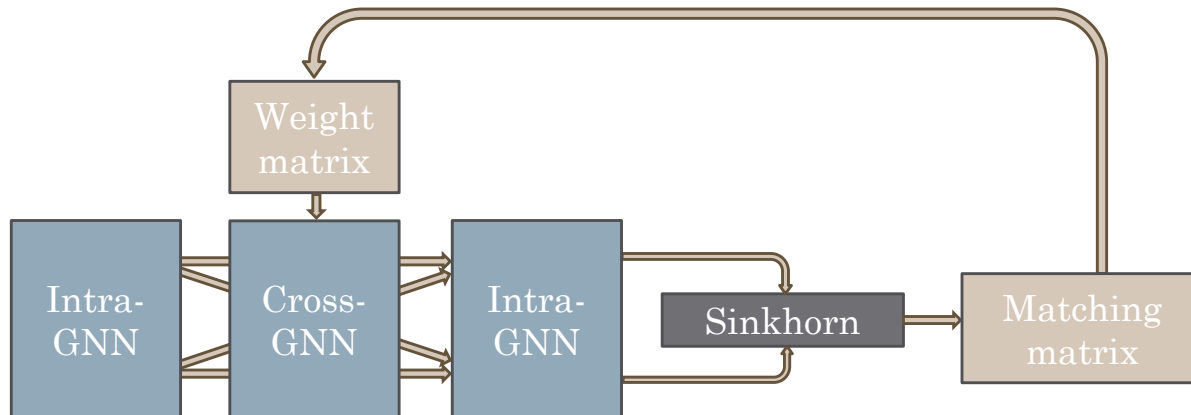
Aggregation steps are repeated for every node in Graph 2.

# IPCA-GM: Permutation loss & Iterative Cross-graph Affinity



Cross-graph GNN is updated **iteratively**.

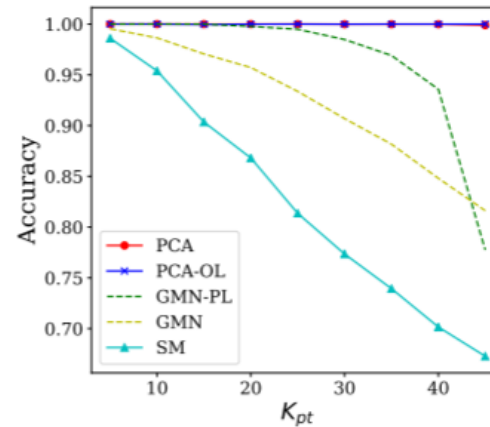
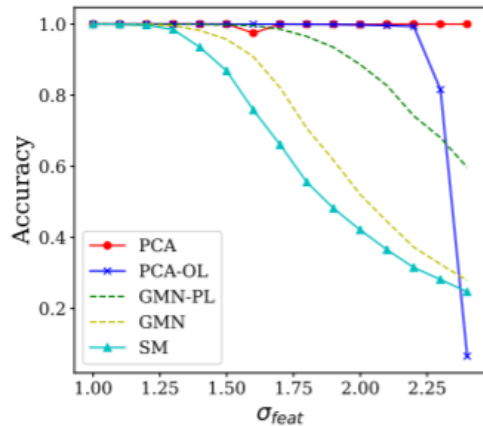
- Learnable
- Fixed
- Iterative Path



# Comparison

	GMN (Zanfir et al. CVPR2018)	PCA-GM (Ours)
Graph Modeling	Performance bottleneck of SM	Deep GNN embedding model with cross-graph aggregation
Matching Solver	High <b>computational cost</b> of SM: $O(km^2n^2)$ time complexity	<b>Sinkhorn</b> algorithm: $O(kmn)$ time complexity
Loss Function	<b>Offset loss:</b> Weaker supervision.	<b>Permutation loss:</b> Stronger supervision and inherent choice of graph matching.
Learned Features	Only CNN features	CNN features, graph affinities

# Result on Synthetic Dataset



- Our method (PCA) is robust to noise.
- Our method (PCA) is robust to the scale of graphs.

# Result on VOC Keypoint

Method	Graph Modeling	Loss Function	Accuracy
GMN (Zanfir et al.)	SM	Offset loss	55.3
GMN-PL	SM	Perm loss	57.9
PIA-OL	Intra-GNN	Offset loss	61.6
PIA	Intra-GNN	Perm loss	63.0
<b>PCA</b>	<b>Cross-GNN</b>	<b>Perm loss</b>	<b>63.8</b>

VOC Keypoint dataset contains 20 categories of instances, labeled with keypoint coordinates.

Our method outperforms deep learning peer method GMN (Zanfir et al. CVPR 2018).

# Result on WillowObject

Method	face	m-bike	car	duck	w-bottle
HARG-SSVM (Cho et al.)	91.2	44.4	58.4	55.2	66.6
GMN trained on VOC	98.1	65.0	72.9	74.3	70.5
GMN finetuned on Willow	99.3	71.4	74.3	82.8	76.7
PCA trained on VOC	100.0	69.8	78.6	82.4	95.1
PCA finetuned on Willow	100.0	76.7	84.0	93.5	96.9

Our method outperformed non-deep learning method HARG-SSVM (Cho et al. ICCV 2013).

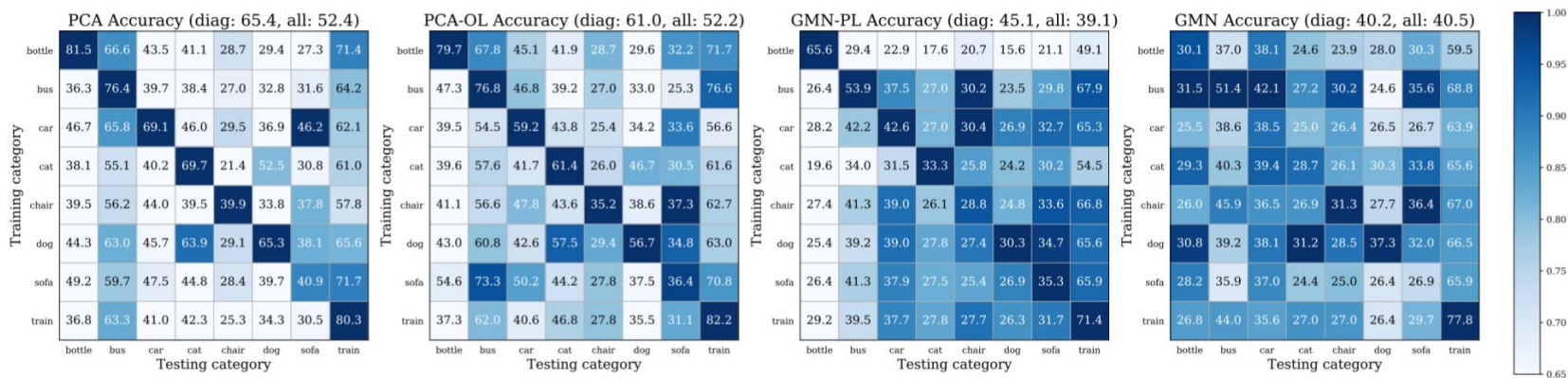
It also shows that the learned knowledge can be efficiently transformed among different datasets.

# Ablation Study

VGG16 feature	Intra-graph embedding	Cross-graph embedding	Distance metric	Accuracy
✓	✓	✓	✓	63.8
✓	✓	✓	×	63.6
✓	✓	×	×	62.1
✓	×	×	×	54.8
×	×	×	×	41.9

Ablation study is performed on VOC Keypoint dataset.

# Generability Study



Confusion matrix on 8 categories of VOC Keypoint dataset.

Result shows PCA learns generable knowledge among similar categories (like cat and dog).

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
GMN	31.9	47.2	51.9	40.8	68.7	72.2	53.6	52.8	34.6	48.6	72.3	47.7	54.8	51.0	38.6	75.1	49.5	45.0	83.0	86.3	55.3
GMN-PL	31.1	46.2	58.2	45.9	70.6	76.4	61.2	61.7	<b>35.5</b>	53.7	58.9	57.5	56.9	49.3	34.1	77.5	57.1	53.6	83.2	88.6	57.9
PIA-OL	39.7	<b>57.7</b>	58.6	47.2	74.0	74.5	62.1	66.6	33.6	61.7	<b>65.4</b>	58.0	67.1	58.9	41.9	77.7	64.7	50.5	81.8	89.9	61.6
PIA	<b>41.5</b>	55.8	60.9	<b>51.9</b>	75.0	75.8	59.6	65.2	33.3	<b>65.9</b>	62.8	<b>62.7</b>	67.7	62.1	42.9	<b>80.2</b>	64.3	<b>59.5</b>	82.7	90.1	63.0
PCA	40.9	55.0	<b>65.8</b>	47.9	<b>76.9</b>	<b>77.9</b>	<b>63.5</b>	<b>67.4</b>	33.7	65.5	63.6	61.3	<b>68.9</b>	<b>62.8</b>	<b>44.9</b>	77.5	<b>67.4</b>	57.5	<b>86.7</b>	<b>90.9</b>	<b>63.8</b>

Table 2. Accuracy on Pascal VOC Keypoint. Note after replacing the offset loss by the permutation loss, the GMN-PL outperforms GMN [51] almost in all categories. While our method PIA’s performance degenerates when its permutation loss is changed to offset loss.

# Neural Graph Matching Network: Learning Lawler's Quadratic Assignment Problem with Extension to Hypergraph and Multiple-graph Matching

Runzhong Wang, Junchi Yan\* and Xiaokang Yang

Accepted to IEEE TPAMI 2021

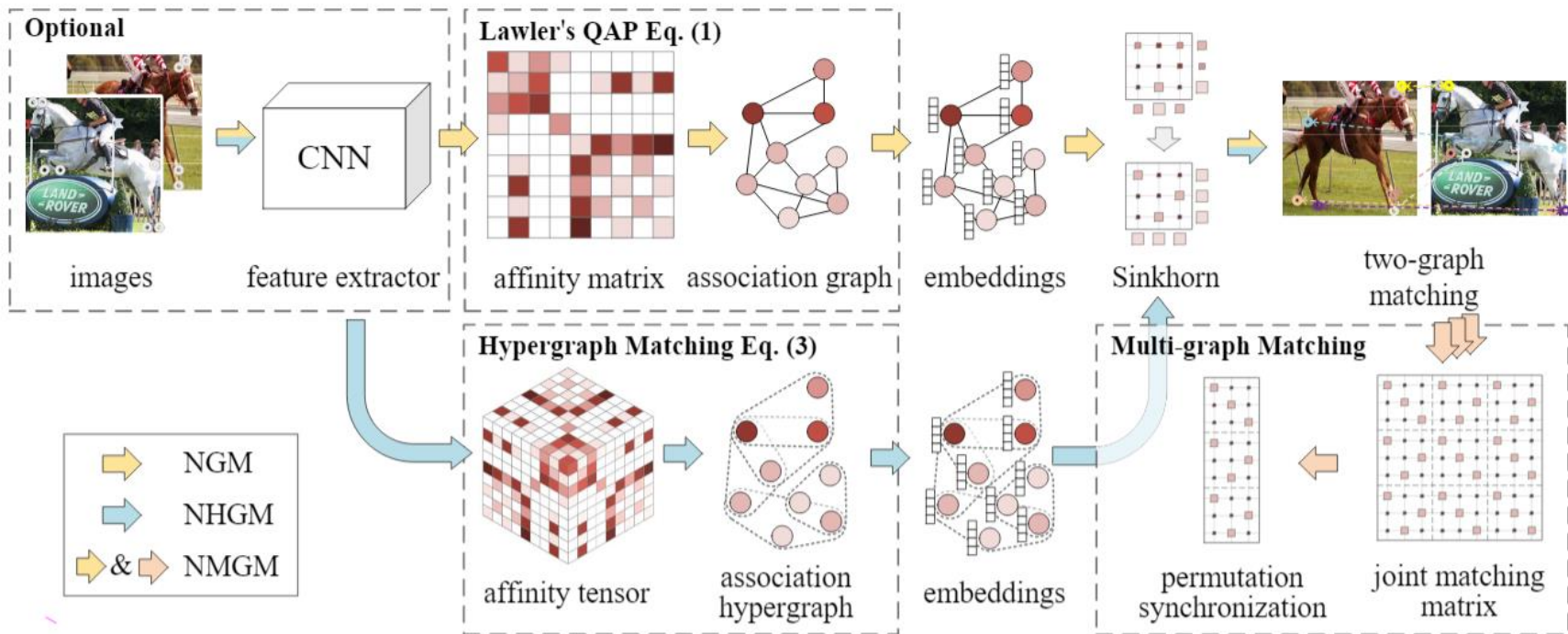
<https://arxiv.org/abs/1911.11308>

# Overview

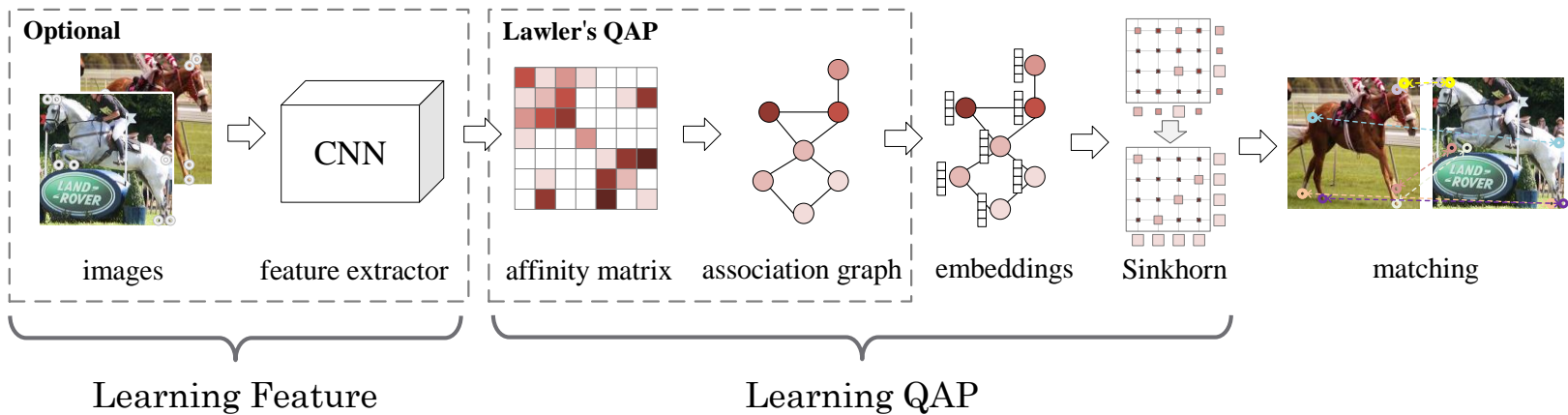
Our pipeline includes **end-to-end** learning of:

- Neural Graph Matching (NGM)
- Neural Hyper-Graph Matching (NHGM)
- Neural Multi-Graph Matching (NMGM)

Especially, NGM is the **first learnable solver** for QAP



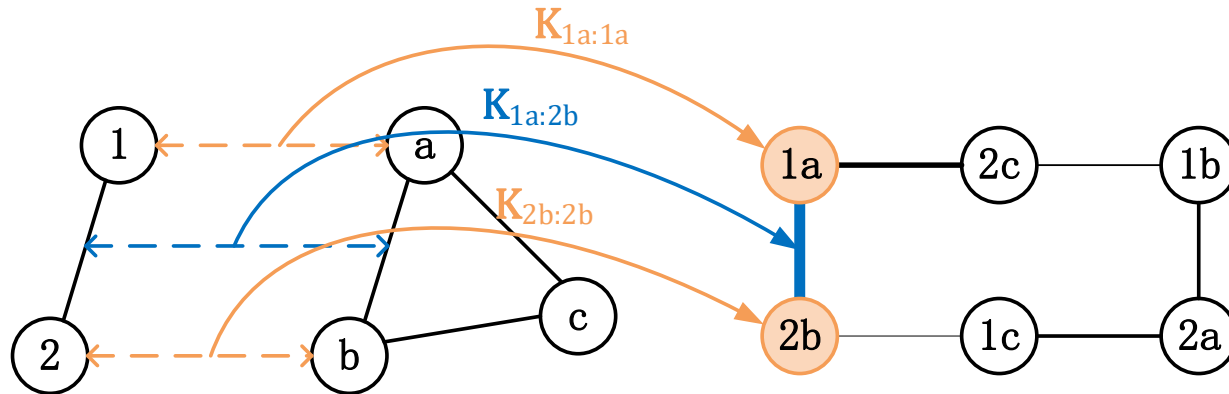
# Neural Graph Matching



- Joint learning on **CNN feature** and **QAP solver**
- Problem solved effectively via **deep embedding** on **association graph**

# Association Graph

- Graph Matching Problem  $\rightarrow$  Classification on Association Graph

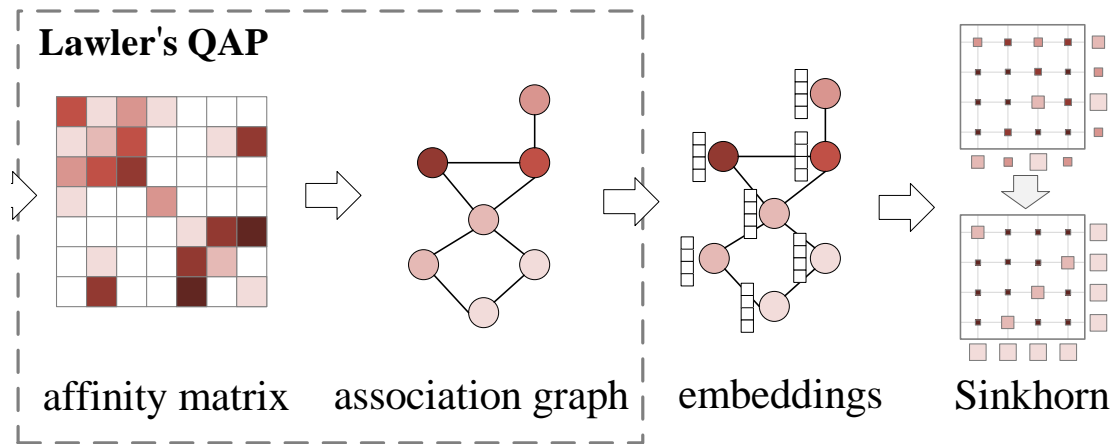


- Node-node affinity  $\rightarrow$  node in association graph
- Edge-edge affinity  $\rightarrow$  edge in association graph

M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in ICCV, 2005

# Embedding based Classifier (QAP Solver)

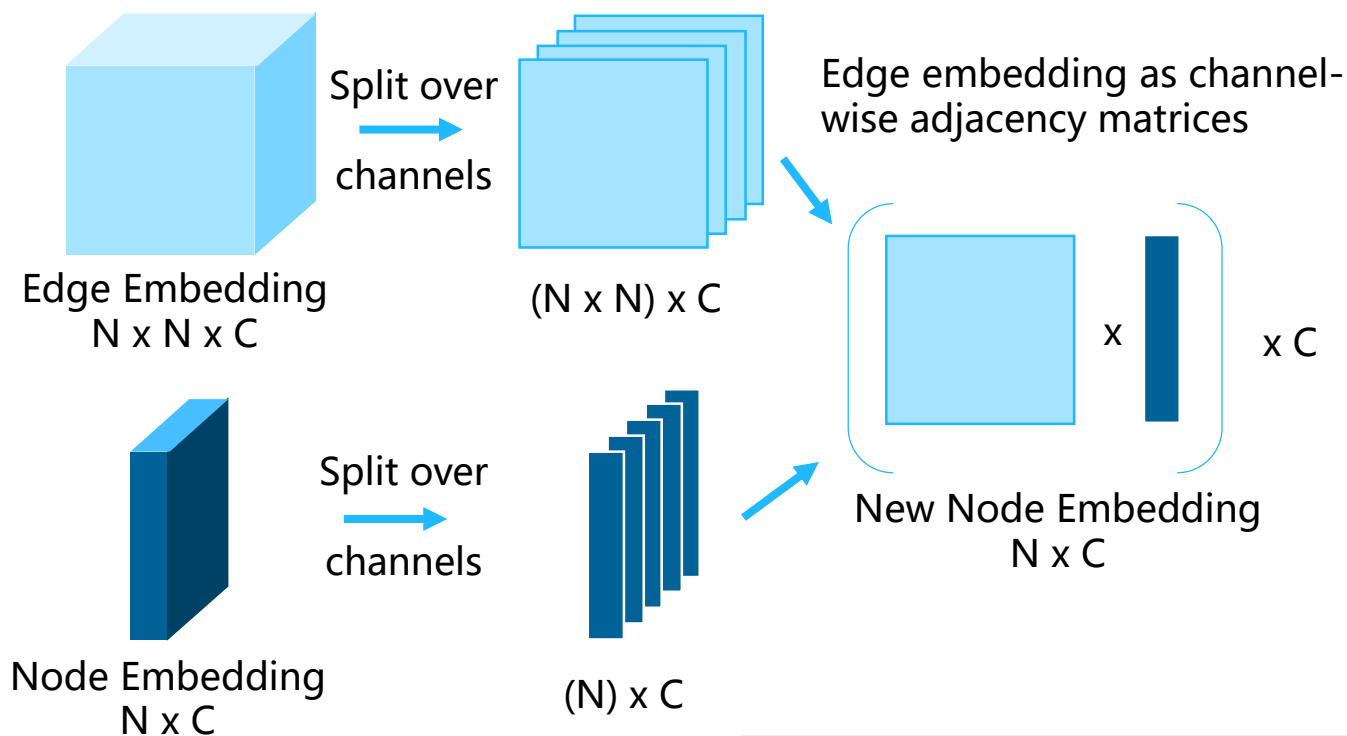
- Efficient node embedding on the association graph



- Affinity matrix  $\rightarrow$  Matching matrix
- We adopt vanilla GCN for its effectiveness and simplicity:

$$\mathbf{m}^{(k)} = \mathbf{A}'\mathbf{W}f_m(\mathbf{v}^{(k-1)}) + f_v(\mathbf{v}^{k-1}), \quad \mathbf{v}^{(k)} = \mathbf{m}^{(k)} \quad (6)$$

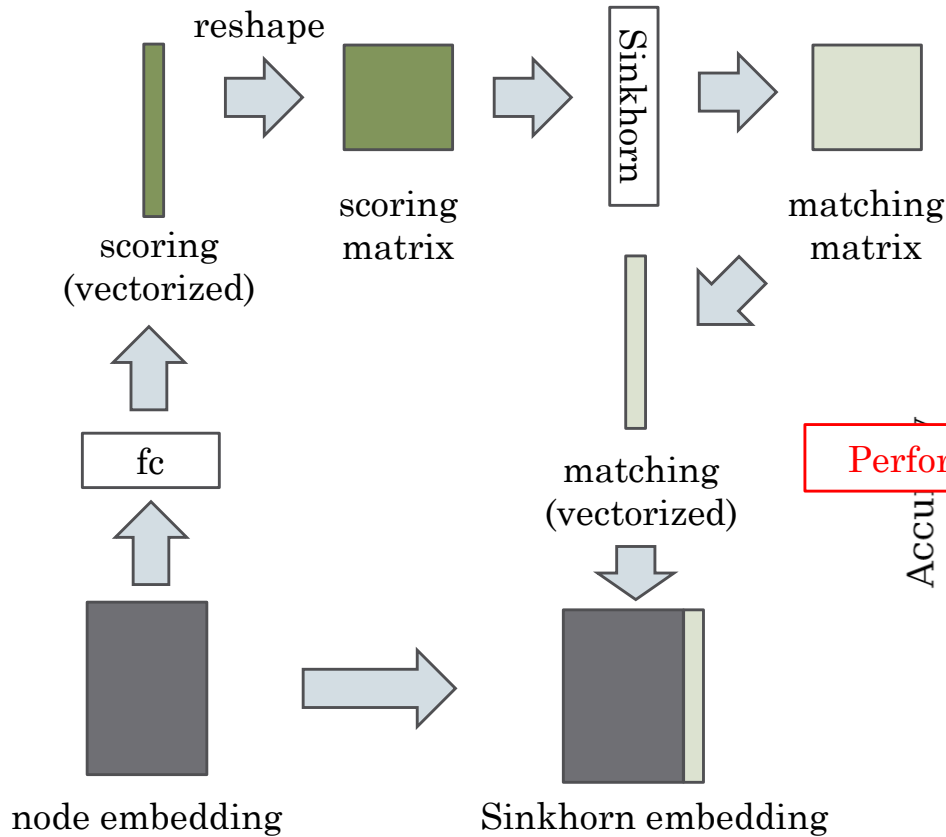
# Channel-wise Edge Embedding



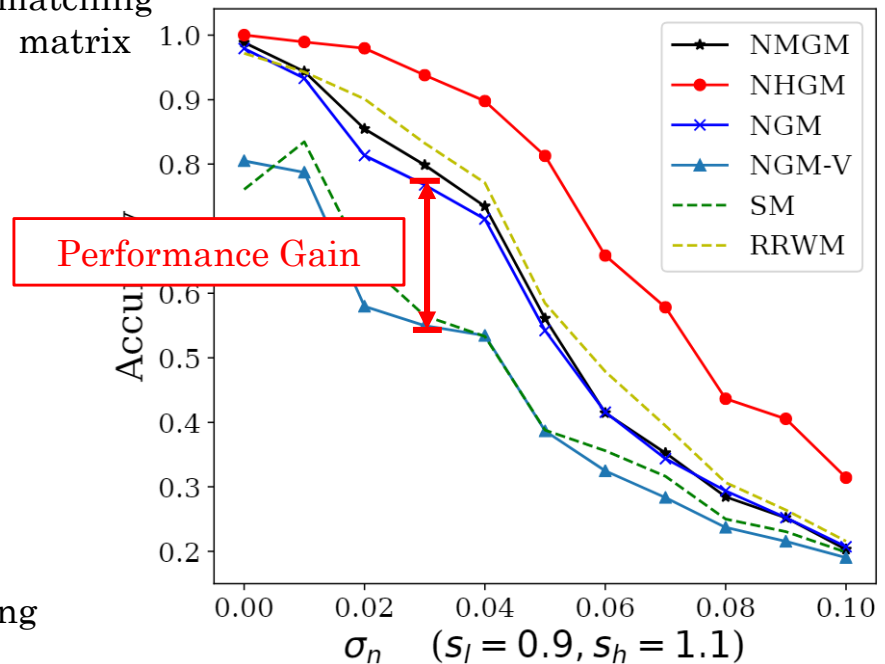
edge embedding	EGNN(C) [14]	HyperConv [15]	Ours Eq. (12)
accuracy	53.3	55.9	66.1

Further improvement can be achieved, namely NGM+.

# Sinkhorn Embedding



Enforcing matching constraint during embedding



model	baseline NGM	+ node affinity	+ Sinkhorn embedding
accuracy (rel.)	58.7	59.6 (+0.9)	64.1 (+4.5)

# Extension to Hyper-Graph Matching

Neural Graph Matching  
(NGM)



Neural Hyper Graph Matching  
(NHGM)

1<sup>st</sup>, 2<sup>nd</sup> features



**higher order** features

association graph

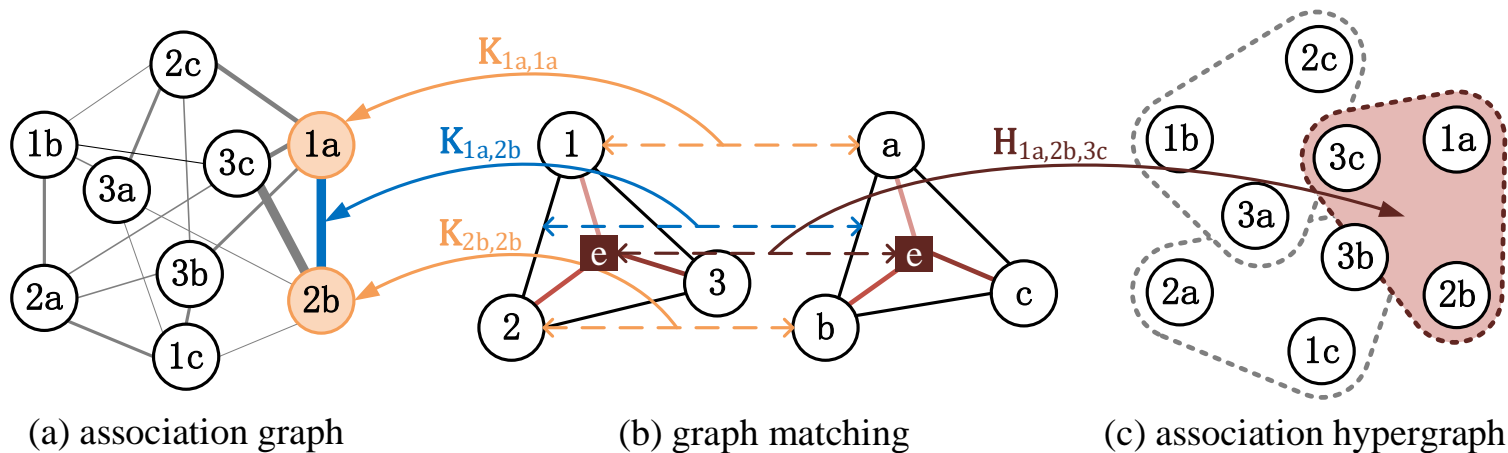


association **hypergraph**

aggregation along edge



aggregation along **hyperedge**

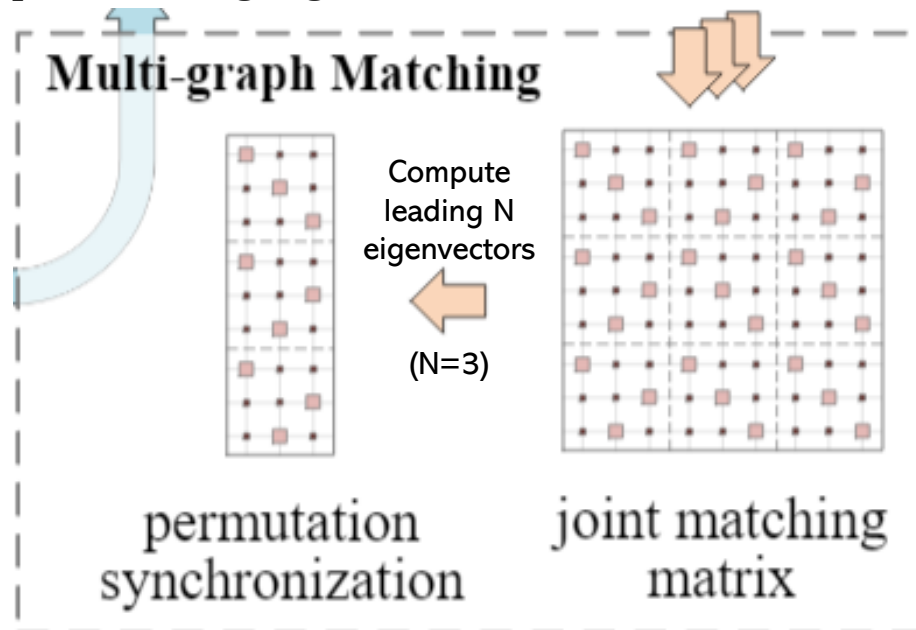


# Extension to Multi-Graph Matching

- End-to-end Multi-Graph Matching via Permutation Synchronization:

D. Pachauriy, R. Kondorx, V. Singh, Solving the multi-way matching problem by permutation synchronization, in NIPS 2013

- 1) Get pairwise matching result by NGM
- 2) Build joint matching matrix
- 3) Compute leading eigenvectors



# Synthetic Point Registration

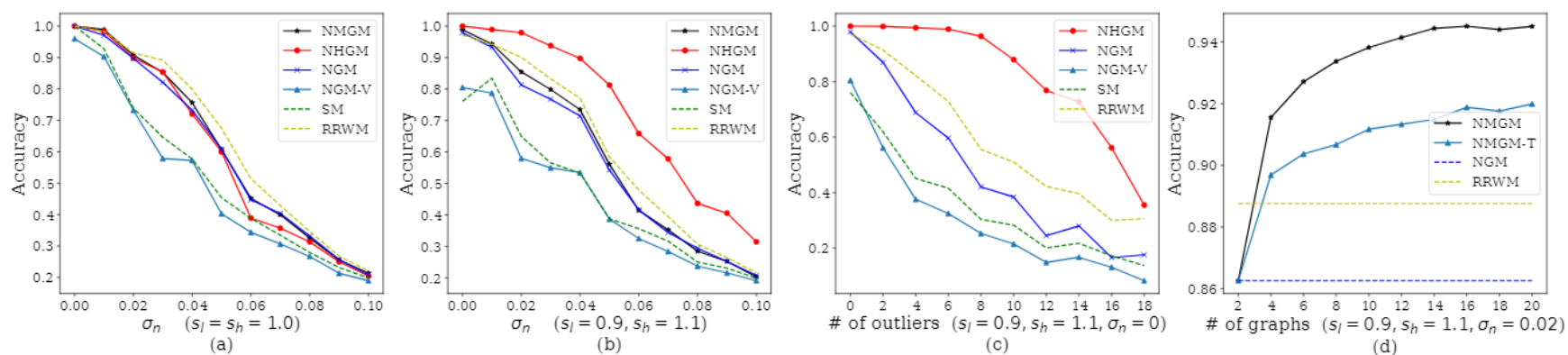
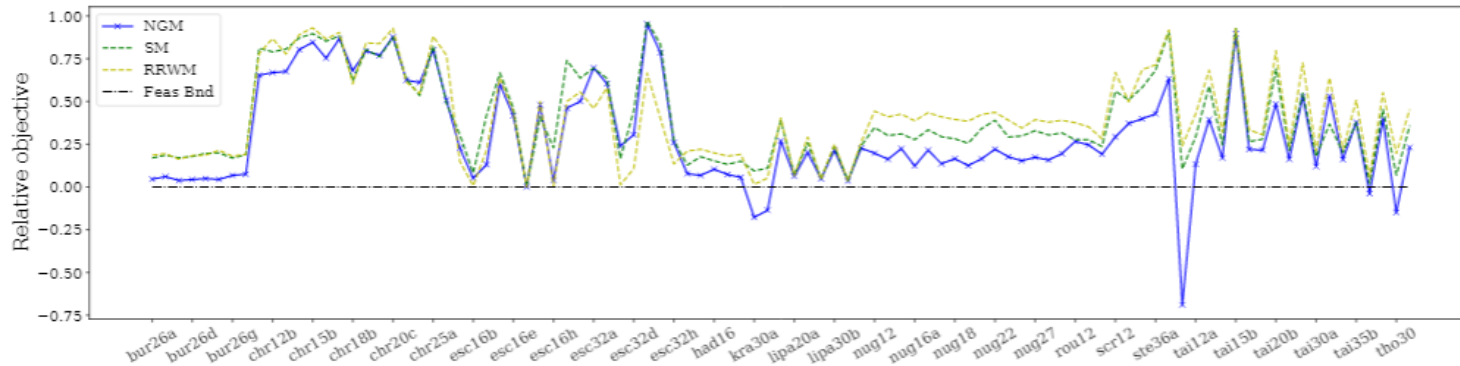


Figure 3. Synthetic test by varying deformation level  $\sigma_n$ , number of outliers and graphs. This test involves purely learning a combinatorial solver for QAP, rather than any affinity model. This feature is not supported in GMN [39] and PCA-GM [32] thus they cannot be compared.

- (a) NMG & NHGM performs comparatively against powerful RRWM
- (b)(c) NHGM shows its robustness against deformation and outliers
- (d) NMGM gains steadily w.r.t. number of graphs.

# QAPLIB Benchmark



(a) Relative objective score. Not all instance labels are shown due to limited space.

NGM outperforms popular SM and RRWM on most instances

The learned solver generalizes soundly to unseen QAP instances

bur26a	5745398	5882	123120	4406	227404	1439332	327610
had18	5953371	5860	127730	4962	274630	1483226	350352
kra30a	5801101	5894	125410	5086	206570	1422984	330654
nug25	5945588	5998	137840	4480	180220	1428212	334760
scr20	5873473	5778	138020	4966	178056	1427996	331832
tai25a	5847762	5868	125850	4938	239580	1369866	346560
tho40	5895581	5944	127910	5190	235470	1484768	303772
	bur26a	had18	kra30a	nug25	scr20	tai25a	tho40

(b) Confusion matrix

# Real Image Matching

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
RRWM [9]	41.5	54.7	54.3	50.3	67.9	74.3	70.3	60.6	<b>42.3</b>	59.1	48.1	57.3	59.1	56.2	40.6	69.6	63.1	52.2	76.3	87.8	59.3
GMN [39]	31.9	47.2	51.9	40.8	68.7	72.2	53.6	52.8	34.6	48.6	<b>72.3</b>	47.7	54.8	51.0	38.6	75.1	49.5	45.0	83.0	86.3	55.3
PCA-GM [32]	40.9	55.0	<b>65.8</b>	47.9	76.9	<b>77.9</b>	63.5	67.4	33.7	65.5	63.6	61.3	<b>68.9</b>	62.8	44.9	77.5	67.4	57.5	<b>86.7</b>	<b>90.9</b>	63.8
NGM (ours)	50.1	63.5	57.9	53.4	<b>79.8</b>	77.1	73.6	68.2	41.1	<b>66.4</b>	40.8	60.3	61.9	63.5	45.6	77.1	<b>69.3</b>	<b>65.5</b>	79.2	88.2	64.1
NHGM (ours)	<b>52.4</b>	62.2	58.3	55.7	78.7	77.7	<b>74.4</b>	<b>70.7</b>	42.0	64.6	53.8	61.0	61.9	60.8	46.8	<b>79.1</b>	66.8	55.1	80.9	88.7	64.6
NGM+ (ours)	50.8	<b>64.5</b>	59.5	<b>57.6</b>	79.4	76.9	<b>74.4</b>	69.9	41.5	62.3	68.5	<b>62.2</b>	62.4	<b>64.7</b>	<b>47.8</b>	78.7	66.0	63.3	81.4	89.6	<b>66.1</b>

Table 1. Accuracy (%) on Pascal VOC Keypoint. We test our proposed NGM, NHGM, NGM+ with RRWM [9], GMN [39], PCA-GM [32].

- NGM performs comparatively as PCA-GM (even slightly better)
- NHGM outperforms NGM by 3<sup>rd</sup> order affinity
- NMGM achieves improvement by fusing multi-graph information
- NGM+ reaches state-of-the-art performance

method	extra data	face	m-bike	car	duck	w-bottle
HARG-SSVM [8]	×	91.2	44.4	58.4	55.2	66.6
GMN [39]	✓	99.3	71.4	74.3	82.8	76.7
PCA-GM [32]	✓	<b>100.0</b>	76.7	84.0	<b>93.5</b>	96.9
NGM	×	99.2	82.1	84.1	77.4	93.5
NMGM-3-T	×	99.4	84.6	91.3	83.1	94.6
NMGM-6-T	×	99.5	87.3	90.4	82.3	94.3
NMGM-3	×	<b>100.0</b>	84.8	97.2	88.1	94.9
NMGM-6	×	<b>100.0</b>	<b>86.0</b>	<b>97.3</b>	85.4	<b>97.7</b>

Table 2. Accuracy (%) on Willow ObjectClass dataset.

# Outline

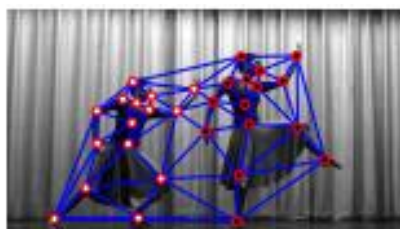
- Introduction
- Background: Learning on Graph Matching
- Embedding approach for Deep Graph Matching
- **Other techniques**
- Outlook
- Brief experience sharing

# Joint matching and cuts (CVPR'18)

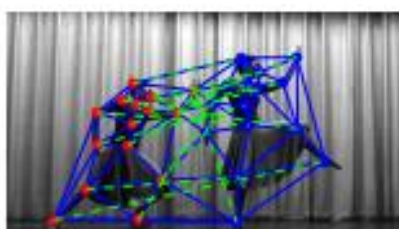
**Problem formulation:** Joint cuts and matching of two objects in one single image

**Motivation:** graph cuts aim to separate dissimilar objects, matching aims not

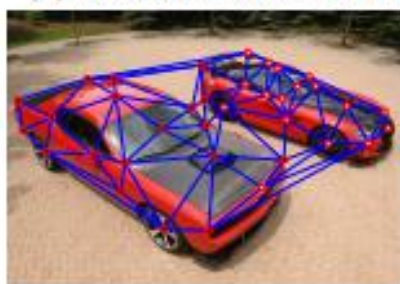
**Limitation:** the key points are assumed to be detected in prior.



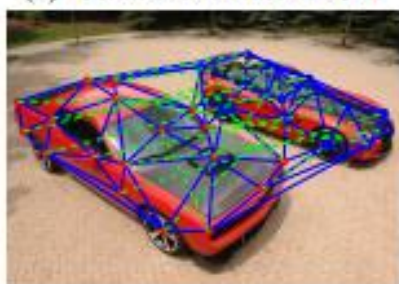
(o) CutMatch-cuts: 15/15



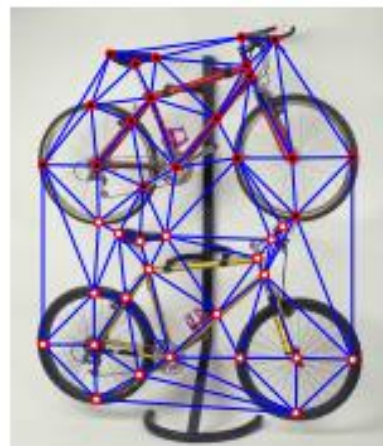
(s) CutMatch-match: 7/15



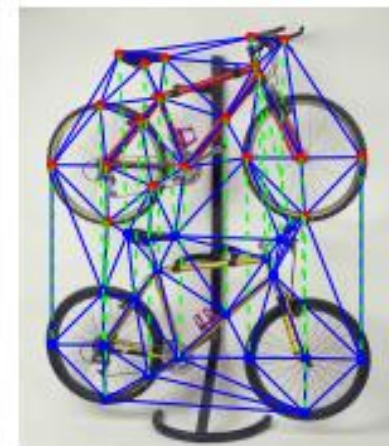
(p) CutMatch-cuts: 15/18



(t) CutMatch-match: 4/18



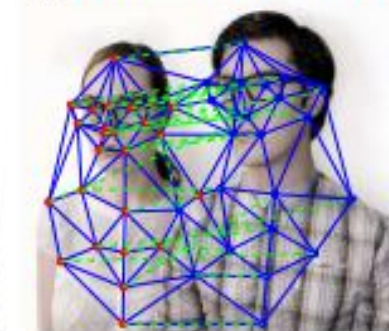
(m) CutMatch-cuts: 19/20



(q) CutMatch-match: 20/20



(n) CutMatch-cuts: 25/25

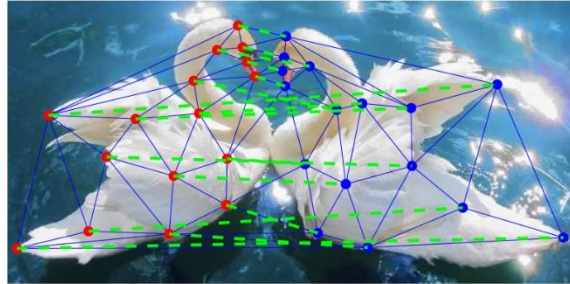


(r) CutMatch-match: 16/25

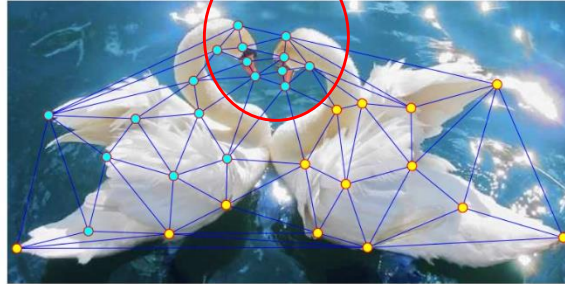
# Main idea (CVPR'18)

Joint Cuts and Matching of Partitions in One Graph,  
CVPR 2018

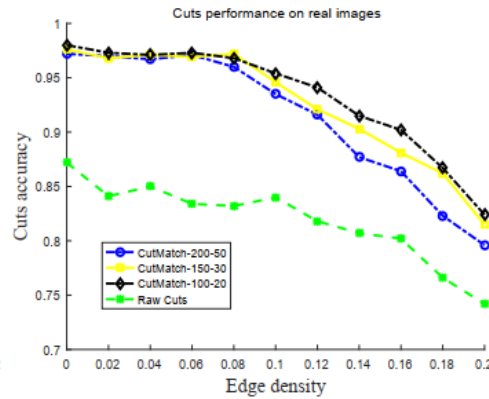
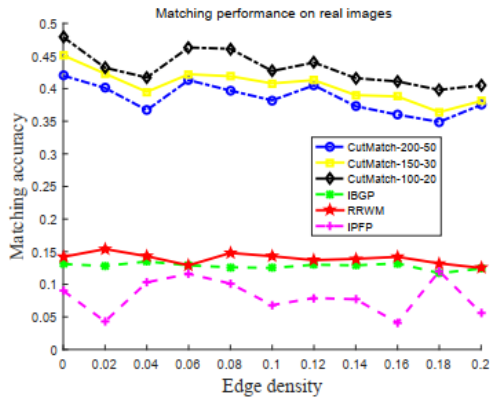
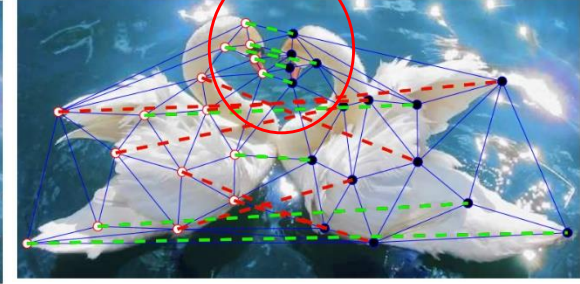
Ground-truth partitions and matchings



Initial graph cut result



Matching and cut with CutMatch



$$\min_{\mathbf{y}} \frac{\mathbf{y}^T \mathbf{L}^g \mathbf{y}}{\mathbf{y}^T \mathbf{y}}$$

Graph cuts

$$\max_{\mathbf{X}} \text{vec}(\mathbf{X})^T \mathbf{A} \text{vec}(\mathbf{X}) \quad \text{Graph matching}$$

$$\text{s.t. } \sum_i \mathbf{X}_{ij} = \mathbf{1}, \sum_j \mathbf{X}_{ij} = \mathbf{1}^T, \mathbf{X}_{ii} = 0, \mathbf{X} \in \mathbb{S}^+$$

$$\mathbf{y}^T (\mathbf{I} - \mathbf{X}) \mathbf{y} \quad \text{Match agrees cuts}$$

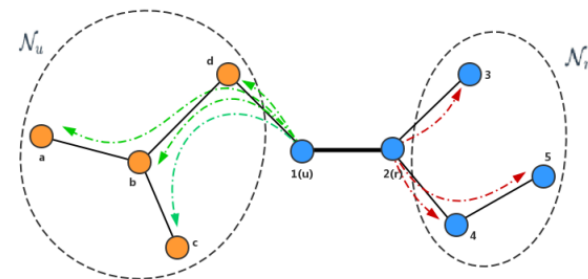
$$\sum_{i,j} \mathbf{X}_{ij} (\mathbf{y}_i - \mathbf{y}_j)^2$$

$$\max_{\mathbf{X}, \mathbf{y}} \mathbf{x}^T \mathbf{A} \mathbf{x} - \lambda_1 \mathbf{y}^T \mathbf{L}^g \mathbf{y} + \lambda_2 \mathbf{y}^T (\mathbf{I} - \mathbf{X}) \mathbf{y} \quad (4)$$

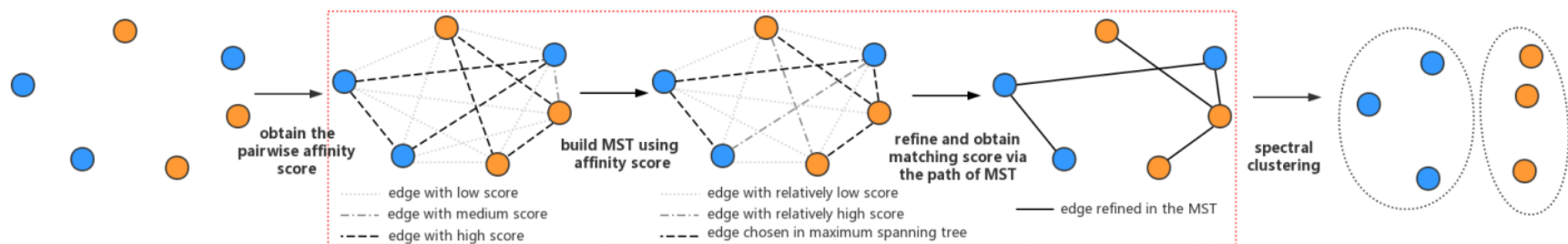
$$\text{s.t. } \sum_i \mathbf{X}_{ij} = \mathbf{1}, \sum_j \mathbf{X}_{ij} = \mathbf{1}^T, \mathbf{X}_{ii} = 0, \mathbf{X} \in \mathbb{S}^+, \|\mathbf{y}\|_2^2 = 1$$

# Clustering aware matching (AAAI'20, NeurIPS'20)

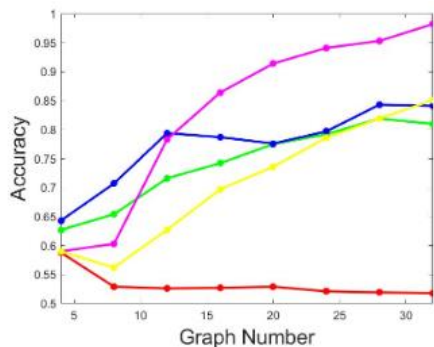
- Clustering-aware Multiple Graph Matching via Decayed Pairwise Matching Composition, AAAI 2020
- Graduated Assignment for Joint Multi-Graph Matching and Clustering with Application to Unsupervised Graph Matching Network Learning, NeurIPS 2020



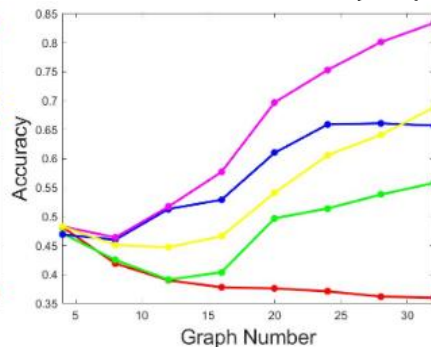
(a) matching updating along maximum spanning tree



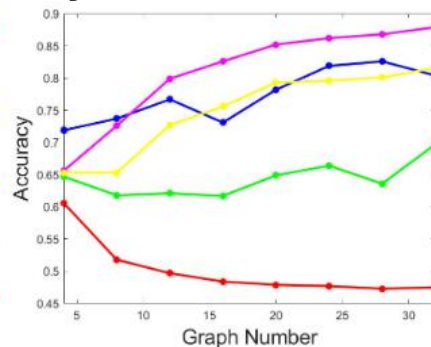
Multiple Graph Matching Solver



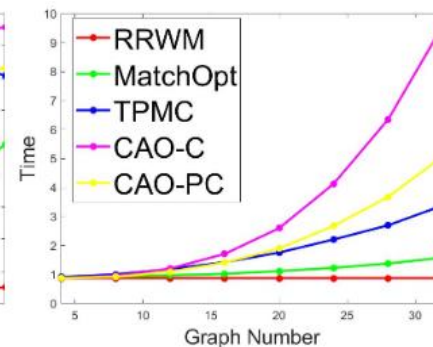
(a) accuracy: outlier



(b) accuracy: deform



(c) accuracy: car



(d) time: car

# Incremental matching

## ECCV'18, ECCV'20

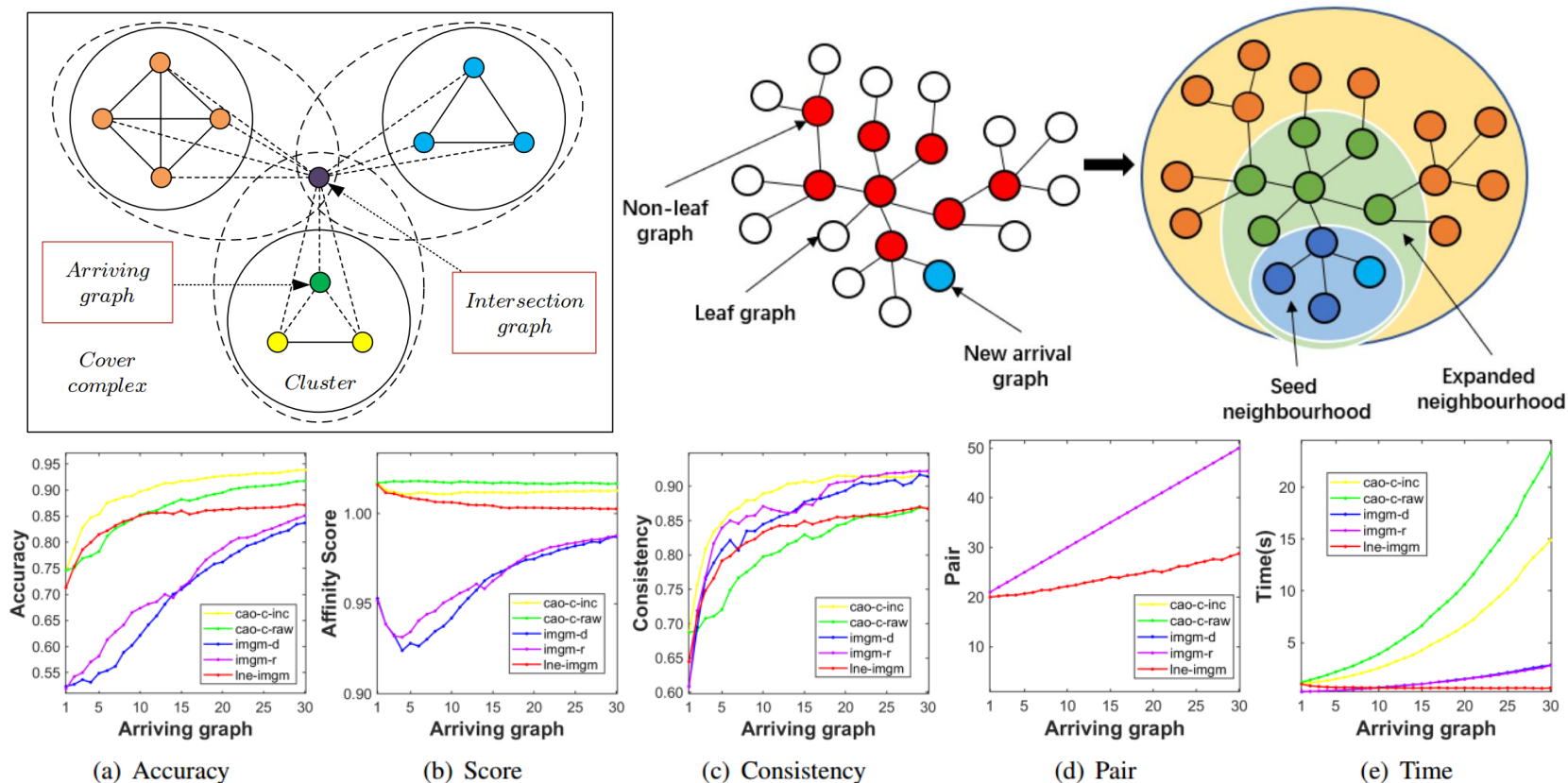


Figure 2. Evaluation on synthetic deformed graphs in **online IMG M** setting. LNE-IMG M is our method.

T. Yu, J. Yan, W. Liu, B. Li, Incremental Multi-graph Matching via Diversity and Randomness based Graph Clustering, ECCV 2018

Z. Chen, Z. Xie, J. Yan, Y. Zheng, X. Yang, Layered Neighborhood Expansion for Incremental Multiple Graph Matching, ECCV 2020

# Large-scale graph matching for social networks/bioinformatics

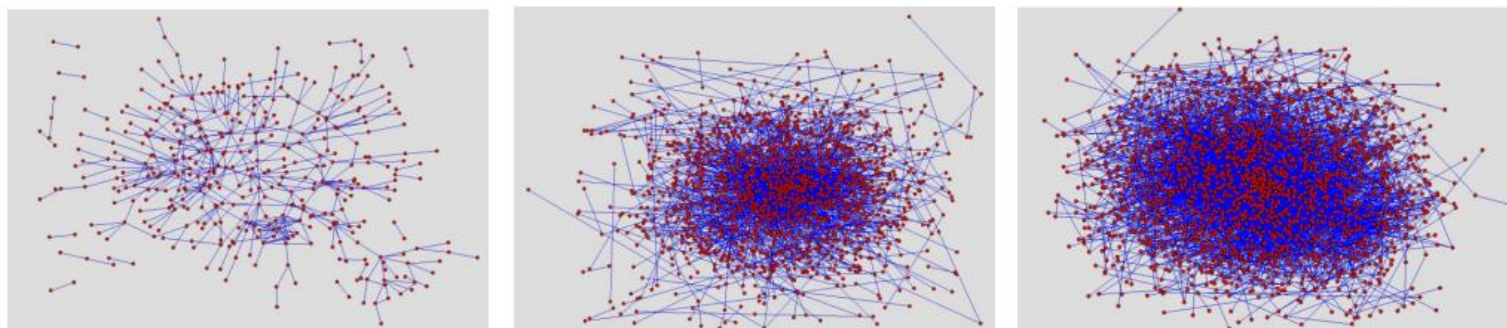
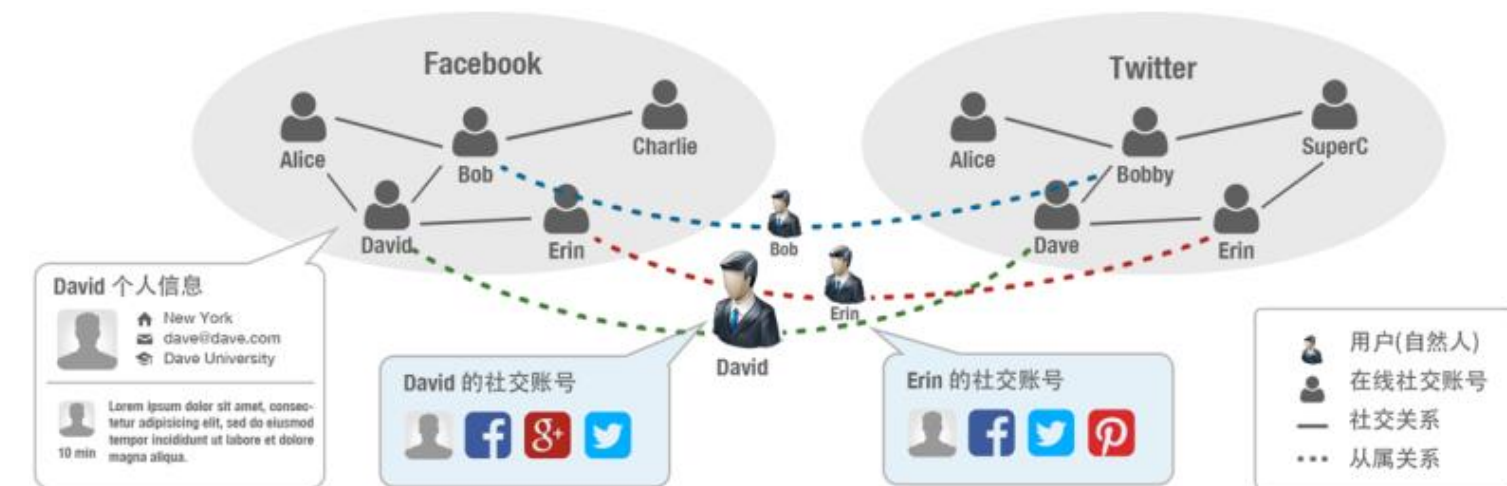


图 3-1 蛋白质相互作用网络示例图 (从左至右为: 果蝇、蛔虫、酵母)

# Joint matching and link prediction (IJCAI'19/TKDE'20)

**Problem:** Joint matching and link prediction

**Motivation:** two can benefit each other

**Limitation:** result relies on data's distribution.

## Assumption

A person in different social networks will have:

similar **degrees**

similar **neighbors** (friendship)

similar **attributes** (name, age, sex, location, text...)

.....

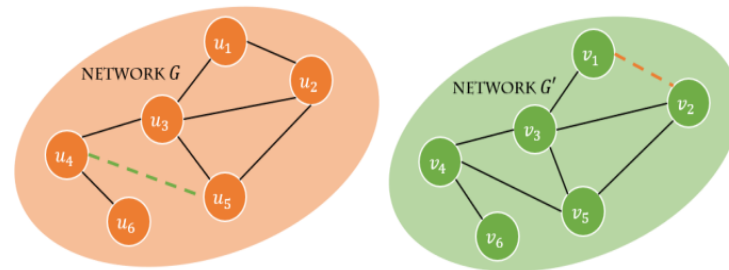


Figure 1: Illustration for how joint modeling of link prediction and network alignment can benefit to each other. Given the correspondence  $v_1 = \pi(u_1)$ ,  $v_2 = \pi(u_2)$ , it is convincing that  $(v_1, v_2)$  should be linked as  $(u_1, u_2)$  is linked. On the other hand, if  $(u_4, u_5)$  and  $(v_4, v_5)$  are not linked,  $u_4$  will be unlikely to link to  $v_4$  since they bear different local structures. Conversely, if  $(u_4, u_5)$  and  $(v_4, v_5)$  are linked,  $u_4$  is more likely to align with  $v_4$  (similarly  $u_5$  corresponds to  $v_5$ ), as they have similar local structures.

Joint Link Prediction and Network Alignment via Cross-graph Embedding, **IJCAI** 2019

Cross-network Skip-gram Embedding for Joint Network Alignment and Link Prediction, **TKDE** 2020

# Construct an undirected compound network

For each account  $u$  in  $G$  and  $v$  in  $G'$ , we denote  $w(u, v)$  as the **weight** between  $u$  and  $v$ . Then, an undirected compound network is obtained.

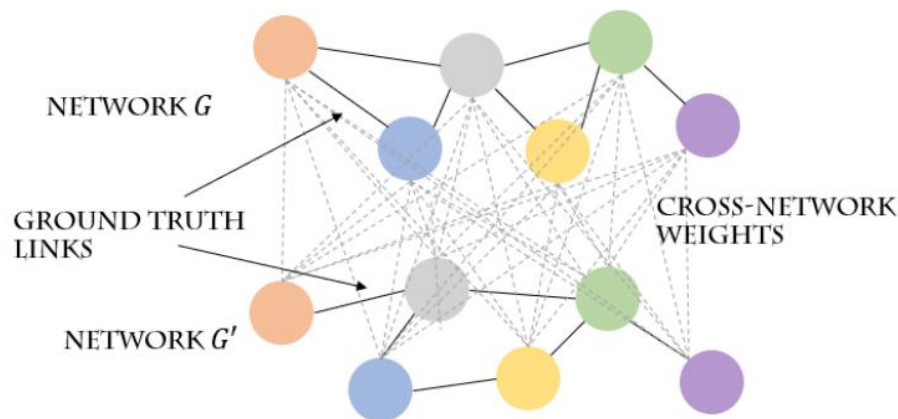
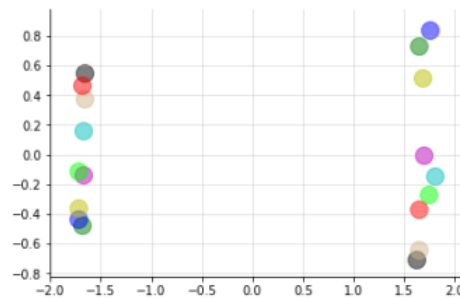
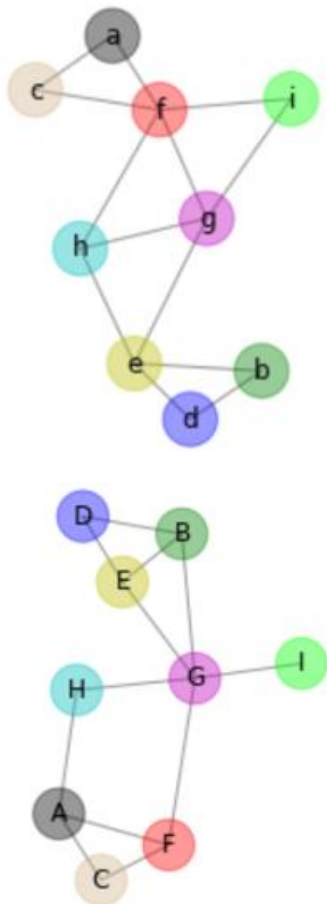
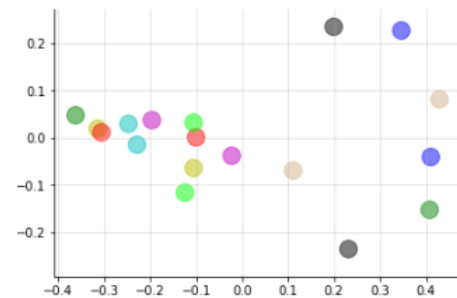


Figure 2: Weighted, undirected network  $\tilde{G}$  consisting of two networks with cross-graph links among vertices in two networks.

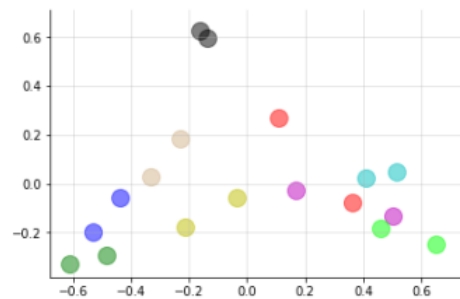
# Embedding visualization



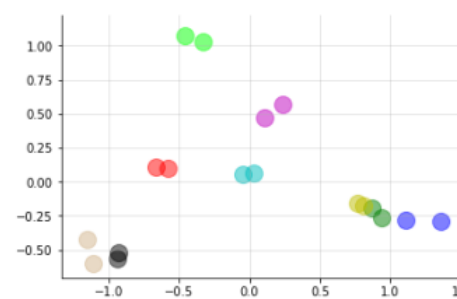
(b) DeepWalk



(c) Struc2vec



(d) CENA

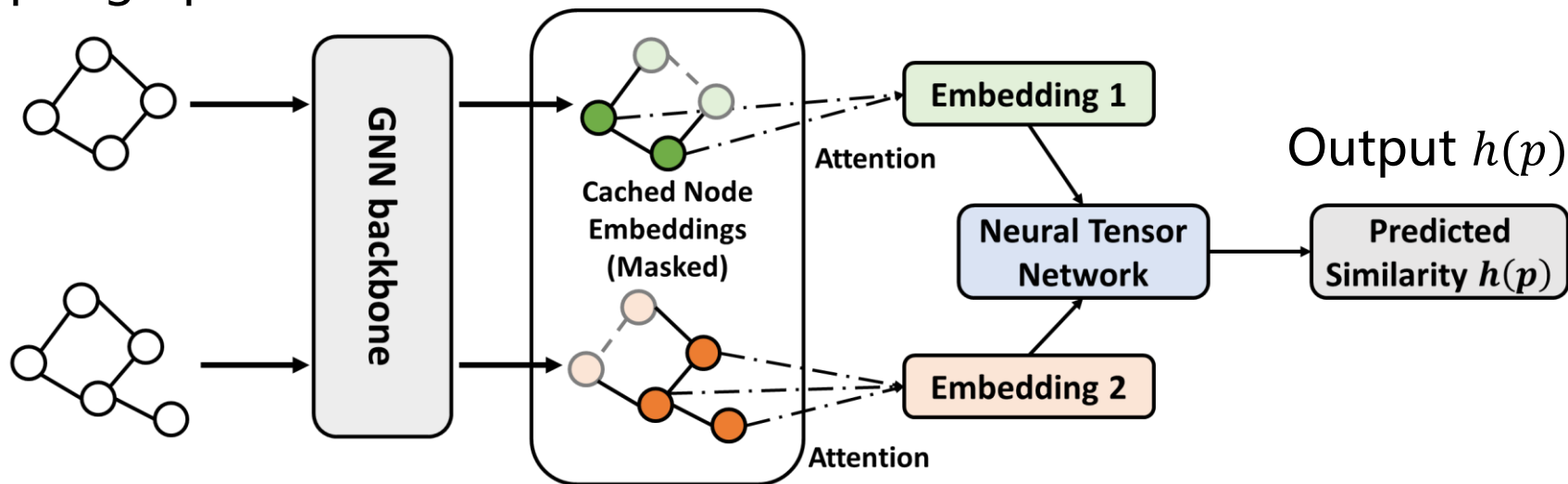


(e) CENALP

Figure 7: Distribution of embedded vertices using different embedding techniques. The embedding vectors are decomposed to 2-dimension using principle component analysis (PCA).

# Learning to solve Graph Edit Distance (CVPR21): The Best of Two Worlds: Compute $h(p)$ in $A^*$ by Neural Network

Input graphs



Source Graph



GED=1.62



GED=1.94



GED=2.34



GED=3.80

Graph Edit Distance

# Joint Graph Edge Building and Matching (ICML'21)

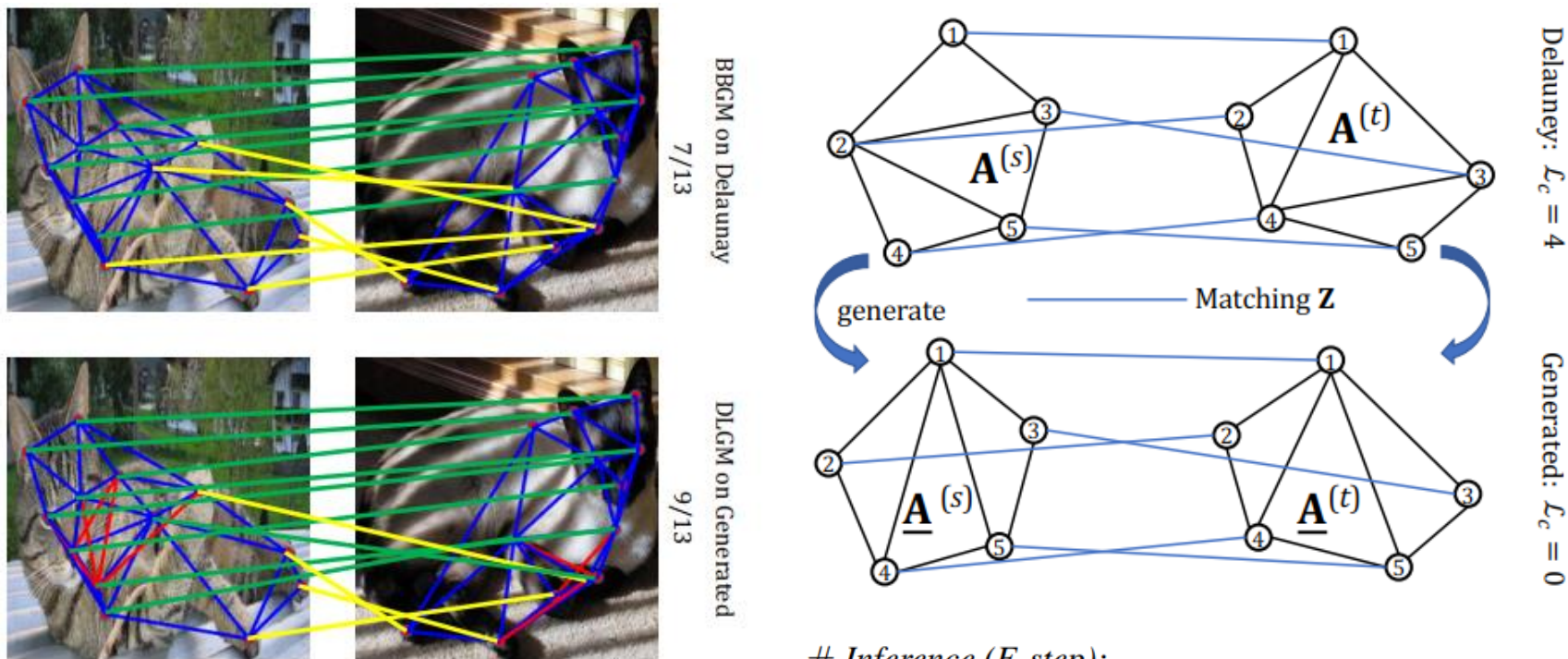


Figure 1. Matching of BBGM (Rolínek et al., 2020) 7/13 with Delaunay triangulation and our DLGM-G 9/13 using generated graph (Pascal VOC). Red edges are the ones different from blue Delaunay edges. Green and yellow lines correspond to positive and negative matchings, respectively. GM solver favors topology generated from DLGM-G, and thus leads to better accuracy.

# Inference (E-step):

Obtain predicted **matching**  $\hat{Z}$  using fixed  $P_\theta$ ;  
Update  $Q_\phi$  (i.e.  $N_G$ ) with loss  $\mathcal{L}_L + \mathcal{L}_C(\cdot | \hat{Z})$  according to Eq. (16);

# Learning (M-step):

Obtain predicted graph **topology**  $\underline{A}^{(s)}$  and  $\underline{A}^{(t)}$  using  $Q_\phi$ ;

Update  $P_\theta$  (i.e.  $N_B$  and  $N_R$ ) with loss  $\mathcal{L}_M$  given  $\underline{A}^{(s)}$  and  $\underline{A}^{(t)}$  according to Eq. (17);

# Outline

- Introduction
- Background: Learning on Graph Matching
- Embedding approach for Deep Graph Matching
- Other techniques
- **Outlook**

# Summary

- Introduction on graph matching
- Learning of graph matching
- Embedding based deep graph matching pipeline
- Other techniques
  - Joint graph matching and graph cut
  - Joint graph matching and clustering
  - Incremental matching of multiple graphs
  - Joint link prediction and matching
  - Optimal transport examples

# Outlook

- Large-scale & robust network alignment
- Incremental matching of graphs online
- Matching against massive outliers
- Discovery and construction of graphs
- Meta-learning of graph matching solvers
- Multi-task with graph matching e.g. cut, link prediction etc.

# Publications

1. J. Ma, X. Jiang, A. Fan, J. Jiang, Junchi Yan\*, Image Matching from Handcrafted to Deep Features: A Survey, [International Journal of Computer Vision \(IJCV\)](#), 2021
2. R. Wang, Junchi Yan\*, X. Yang: Neural Graph Matching Network: Learning Lawler's Quadratic Assignment Problem with Extension to Hypergraph and Multiple-graph Matching. [IEEE Transactions on Pattern Analysis and Machine Intelligence \(TPAMI\)](#), 2021
3. Wang, Junchi Yan\*, X. Yang, Combinatorial Learning of Robust Deep Graph Matching: an Embedding based Approach, [IEEE Transactions on Pattern Analysis and Machine Intelligence \(TPAMI\)](#), 2020
4. Z. Jiang, T. Wang, Junchi Yan\*, Unifying Offline and Online Multi-graph Matching via Finding Shortest Paths on Supergraph, [IEEE Transactions on Pattern Analysis and Machine Intelligence \(TPAMI\)](#), 2020
5. Junchi Yan, M. Cho, H. Zha, X. Yang, S. Chu: Multi-Graph Matching via Affinity Optimization with Graduated Consistency Regularization, [IEEE Transactions on Pattern Analysis and Machine Intelligence \(TPAMI\)](#), 2016, 38(6): 1228-1242

# Publications

6. Junchi Yan, C. Li, Y. Li, G. Cao: Adaptive Discrete Hypergraph Matching. [IEEE Transactions on Cybernetics \(TCYB\)](#), 2018, 48(2): 765-779
7. Junchi Yan, X. Liu, L. Shi, C. Li, H. Zha: Consistency-Driven Alternating Optimization for Multigraph Matching: A Unified Approach, [IEEE Transactions on Image Processing \(TIP\)](#), 2015, 24(3): 994-1009
8. Z. Ren, W. Luo, Junchi Yan\*, W. Liao, X. Yang, A. Yuille, H. Zha, STFlow: Self-Taught Optical Flow Estimation Using Pseudo Labels, [IEEE Transactions on Image Processing \(TIP\)](#), 2020
9. Z. Ren, Junchi Yan\*, X. Yang, A. Yuille, H. Zha, Unsupervised Learning of Optical Flow With Patch Consistency and Occlusion Estimation, [Pattern Recognition \(PR\)](#), 2020
10. X. Du, Junchi Yan\*, R. Zhang, H. Zha, Cross-network Skip-gram Embedding for Joint Network Alignment and Link Prediction, [IEEE Transactions on Knowledge Discovery and Data Engineering \(TKDE\)](#), 2020
11. X. Hao, Junchi Yan\*, BTWalk: Branching Tree Random Walk for Multi-order Structured Network Embedding, [IEEE Transactions on Knowledge Discovery and Data Engineering \(TKDE\)](#), 2020

# Publications

12. Junchi Yan, H. Xu, H. Zha, X. Yang, S. Chu. A Matrix Decomposition Perspective to Multiple Graph Matching, [ICCV, 2015](#).
13. Junchi Yan, S. Xiao, C. Zhang, H. Zha, W. Liu, X. Yang, S. Chu: Discrete Hyper-graph Matching, [CVPR 2015](#)
14. Junchi Yan, Y. Li, W. Liu, H. Zha, X. Yang, S. Chu: Graduated Consistency-regularized Optimization for Multi-graph Matching. [ECCV 2014](#)
15. Junchi Yan, J. Wang, H. Zha, X. Yang, S. Chu: Multi-view Point Registration via Alternating Optimization. [AAAI 2015](#)
16. Junchi Yan, Y. Tian, H. Zha, X. Yang, Y. Zhang, S. Chu: Joint Optimization for Consistent Multiple Graph Matching. [ICCV 2013](#)
17. T. Yu, Junchi Yan\*, J. Zhao, B. Li: Joint Cuts and Matching of Partitions in One Graph, [CVPR 2018](#)
18. T. Yu, Junchi Yan, W. Liu, B. Li: Incremental Multi-graph Matching via Diversity and Randomness based Graph Clustering. [ECCV 2018](#).

# Publications

19. T. Yu, Junchi Yan\*, B. Li, Determinant Regularization for Gradient-Efficient Graph Matching, [CVPR 2020](#)
20. T. Wang, Z. Jiang, Junchi Yan\*, Clustering-aware Multiple Graph Matching via Decayed Pairwise Matching Composition, [AAAI 2020](#)
21. Junchi Yan, S. Yang, E. Hancock, Learning Graph Matching and Related Combinatorial Optimization Problems, [IJCAI 2020](#).
22. T. Yu, R. Wang, Junchi Yan, B. Li, Learning deep graph matching with channel-independent embedding and Hungarian attention, [ICLR 2020](#)
23. Z. Chen, Z. Xie, Junchi Yan\*, Y. Zheng, X. Yang, Layered Neighborhood Expansion for Incremental Multiple Graph Matching, [ECCV 2020](#)
24. X. Du, Junchi Yan\*, H. Zha, Joint Link Prediction and Network Alignment via Cross-graph Embedding, [IJCAI 2019](#)
25. H. Xiong, Junchi Yan\*, Li Pan, Contrastive Multi-View Multiplex Network Embedding with Applications to Robust Network Alignment, [KDD 2021](#)

# Publications

26. T. Yu, R. Wang, Junchi Yan, B. Li, Deep Latent Graph Matching, [ICML 2021](#)
27. R. Wang, Junchi Yan\*, X. Yang, Graduated Assignment for Joint Multi-Graph Matching and Clustering with Application to Unsupervised Graph Matching Network Learning, [NeurIPS 2020](#)
28. T. Yu, Junchi Yan, Y. Wang, W. Liu, B. Li: Generalizing Graph Matching beyond Quadratic Assignment Model. [NIPS 2018](#)
29. Z. Ren, Junchi Yan\*, B. Ni, B. Liu, H. Zha, X. Yang. Unsupervised Deep Learning for Optical Flow Estimation. [AAAI 2017](#).
30. R. Wang, Junchi Yan\*, X. Yang, Learning Combinatorial Embedding Networks for Deep Graph Matching, [ICCV 2019 Oral](#)
31. R. Wang, T. Zhang, T. Yu, Junchi Yan\*, X. Yang, Combinatorial Learning of Graph Edit Distance via Dynamic Embedding, [CVPR 2021](#)

# Thanks!

# Q&A



<https://github.com/Thinklab-SJTU/awesome-ml4co>

yanjunchi@sjtu.edu.cn