Neural Graph Matching and Beyond

Junchi Yan
Dept. of CSE, SJTU
MoE Key Lab for AI
thinklab.sjtu.edu.cn

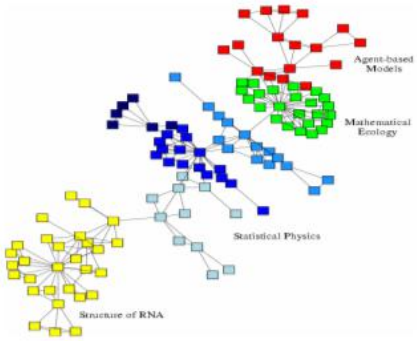
严骏驰
上海交通大学计算机系
人工智能教育部重点实验室
thinklab.sjtu.edu.cn

Outline

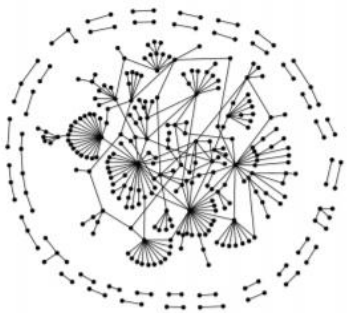
- **Introduction**
- Background: Learning on Graph Matching
- Embedding approach for Deep Graph Matching
- Other techniques
- Outlook

Graphs Everywhere

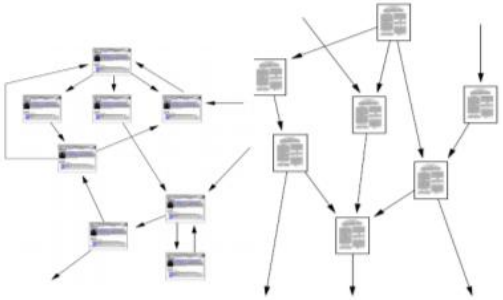
Social networks



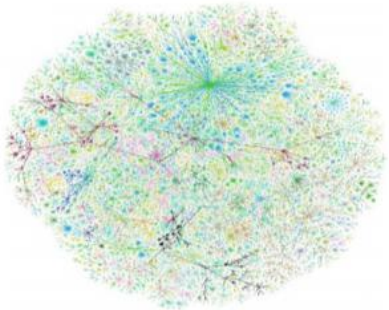
Economic networks



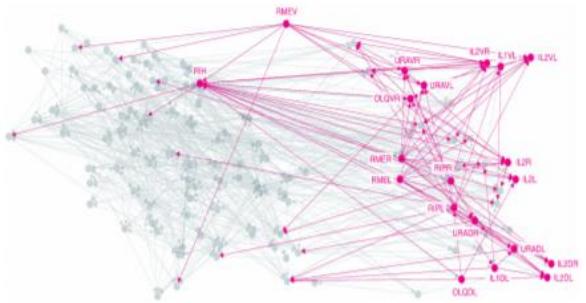
Communication graphs



Information networks:
Web & citations



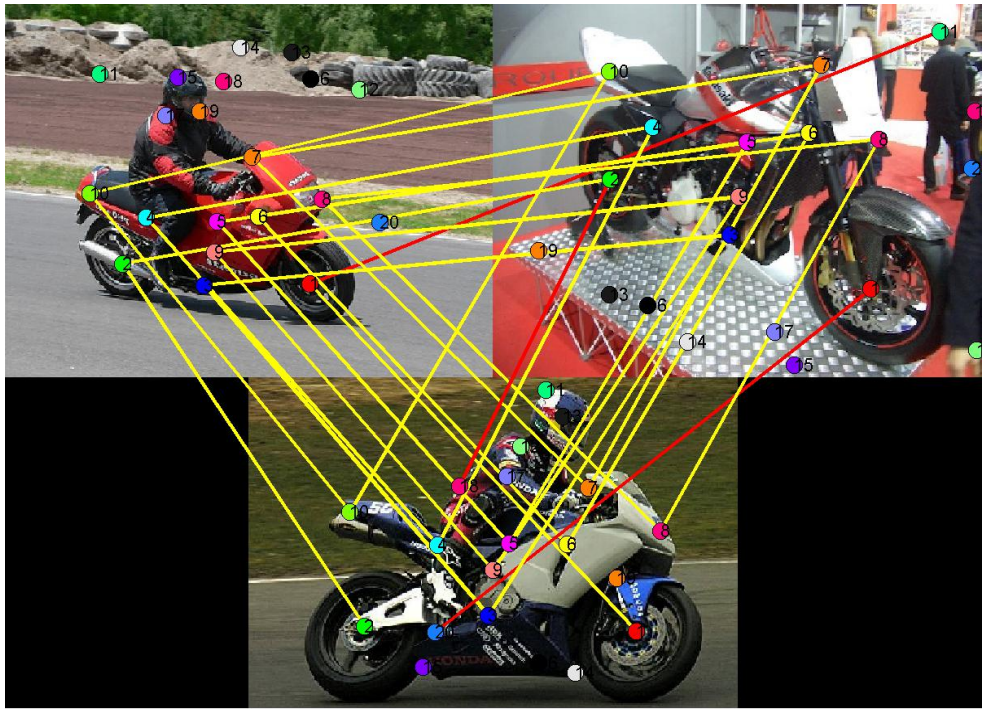
Internet



Networks of neurons

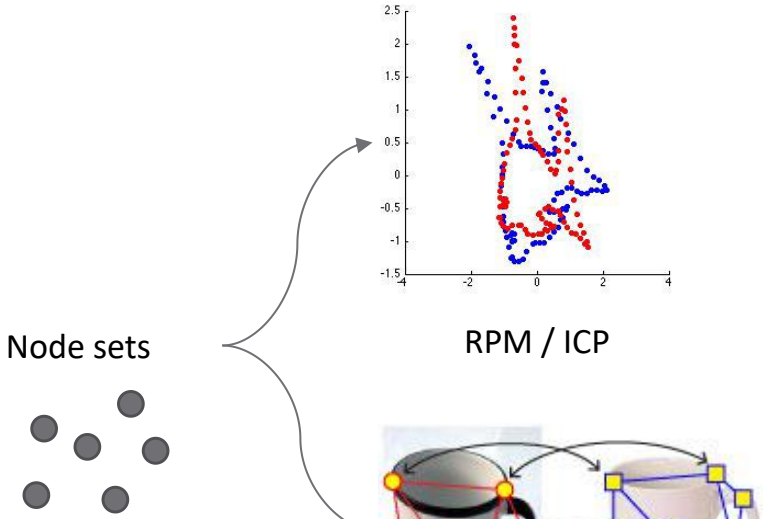
What is Graph Matching

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.

Graph matching vs. registration



Point registration: Often use parametric transformation model between point sets, e.g. RPM, ICP

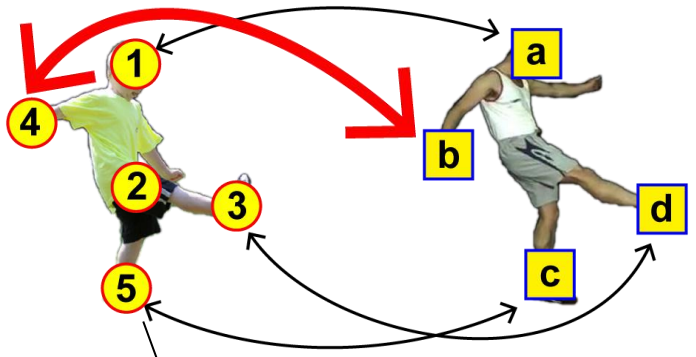
Transformation->correspondence ->Transformation->correspondence

Graph matching: non-parametric form, no prior transform model between graphs

RPM: Chui, H., Rangarajan, A.: A new point matching algorithm for non-rigid registration. Computer Vision and Image Understanding 89 (2003) 114–141
ICP: Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *IJCV*, 1994.
Graph matching: Thirty years of graph matching in pattern recognition. IJPRAI, 2004.

Node correspondence by an assignment matrix

Assume one-to-one correspondence

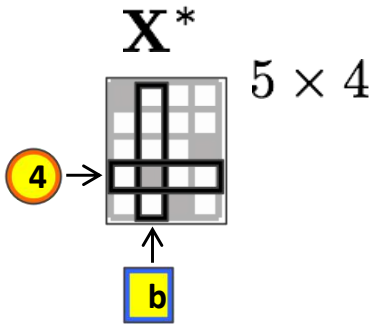


5 detected features

4 detected features

Sift feature
CNN feature
…

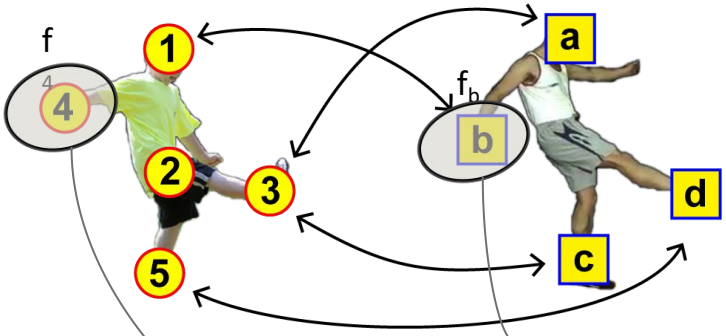
Solution: assignment matrix, i.e. a partial permutation matrix



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

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

Node-wise linear assignment problem



Linear Assignment Problem

$$\max_{\mathbf{X}} \quad \mathrm{tr}\left(\mathbf{X}^T \mathbf{K}_p\right)$$

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

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

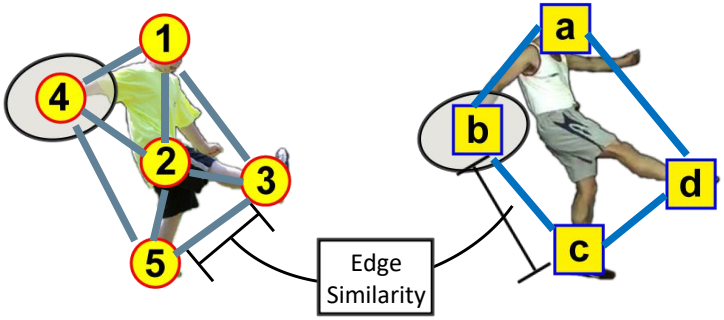
Node-to-node affinity/cost matrix

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

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)

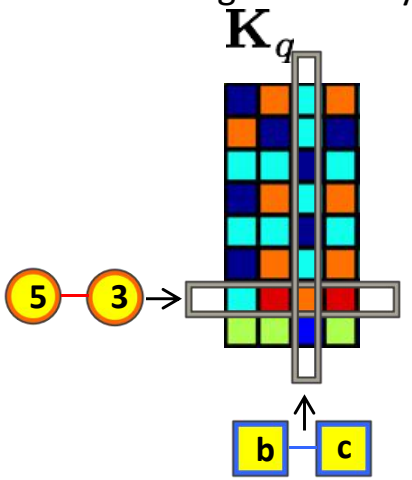
Feature Matching (linear)

+

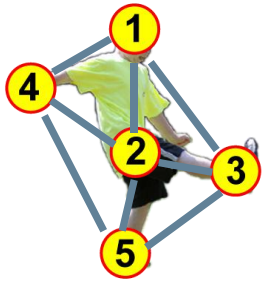
2nd-order Feature (eg, Edge Length)

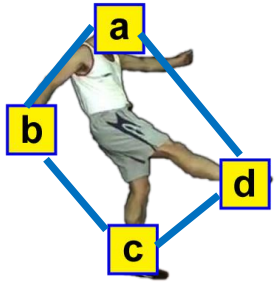
Graph Matching (quadratic)

Edge Similarity

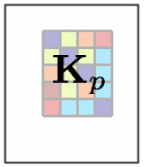


Graph matching: a combinatorial optimization formulation

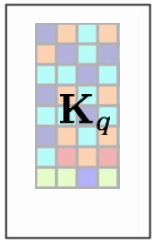




Node Similarity



Edge Similarity



Affinity maximization

Steven Gold

A. Rangarajan

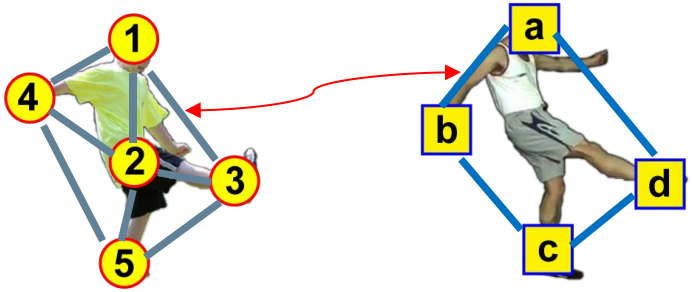
$$\max_{\mathbf{X}} \underbrace{\sum_{i_1 i_2} x_{i_1 i_2} \kappa^p_{i_1 i_2}}_{\text{Node Compatibility}} + \underbrace{\sum_{\substack{(i_1,j_1)\in E_1 \\ (i_2,j_2)\in E_2}} x_{i_1 i_2} x_{j_1 j_2} \kappa^q_{c(i_1,j_1)c(i_2,j_2)}}_{\text{Edge Compatibility}}$$

Node-Edge Index

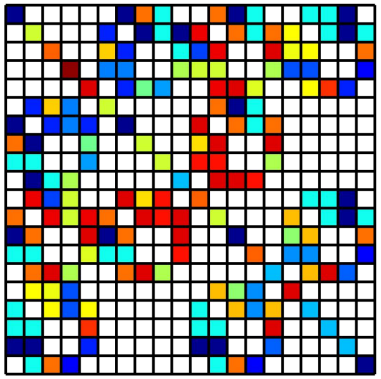
$$\text{s.t.} \quad \mathbf{X} \in \{0,1\}^{5\times 4}, \quad \mathbf{X}\mathbf{1} \le \mathbf{1}, \quad \mathbf{X}^T\mathbf{1} = \mathbf{1}$$

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)*



20×20

Edge-to-edge relations ⇧

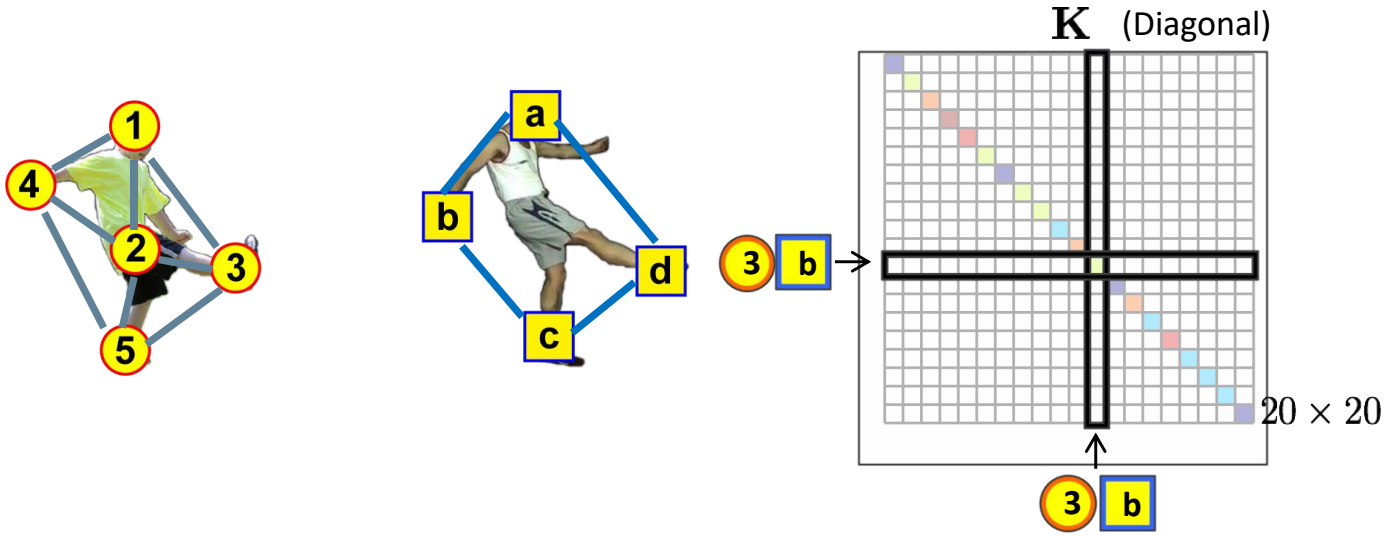
$5 \cdot 4$

M. Leordeanu

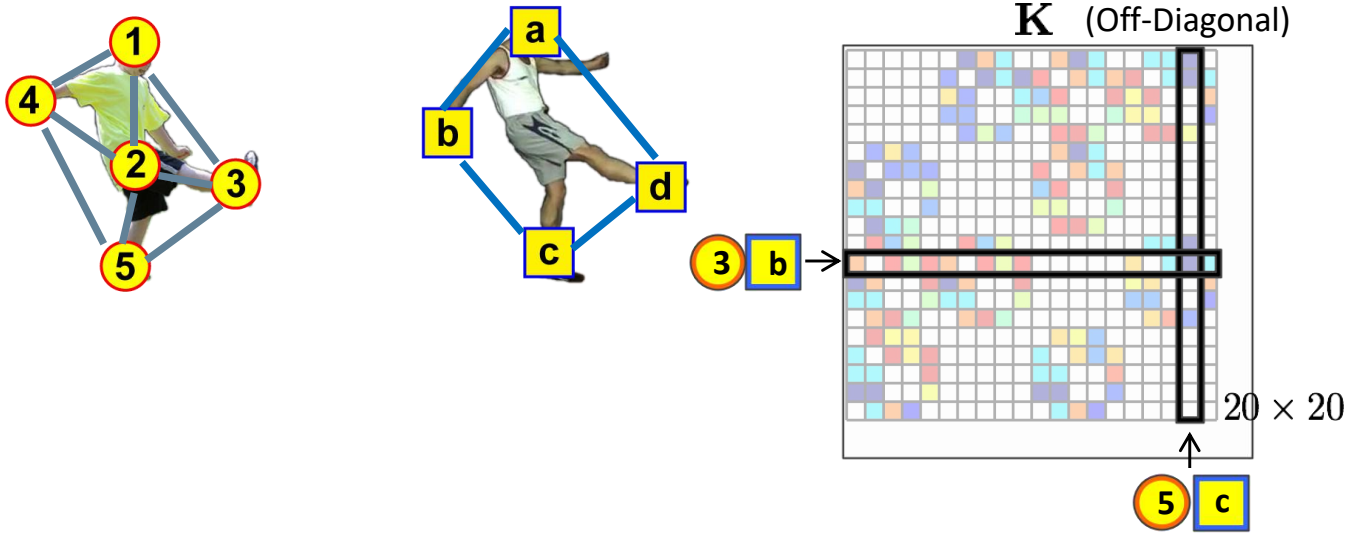
M. Hebert

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

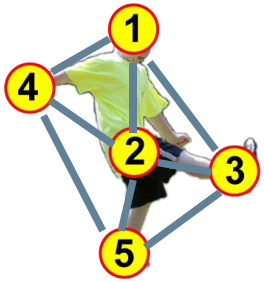
Node-to-node affinity

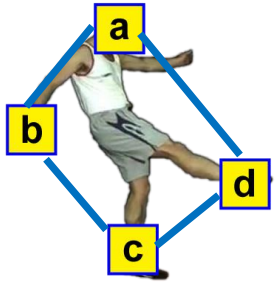


Edge-to-edge affinity



Quadratic Assignment Problem





$$\max_{\mathbf{X}} \quad \mathrm{vec}(\mathbf{X})^T \mathbf{K} \, \mathrm{vec}(\mathbf{X})$$

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

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

NP-hard

Branch-Bound

Koopmans & Beckmann, 1955
Lawler, 1963
Loiola et al, 2007

Spectral Approximation

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

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

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

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

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
- …

Not Convex
(K is Indefinite)

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

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

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

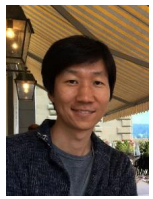
Slower

Not Discrete

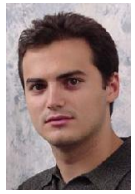
Gradient Method is More Accurate


S. Gold


A. Rangarajan


M. Cho


K. Lee


M. Leordeanu


M. Hebert

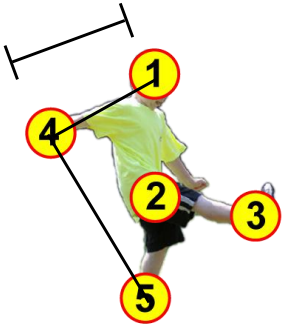

R. Sukthankar

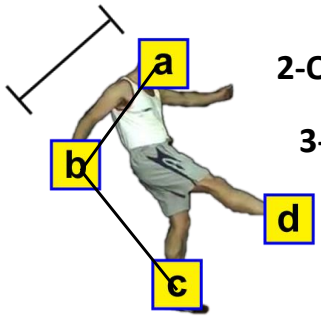
S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," IEEE Transaction on PAMI, 1996

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

Beyond: Higher-order models





2-Order is Rotation / Scale Invariant

3-Order is Similarity Transformation Invariant

4-Order is Affine Transformation Invariant

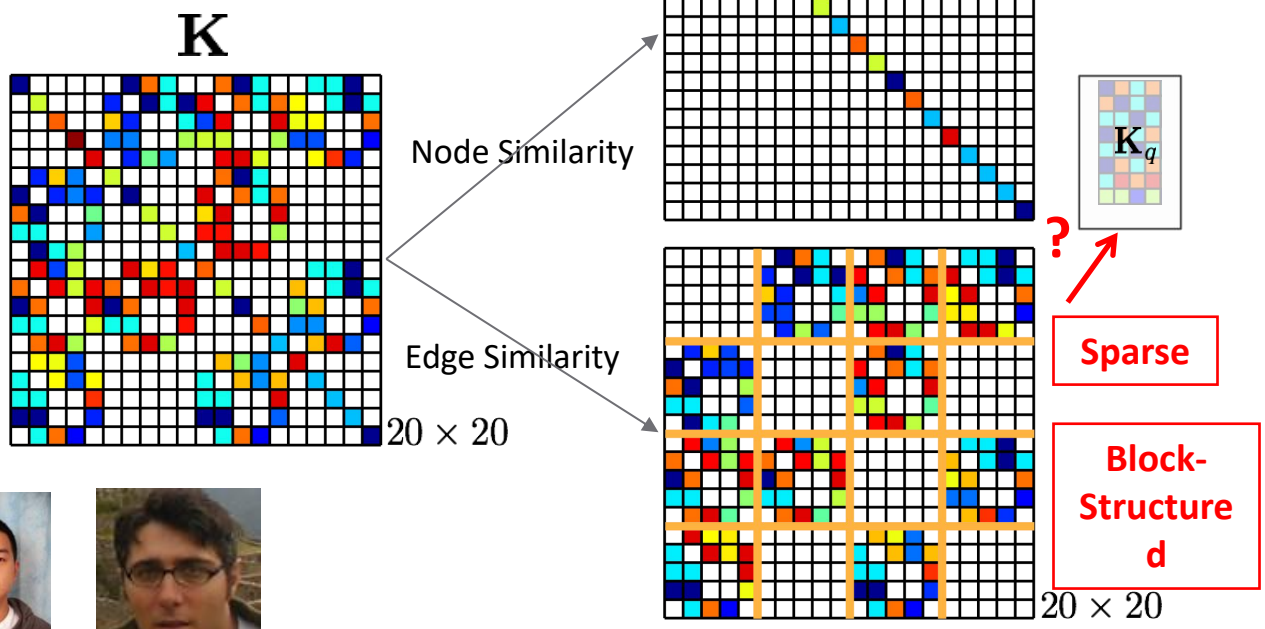
?-Order is Non-rigid Invariant

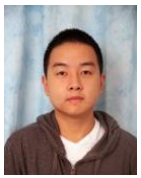
Combinatorial Explosion

K	Complexity	Memory for 100 nodes	
Pair-wise Matrix	$O(n_1^2 n_2^2)$	$100^2 \cdot 100^2 = 10^8$	(381 MB)
Triple-wise Tensor	$O(n_1^2 n_2^2 n_3^2)$	$100^2 \cdot 100^2 \cdot 100^2 = 10^{12}$	(3.7 GB)

Factorized model $\text{diag}\left(\text{vec}\left(\mathbf{K}_p\right)\right)$

5×4





Feng Zhou

Fernando De la Torre

F. Zhou and F. D. Torre, "Factorized graph matching," in CVPR, 2012.

Factorize the edge affinity

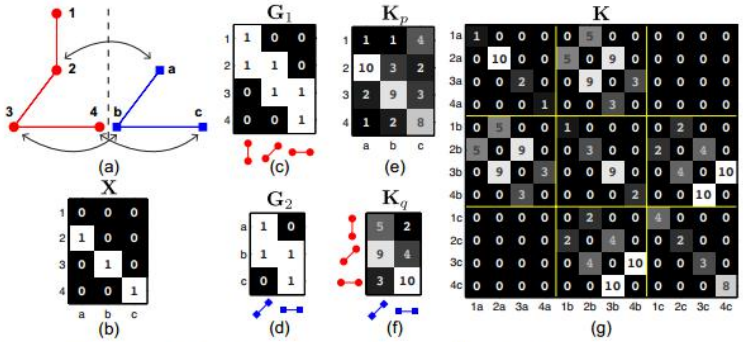
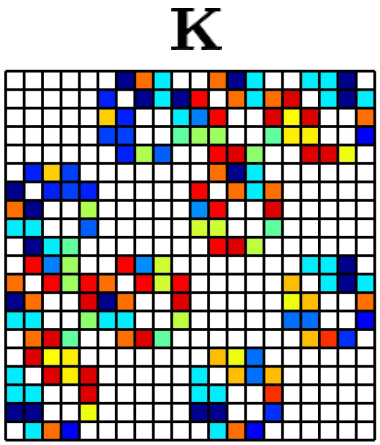


Figure 2. Example of graph matching and related matrices. (a) Two synthetic graphs. (b) The correspondence matrix $\mathbf{X}$. (c) The 1^{st} graph's incidence matrix $\mathbf{G}_1$. (d) The 2^{nd} graph's incidence matrix $\mathbf{G}_2$. (e) The node affinity matrix $\mathbf{K}_p$. (f) The edge affinity matrix $\mathbf{K}_q$. (g) The global affinity matrix $\mathbf{K}$.

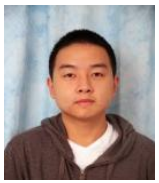


$$\mathbf{K} = (\mathbf{H}_2 \otimes \mathbf{H}_1)\,\mathrm{diag}(\mathrm{vec}(\mathbf{L}))(\mathbf{H}_2 \otimes \mathbf{H}_1)^T, \qquad (2)$$

$$\text{where } \mathbf{H}_1 = [\mathbf{G}_1, \mathbf{I}_{n_1}] \in \{0,1\}^{n_1 \times (m_1+n_1)},$$

$$\mathbf{H}_2 = [\mathbf{G}_2, \mathbf{I}_{n_2}] \in \{0,1\}^{n_2 \times (m_2+n_2)},$$

$$\mathbf{L} = \begin{bmatrix} \mathbf{K}_q & -\mathbf{K}_q\mathbf{G}_2^T \\ -\mathbf{G}_1\mathbf{K}_q & \mathbf{G}_1\mathbf{K}_q\mathbf{G}_2^T + \mathbf{K}_p \end{bmatrix} \in \mathbb{R}^{(m_1+n_1)\times(m_2+n_2)}.$$

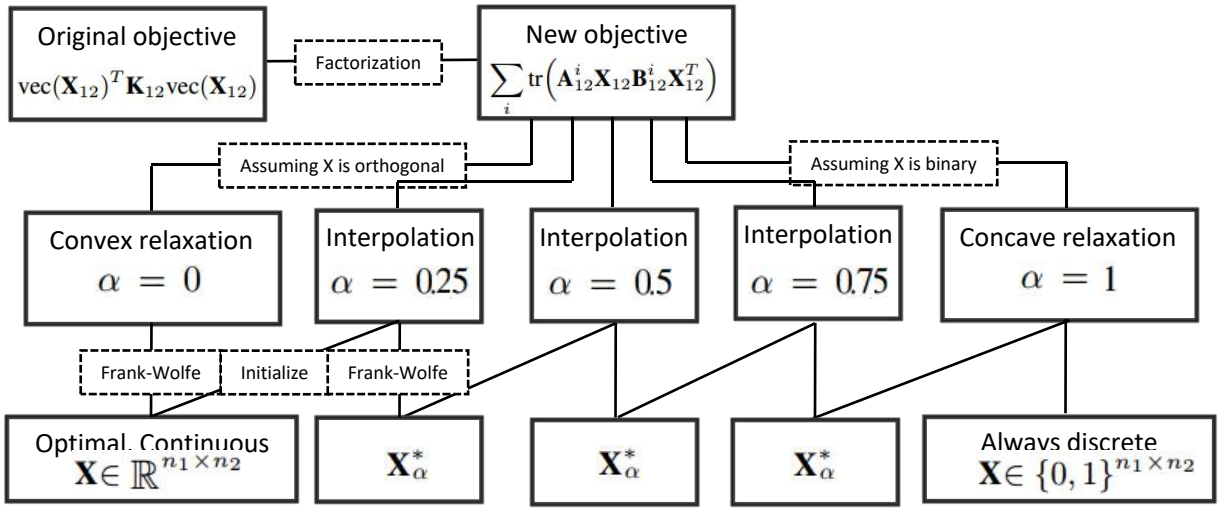


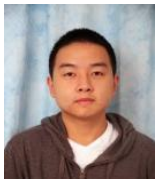
F. Zhou

Fernando De la Torre

F. Zhou and F. D. Torre, "Factorized graph matching," in CVPR, 2012.

Path-following optimization



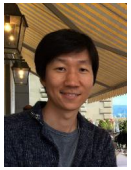


F. Zhou

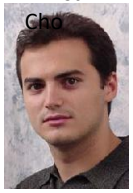
Fernando De la Torre

F. Zhou and F. D. Torre, "Factorized graph matching," in CVPR, 2012.

Two paradigms for two-graph matching

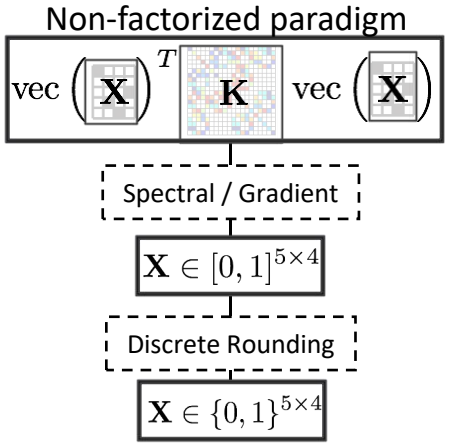


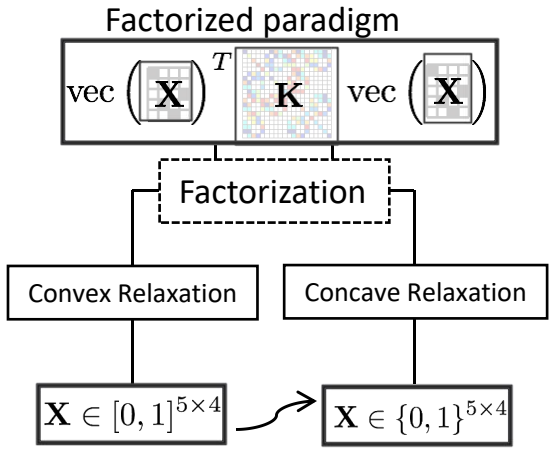
Minsu Cho

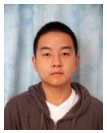


M. Leordeanu

T. Cour







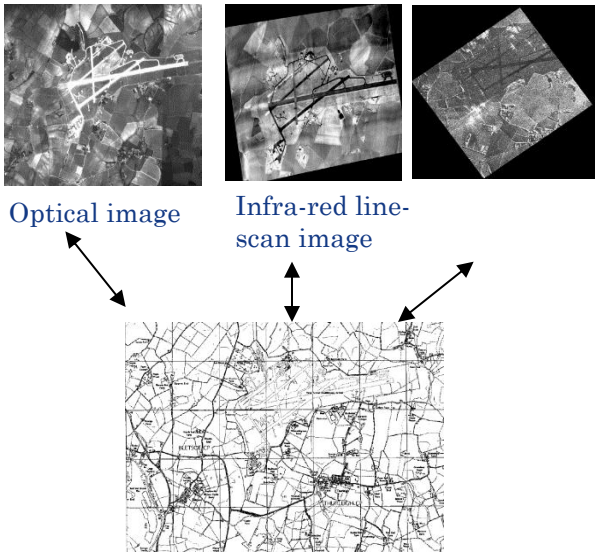
Feng Zhou

Fernando De la Torre

Matching more than two graphs

- More practical problem, with more information to use

Graphical object query and indexing, shape analysis SIGGRAPH'12

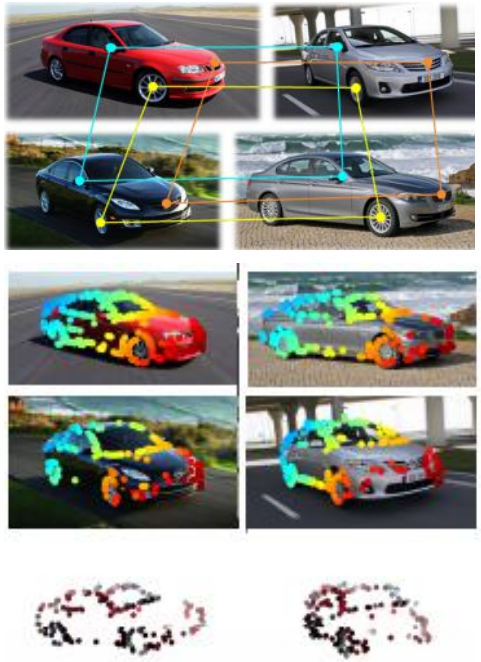


Optical image

Infra-red line-scan image

Cartographic data

Info fusion PRL'97



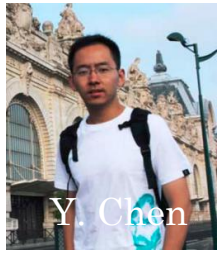
3-D weak reconstruction ICCV'15

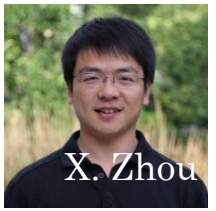
Exploring collections of 3D models using fuzzy correspondences, SIGGRAPH'12
Multiple Graph Matching with Bayesian Inference, Pattern recognition letters'97
Multi-Image Matching via Fast Alternating Minimization, ICCV'15

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





Composition based Affinity Optimization

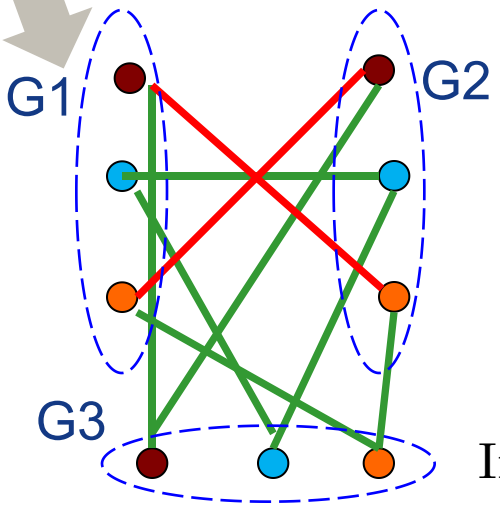
Key idea: efficiently generate new candidate solutions by composition

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

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

Find higher affinity score solution via composition

Mother



Father

Interpolating graph (son)

Affinity J

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

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

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

G1->G2 ✗

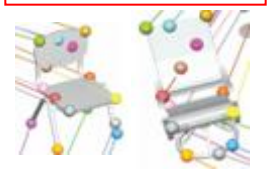
G1->G3->G2 ✓

A simple & general algorithm

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

Local noise

Modeling error



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

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, $\gamma \in (0,1)$;

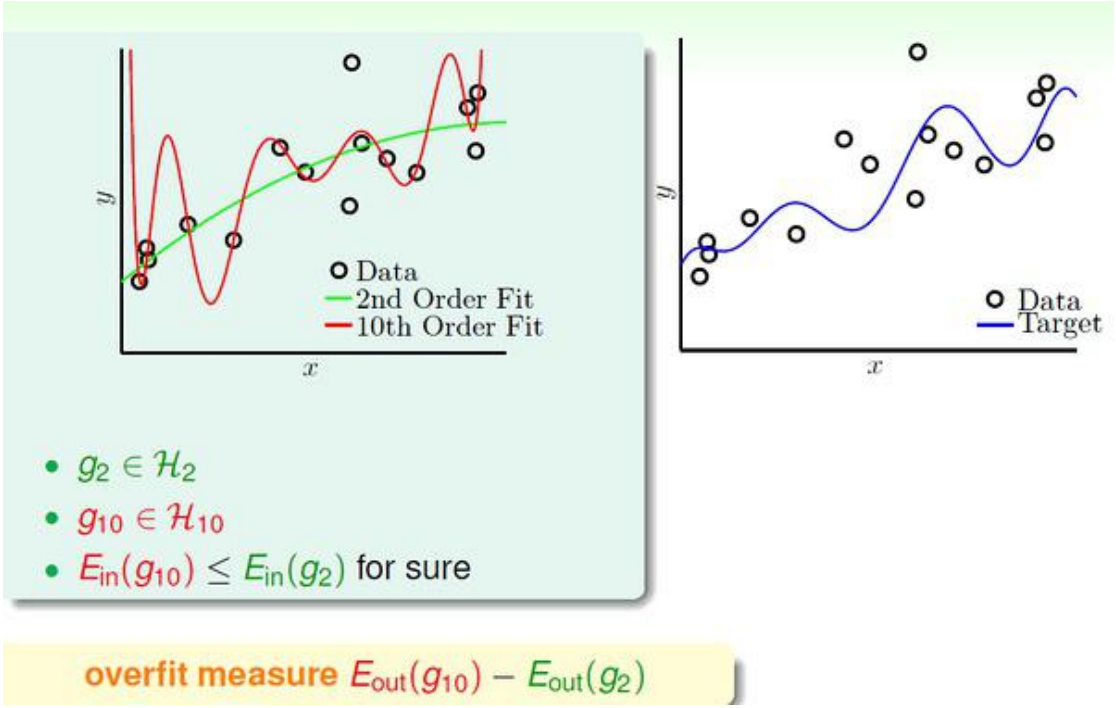
1: Perform pairwise matching to obtain initial $\mathbb{X}^{(0)}$;
2: Calculate $J^{(0)} = \sum_{i=1,j=i+1}^{N-1,N} \mathrm{vec}(\mathbf{X}_{ij}^{(0)})^T \mathbf{K}_{ij} \mathrm{vec}(\mathbf{X}_{ij}^{(0)})$;
3: **for** $t = 1 : T$ **do**
4: **for all** $i = 1, 2, \ldots, N-1; j = i+1, \ldots, N$ **do**
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**

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

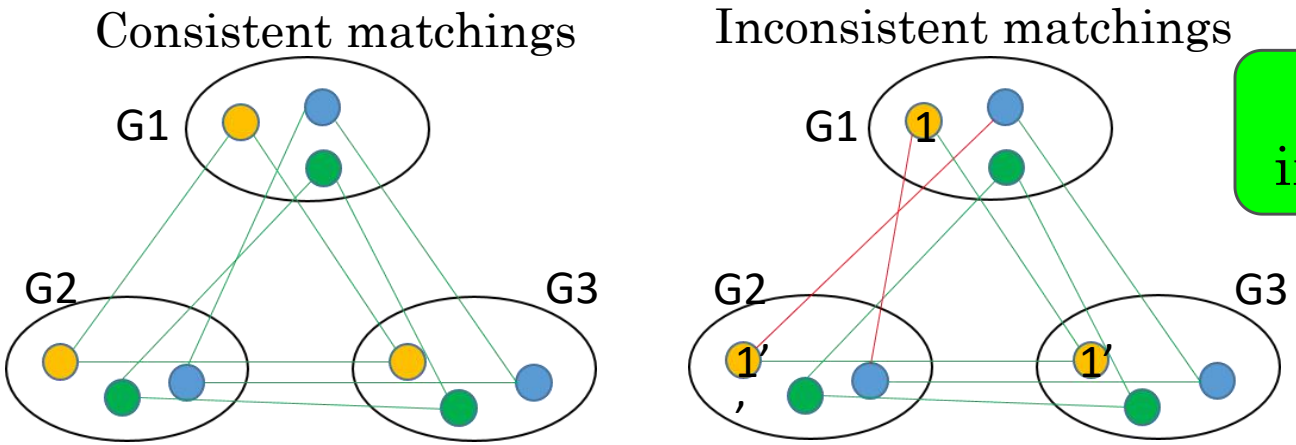
Post-processing
For enforcing overall consistency

Recall over fitting in machine learning

- Affinity score (noise, bias) <-> Empirical term (noise, bias)
- Consistency score <-> Regularization term

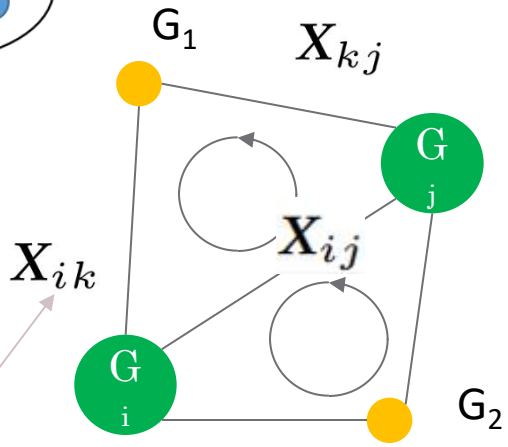


Cycle consistency as regularizer



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



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

K=1,2,…

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

Gradually consistency-weighed CAO (TPAMI'16)

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

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

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, $\lambda^{(T_0)} = \lambda^{(0)}$, $\{\lambda^{(t)}\}_{t=1}^{T_0-1} = 0$, γ;

1: Perform pairwise matching to obtain initial $\mathbb{X}^{(0)}$;
2: Calculate $J^{(0)} = \sum_{i=1,j=i+1}^{N-1,N} \mathrm{vec}(\mathbf{X}_{ij}^{(0)})^T \mathbf{K}_{ij} \mathrm{vec}(\mathbf{X}_{ij}^{(0)})$;
3: **for** $t = 1 : T$ **do**
4: **for all** $i = 1, 2, \ldots, N-1; j = i+1, \ldots, N$ **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).

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

Graph Features

Consider two graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1, \mathcal{A}_1)$, $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2, \mathcal{A}_2)$ built on two images, where $|\mathcal{V}_1| = m, |\mathcal{V}_2| = n$. $\mathcal{A}_1, \mathcal{A}_2$ are attributes (features).

$\mathcal{V}_1, \mathcal{V}_2$ correspond to **1st order (node)** features (e.g. SIFT, CNN).

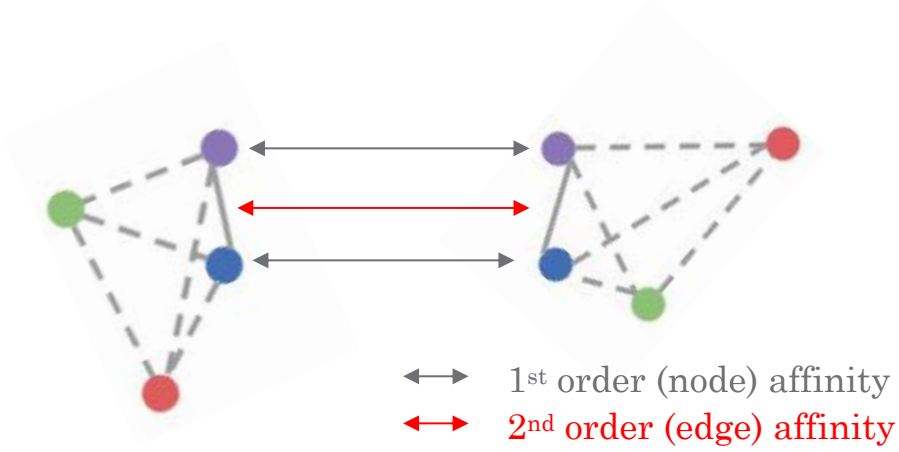
$\mathcal{E}_1$, $\mathcal{E}_1$ correspond to **2nd order (edge)** features (e.g. structural info).

A similarity metric is defined for these features, e.g. gaussian kernel

$$K(\sigma) = \exp\left(\frac{\|x_1 - x_2\|^2}{\sigma^2}\right) \qquad x_1 \in \mathcal{A}_1, x_2 \in \mathcal{A}_2$$

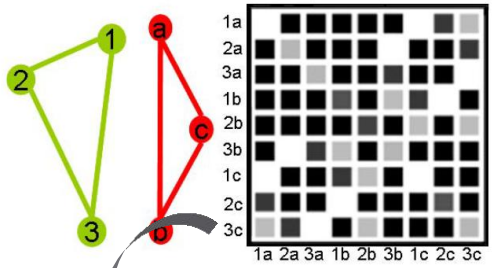
Graph Affinities

In graph matching, both **1st order affinity (node affinity)** and **2nd order affinity (edge affinity)** are considered.



Problem Formulation

Affinity Matrix encodes first order and second order similarities.



The graph matching problem is formulated into an **combinatorial optimization problem**:

$$\max_{\mathbf{X}} \operatorname{vec}(\mathbf{X})^{\mathrm{T}}\, \mathbf{K}\, \operatorname{vec}(\mathbf{X}), \qquad s.t.\ \mathbf{1}^{\mathrm{T}}\mathbf{X} \le \mathbf{1}^{\mathrm{T}}, \mathbf{X}\mathbf{1} \le \mathbf{1}$$

$\mathbf{X} \in \{0,1\}^{m \times n}$: correspondence between two graphs.

$\mathbf{K} \in \mathbb{R}^{mn \times mn}$: affinity matrix.

$\operatorname{vec}(\cdot)$: column-wise vectorization.

Problem Formulation

The combinatorial optimization problem

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

Solving this Quadratic Assignment Problem (aka QAP) is **NP-hard**!

Relaxation approaches have been proposed to solve this problem in reasonable time.

However, **problems exist**:

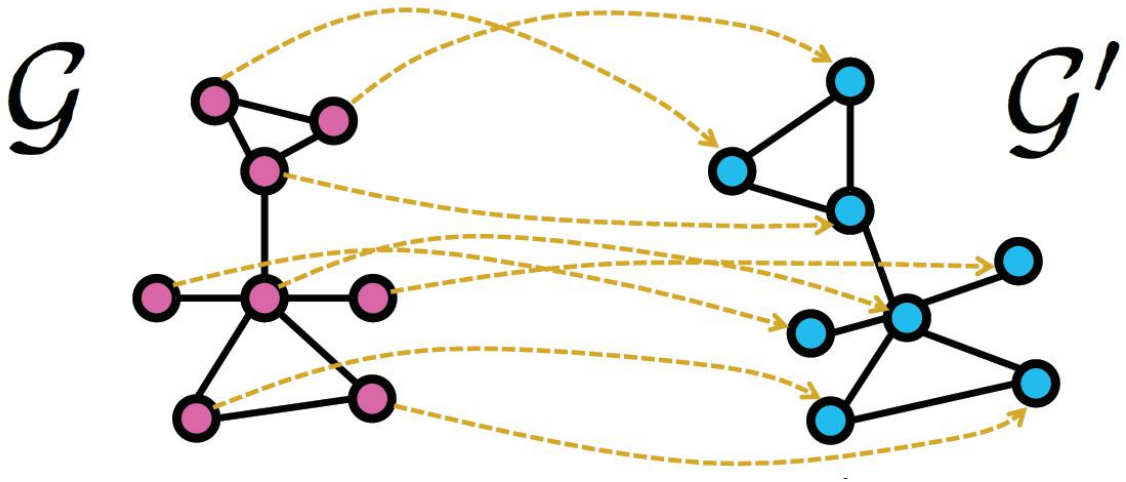
- Not robust to outlier and noise;
- High computational cost (at least $O(m^2n^2)$);
- Optimal assignment not reflect ground truth.

} Learning on graph matching!

Outline

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

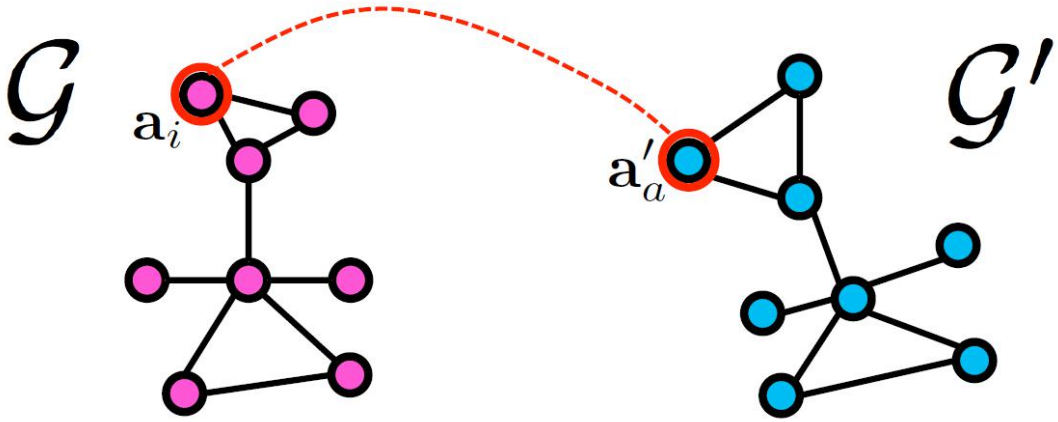
Correspondence



$\pi \in \{0,1\}^{nn'}$ represent correspondence.

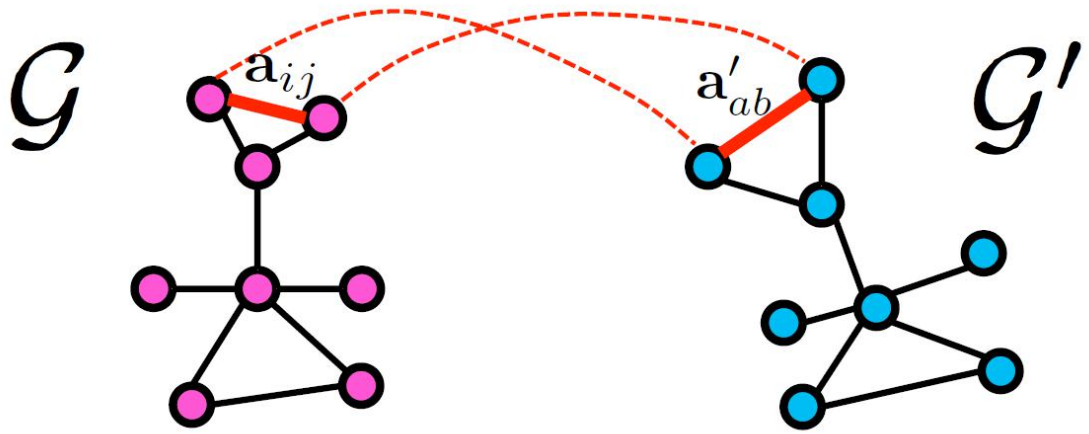
- $\pi_{ia} = 1$ if there is correspondence between i and a;
- $\pi_{ia} = 0$ otherwise.

1st Order Matching Score



$\boldsymbol{s}_V(a_i, a'_a)$: node to node matching score (1st order affinity).

2nd Order Matching Score



$\boldsymbol{s}_E(a_{ij}, a'_{ab})$: edge to edge matching score (2nd order affinity).

Matching Score

A unified formulation of affinity:

$$\Phi(\mathcal{G}, \mathcal{G}', \pi) = [\dots; \boldsymbol{s}_V(a_i, a'_{\pi(i)}); \dots; \boldsymbol{s}_E(a_{ij}, a'_{\pi(i)\pi(j)}); \dots]$$

node affinity　　edge affinity

Given correspondence π, the overall matching score:

$$S(\mathcal{G}, \mathcal{G}', \pi; \omega) = \langle \omega, \Phi(\mathcal{G}, \mathcal{G}', \pi) \rangle$$

The matching function

$$g_\omega(\mathcal{G}, \mathcal{G}') = \operatorname*{argmax}_\pi S(\mathcal{G}, \mathcal{G}', \pi; \omega)$$

Equivalent to graph matching without learning if $\omega = \mathbf{1}$:

$$\begin{aligned} g(\mathcal{G}, \mathcal{G}') &= \operatorname*{argmax}_\pi S(\mathcal{G}, \mathcal{G}', \pi; \mathbf{1}) \\ &= \operatorname*{argmax}_\pi \sum\nolimits_i \boldsymbol{s}_V(a_i, a'_{\pi(i)}) + \sum\nolimits_{i,j} \boldsymbol{s}_E(a_{ij}, a'_{\pi(i)\pi(j)}) \end{aligned}$$

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:

$$L = \underbrace{\frac{1}{N} \sum_{i,j} \Delta\left(g_\omega(\mathcal{G}_i, \mathcal{G}_j), \mathrm{y}^{gt}\right)}_{\text{empirical term}} + \underbrace{\lambda \Omega(\omega)}_{\text{regularization}}$$

Caetano et al., "Learning Graph Matching." TPAMI 2009.

Learning Graph Structure

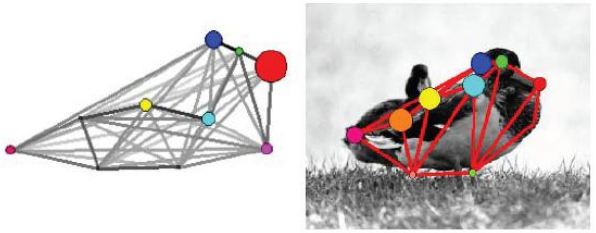
A **reference graph model** ($\mathcal{G}$) is learned for each category of object.

Matching function

reference graph

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

- **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.

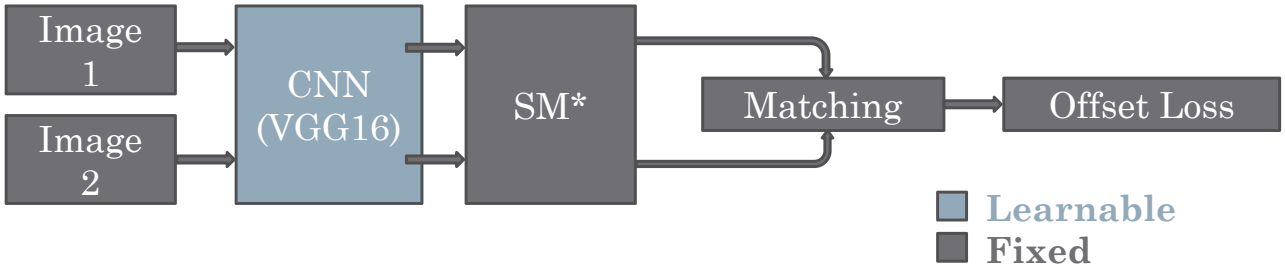


(c) a learned graph model and its matching

Cho, Alahari, and Ponce, "Learning Graphs to Match." ICCV 2013.

Learning Matching Features: Deep Learning of Graph Matching

Combination of CNN and graph matching.

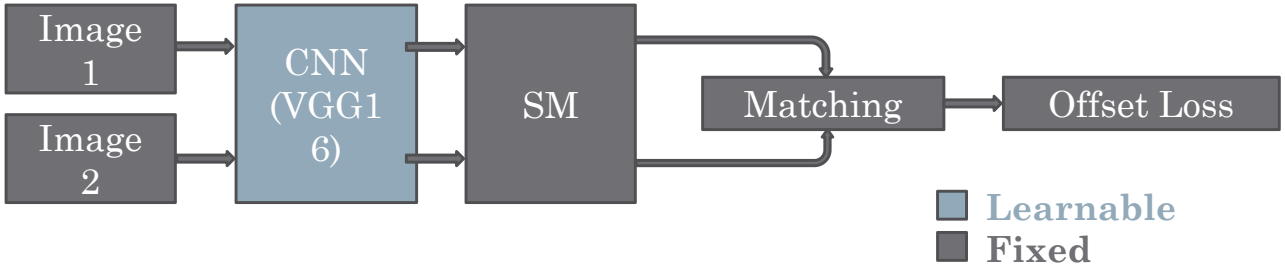


* Leordeanu and Hebert, "A Spectral Technique for Correspondence Problems Using Pairwise Constraints." ICCV2005.

- **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

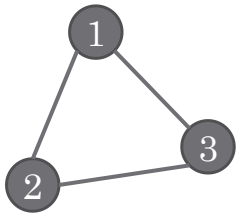


Limitations

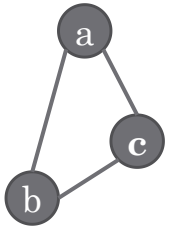
- Performance bottleneck (SM is not a sota GM solver);
- High computational cost;
- Offset loss failed to exploit the combinatorial nature of graph matching;
- Failed to model matching in deep learning (only CNN features are learned).

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

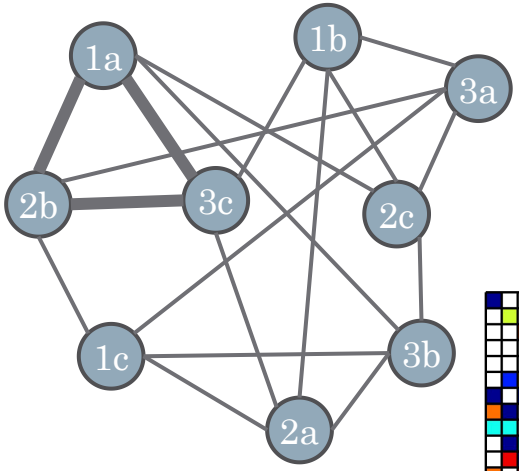
Spectral Matching (SM)



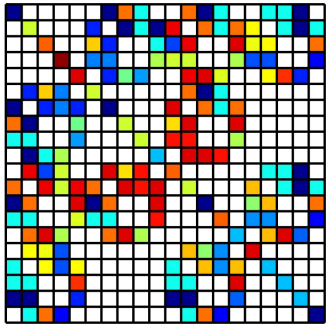
Graph 1



Graph 2



Association Graph



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

Spectral Matching (SM)

Matching of two graphs:

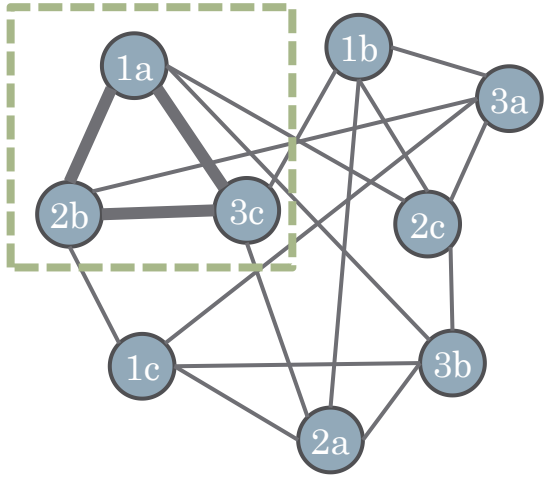
find **clustering** in association graph

(Still NP-hard)

Solved approximately via spectral method:

$$\max_{x} x^T K x, \qquad s.t. \|x\|^2 = 1$$

Equivalent to finding maximum eigenvalue of K.



Association Graph

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

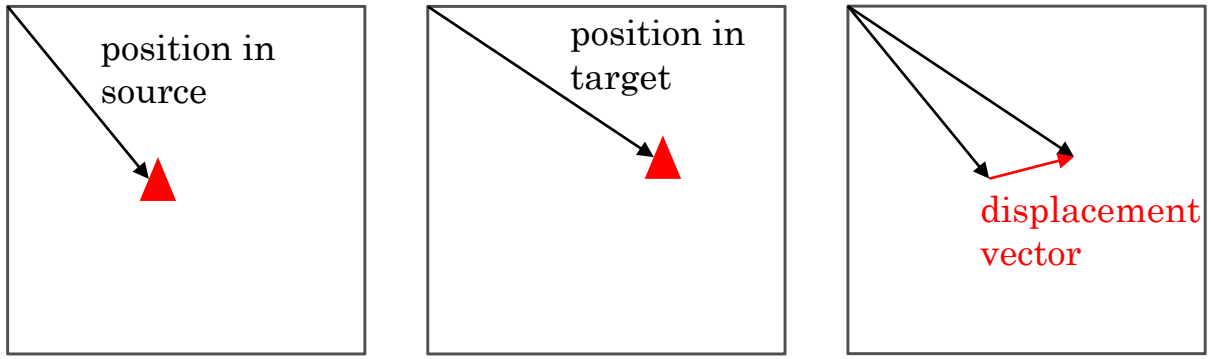
Permutation is not regression

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

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.

Offset Loss (regression)

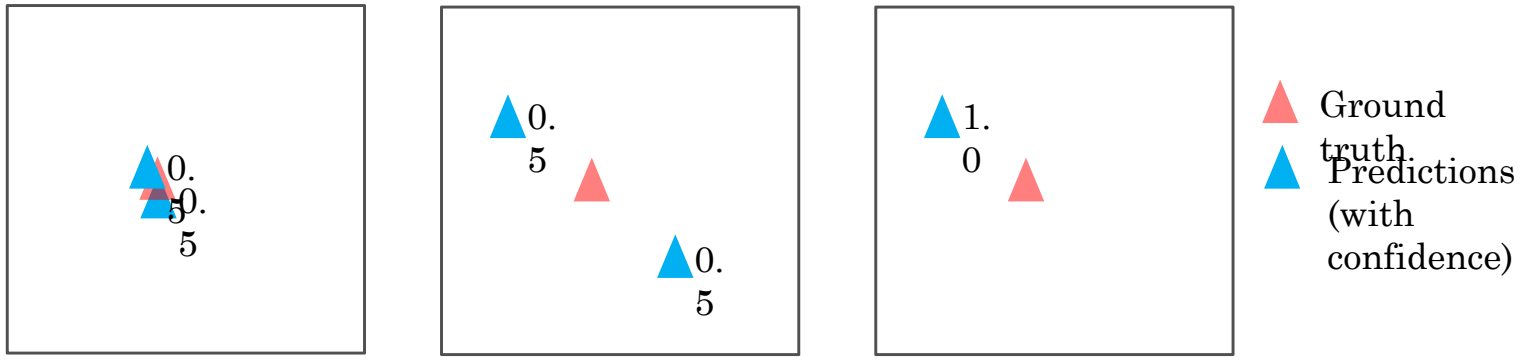
The offset loss adopted, is based on a metric named "**displacement vector**"



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

Offset Loss (regression)

Offset loss enforce the sum of prediction as close to ground truth as possible.



Low loss
good prediction

Low loss
bad prediction

High loss
bad prediction

Outline

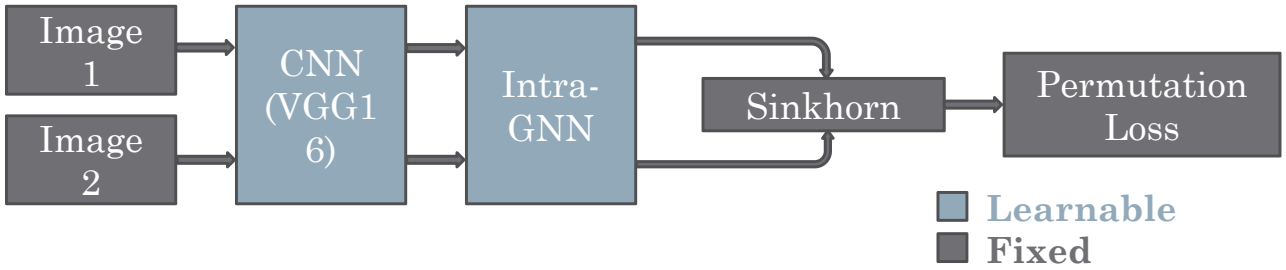
- Introduction
- Background: Learning on Graph Matching
- **Embedding approach for Deep Graph Matching**
- Other techniques
- Outlook

Learning Combinatorial Embedding Networks for Deep Graph Matching

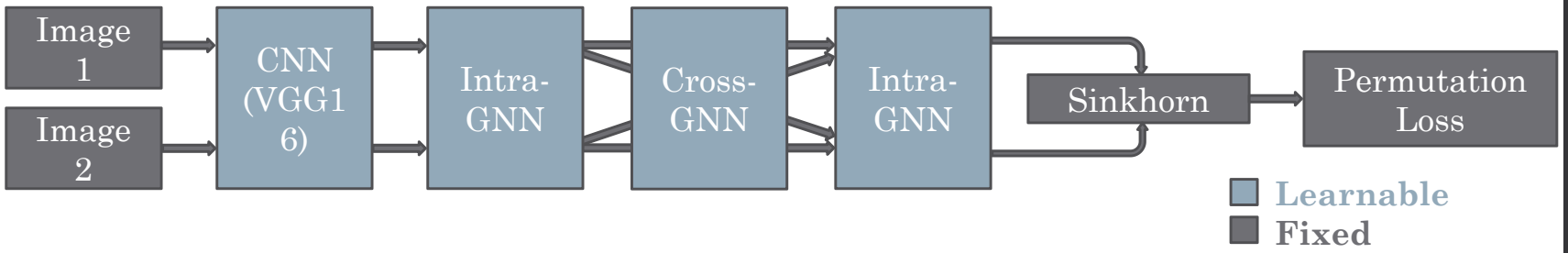
Runzhong Wang, Junchi Yan* and Xiaokang Yang

ICCV'19 Oral https://arxiv.org/abs/1904.00597

Overview



Permutation loss and **I**ntra-graph **A**ffinity based graph matching (**PIA**)



Permutation loss and **C**ross-graph **A**ffinity based graph matching (**PCA**)

Algorithm 1: Permutation and intra/cross-graph affinity learning for graph matching (PIA/PCA)

Input: Graph pairs from $\mathcal{D}$ with node correspondence; initial model weights $\mathbf{W}$; learning rate lr.

repeat
  // Draw image pair and ground truth from dataset, $s = 1, 2$ denoting two input images
  $\{(I_s, \{P_{si}\}_{i\in\mathcal{V}_s}, \mathbf{A}_s)|s = 1, 2\}, \mathbf{S}^{gt} \in \mathcal{D}$;
  // extract CNN features Eq. (4)
  $\{h_{si}^{(0)}\} \leftarrow \text{Interp}(\{P_{si}\}, \text{CNN}(I_s))$;
  if *PIA* **then**
    // intra-graph aggregation Eq. (5, 6, 7)
    for $k \leftarrow \{1, 2, 3\}$ **do**
      $\{\mathbf{h}_{si}^{(k)}\} \leftarrow \text{GConv}_k(\mathbf{A}_s, \{\mathbf{h}_{si}^{(k-1)}\})$;

$$\mathbf{m}_{si}^{(k)} = \frac{1}{|(i,j) \in \mathcal{E}_s|} \sum_{j:(i,j)\in\mathcal{E}_s} f_{msg}(\mathbf{h}_{sj}^{(k-1)}) \quad (5)$$

$$\mathbf{n}_{si}^{(k)} = f_{node}(\mathbf{h}_{si}^{(k-1)}) \quad (6)$$

$$\mathbf{h}_{si}^{(k)} = f_{update}(\mathbf{m}_{si}^{(k)}, \mathbf{n}_{si}^{(k)}) \quad (7)$$

$$\{\mathbf{h}_{si}^{(k+1)}\} = \text{GConv}(\mathbf{A}_s, \{\mathbf{h}_{si}^{(k)}\}), \quad i \in \mathcal{V}_s \quad (8)$$

  if *PCA* **then**
    // 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)
    $\{\mathbf{h}_{si}^{(3)}\} \leftarrow \text{GConv}_2(\mathbf{A}_s, \{\mathbf{h}_{si}^{(2)}\})$;
  // correspondence prediction Eq. (15, 18)
  build $\mathbf{M}^{(0)}$ from $\{\mathbf{h}_{1i}^{(3)}\}, \{\mathbf{h}_{2j}^{(3)}\}$ by Eq. (15);
  $\mathbf{S} \leftarrow \text{Sinkhorn}(\mathbf{M}^{(0)})$;
  // end-to-end training based on Eq. (19)
  $\mathbf{W} \leftarrow -lr \times \frac{\partial L_{perm}(\mathbf{S}, \mathbf{S}^{gt})}{\partial \mathbf{W}} + \mathbf{W}$;
until *convergence*;
Output: learned model weights $\mathbf{W}$.

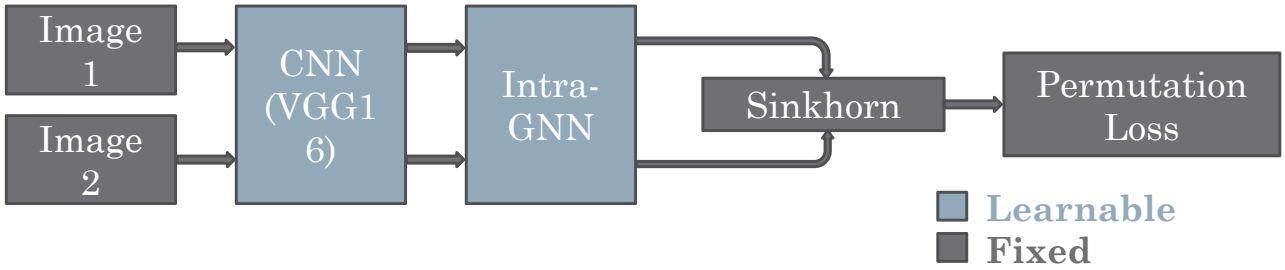
$$\mathbf{m}_{1i}^{(k)} = \sum_{j\in\mathcal{V}_2} \hat{\mathbf{S}}_{i,j} f_{msg\text{-}cross}(\mathbf{h}_{2j}^{(k-1)}) \quad (10)$$

$$\mathbf{n}_{1i}^{(k)} = f_{node\text{-}cross}(\mathbf{h}_{1i}^{(k-1)}) \quad (11)$$

$$\mathbf{h}_{1i}^{(k)} = f_{update\text{-}cross}(\mathbf{m}_{1i}^{(k)}, \mathbf{n}_{1i}^{(k)}) \quad (12)$$

$$\begin{aligned}\{\mathbf{h}_{1i}^{(k+1)}\}_{i\in\mathcal{V}_1} &= \text{CrossConv}(\hat{\mathbf{S}}, \{\mathbf{h}_{1i}^{(k)}\}_{i\in\mathcal{V}_1}, \{\mathbf{h}_{2j}^{(k)}\}_{j\in\mathcal{V}_2}) \\ \{\mathbf{h}_{2j}^{(k+1)}\}_{j\in\mathcal{V}_2} &= \text{CrossConv}(\hat{\mathbf{S}}^\top, \{\mathbf{h}_{2j}^{(k)}\}_{j\in\mathcal{V}_2}, \{\mathbf{h}_{1i}^{(k)}\}_{i\in\mathcal{V}_1})\end{aligned} \quad (13)$$

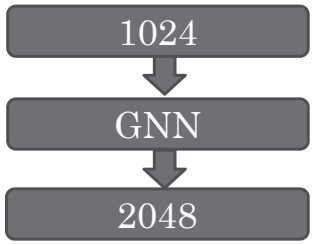
PIA-GM: Permutation loss & Intra-graph Affinity



- 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

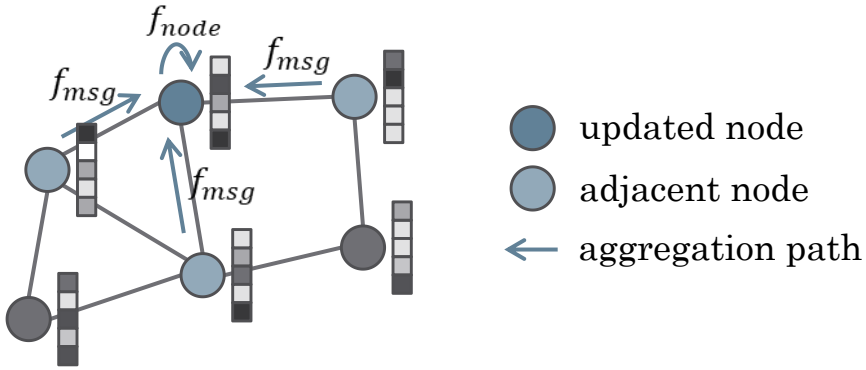
Output vector for each feature point



Intra-GNN

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

Feature aggregated from adjacent nodes.



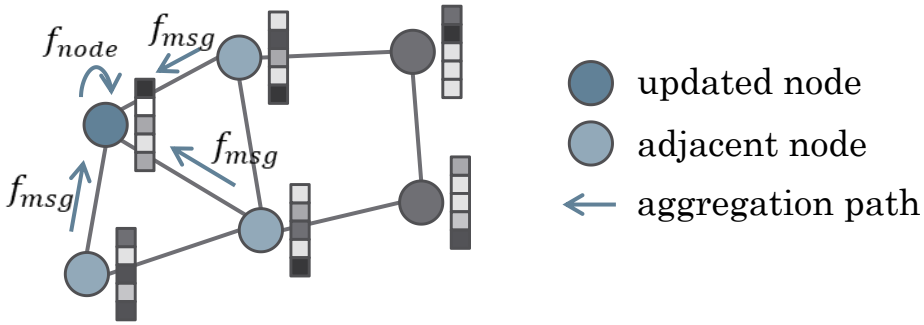
Graph structure → Embedding vectors.

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.



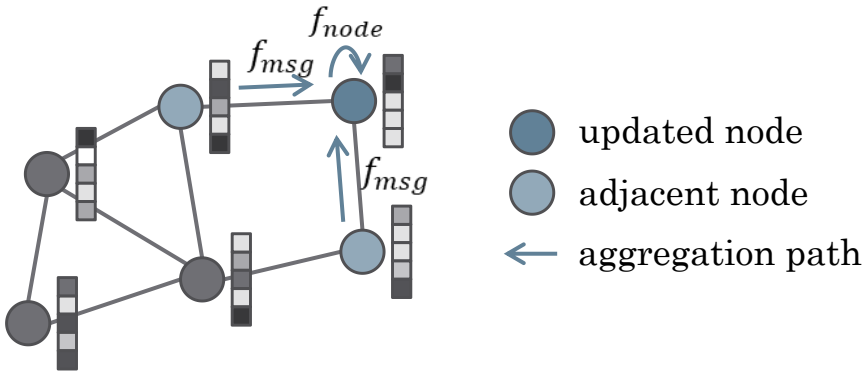
Iterate over all nodes.

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

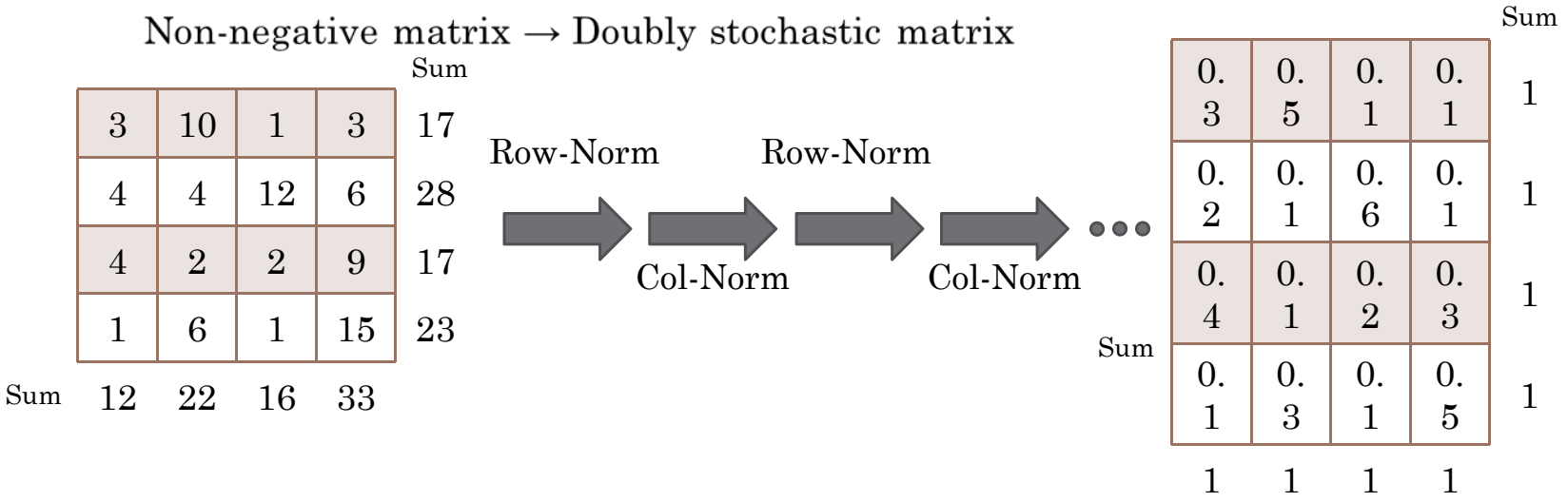
The similarity (distance) between two vectors from the feature space of the output of GNN, is learned.

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

Where **A** is the matrix with learnable weights.

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

Sinkhorn Algorithm



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

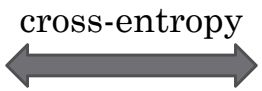
- k: number of iterations; $n \times m$: matrix size

Sinkhorn and Knopp, "Concerning Nonnegative Matrices and Doubly Stochastic Matrices."

Permutation Loss

0.3	0.5	0.1	0.1
0.2	0.1	0.6	0.1
0.4	0.1	0.2	0.3
0.1	0.3	0.1	0.5

prediction S



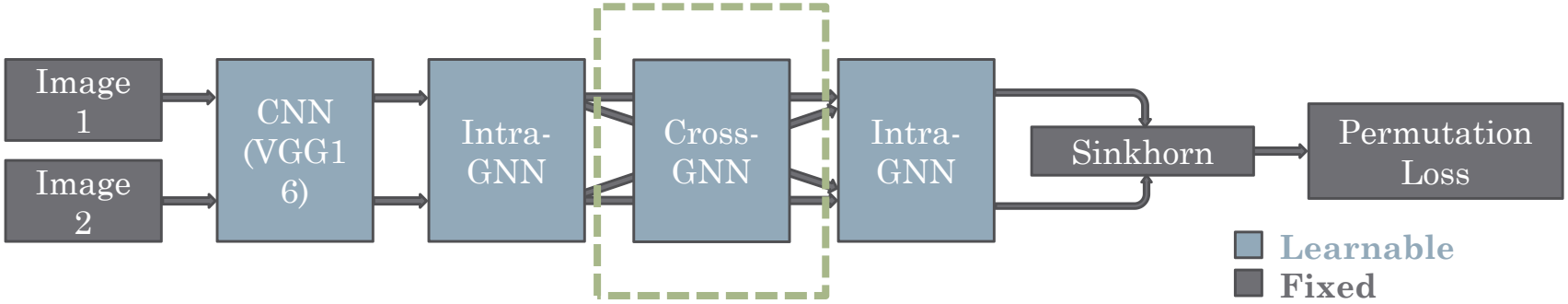
1	0	0	0
0	0	1	0
0	1	0	0
0	0	0	1

ground truth S^{gt}

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

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

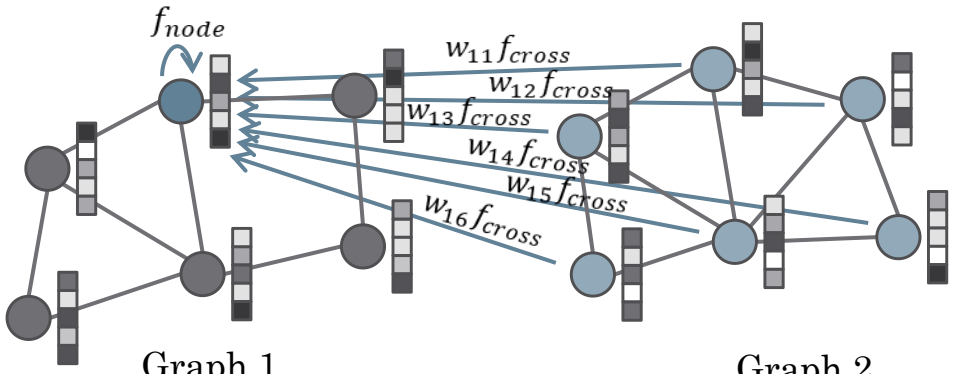
PCA-GM: Permutation loss & Cross-graph Affinity



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

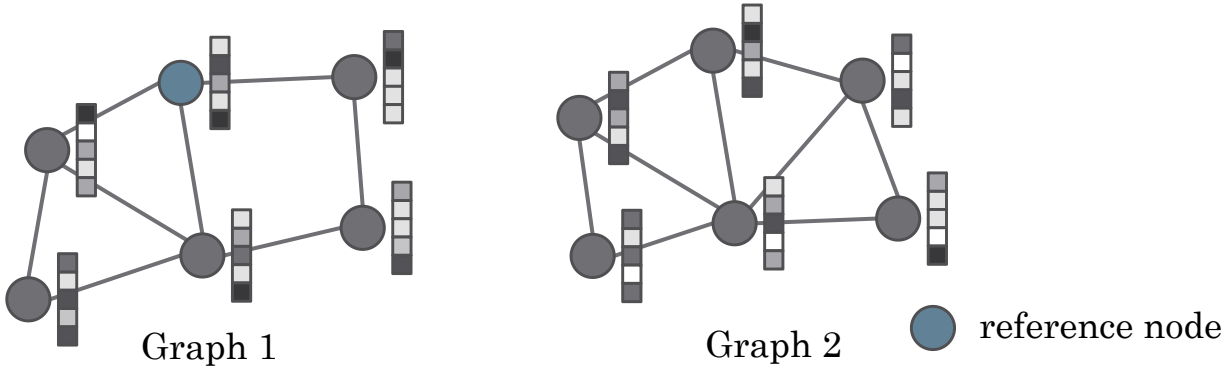
Cross-GNN

Features are aggregated from nodes with similar features across graphs.



Similarity Prediction

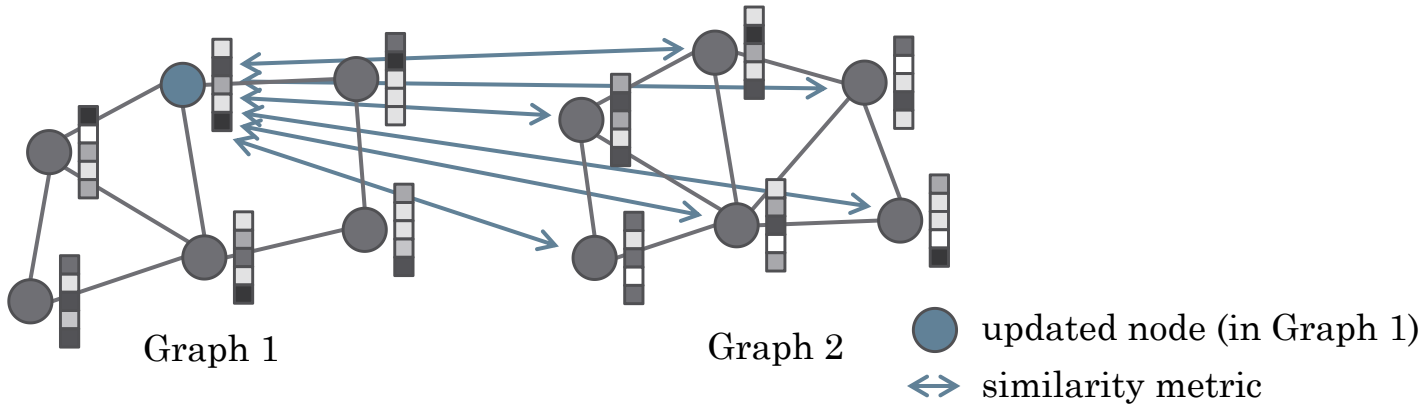
The cross-graph aggregation path is defined by cross-graph similarity, computed from feature vectors from shallower layers.



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

Similarity Prediction

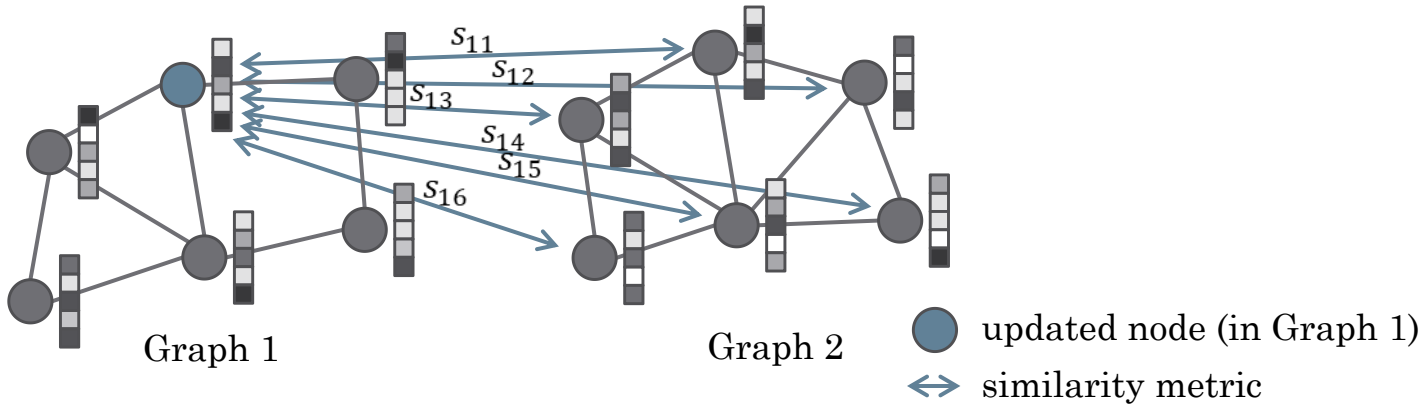
The cross-graph aggregation path is defined by cross-graph similarity, computed from feature vectors from shallower layers.



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

Similarity Prediction

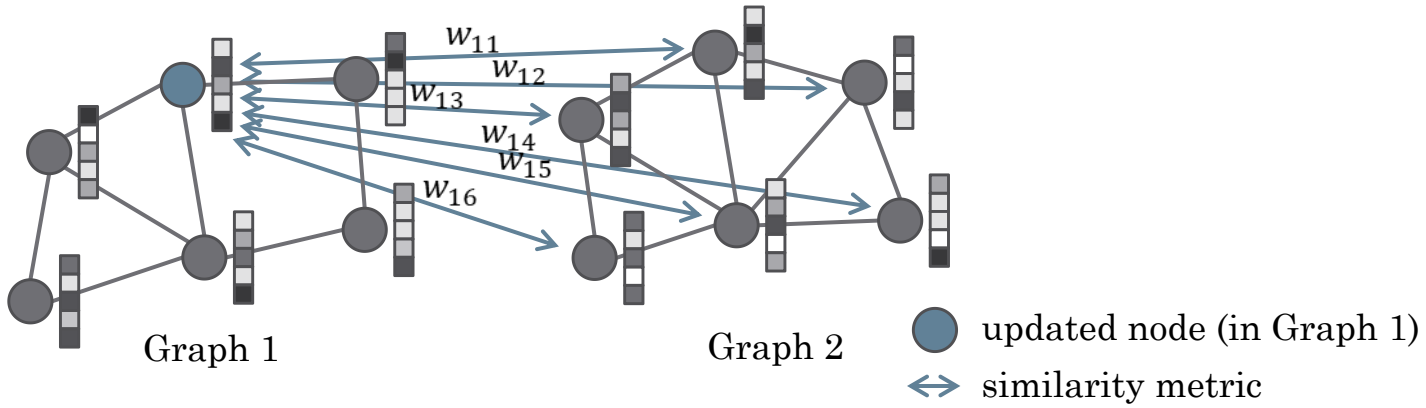
The cross-graph aggregation path is defined by cross-graph similarity, computed from feature vectors from shallower layers.



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

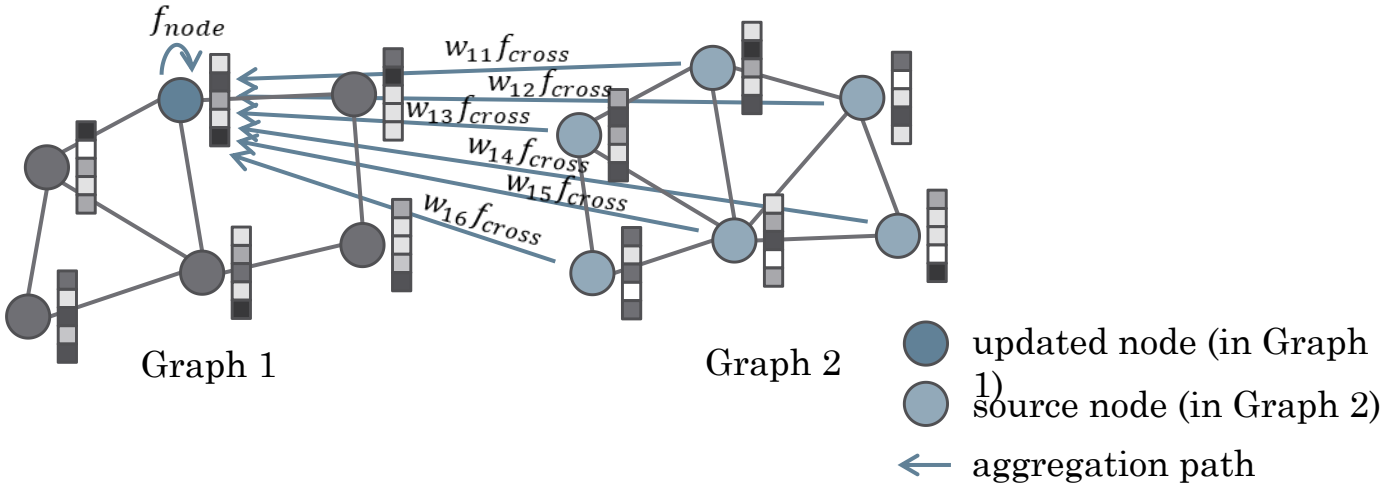
Similarity Prediction

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.

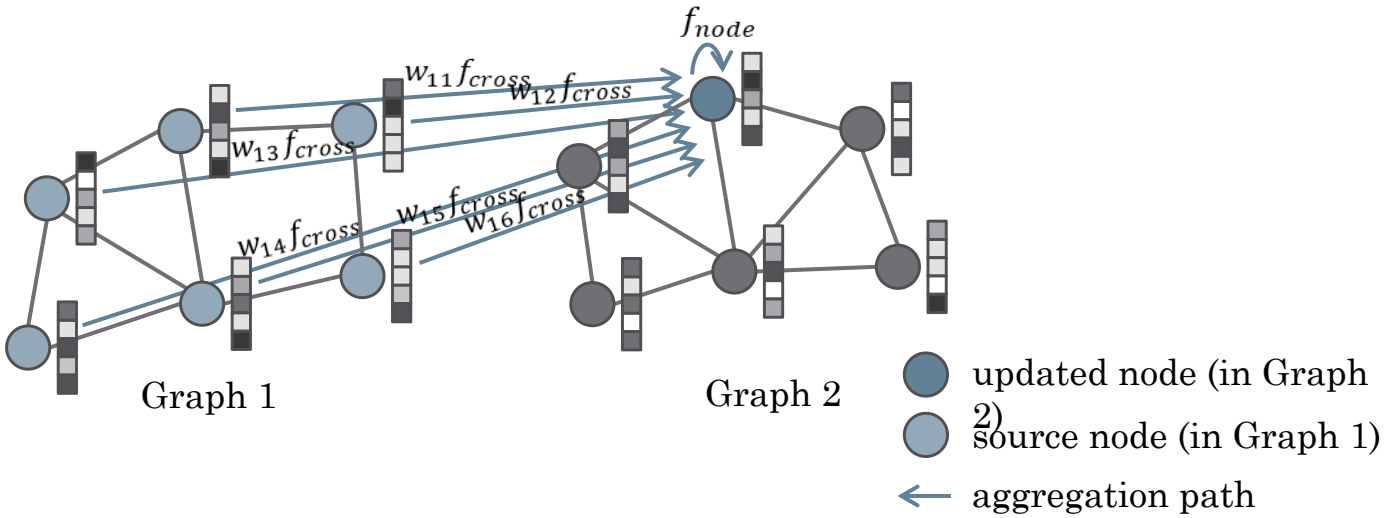
Cross-graph Aggregation



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

Self-loop is reserved in cross-graph aggregation.

Cross-graph Aggregation



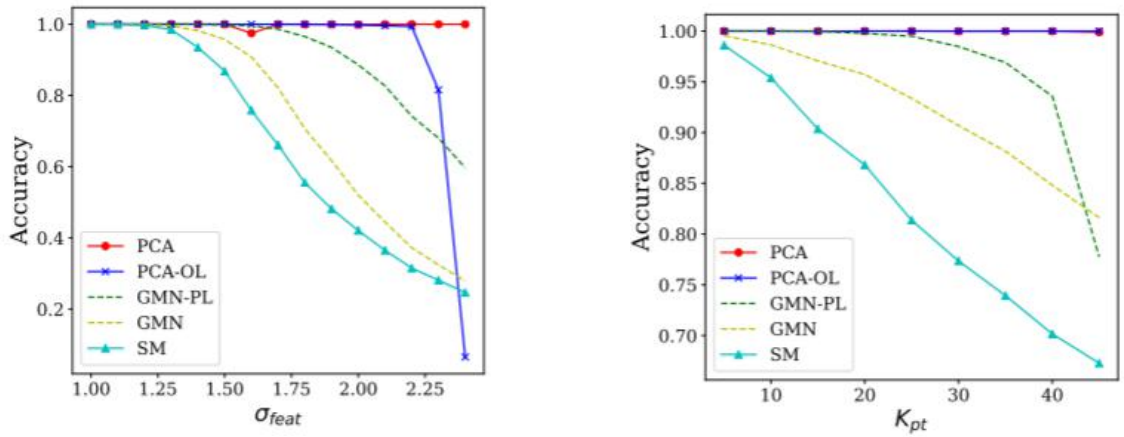
// first intra-graph aggregation Eq. (5, 6, 7)
$\{\mathbf{h}_{si}^{(1)}\} \leftarrow \text{GConv}_1(\mathbf{A}_s, \{\mathbf{h}_{si}^{(0)}\})$;
// similarity prediction Eq. (15, 18)
build $\hat{\mathbf{M}}^{(0)}$ from $\{\mathbf{h}_{1i}^{(1)}, \mathbf{h}_{2j}^{(1)}\}$ by Eq. (15);
$\hat{\mathbf{S}} \leftarrow \text{Sinkhorn}(\hat{\mathbf{M}}^{(0)})$;
// 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.

Comparison

	GMN (Zanfir et al. CVPR2018)	PCA (Ours)
Graph Modeling	**Performance bottleneck** of SM	**Deep GNN** embedding model with cross-graph aggregation
Matching Solver	High **computational cost** of SM: $O(km^2n^2)$ time complexity	**Sinkhorn** algorithm: $O(kmn)$ time complexity
Loss Function	**Offset loss**: Weaker supervision.	**Permutation loss**: 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
PCA	**Cross-GNN**	**Perm loss**	**63.8**

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

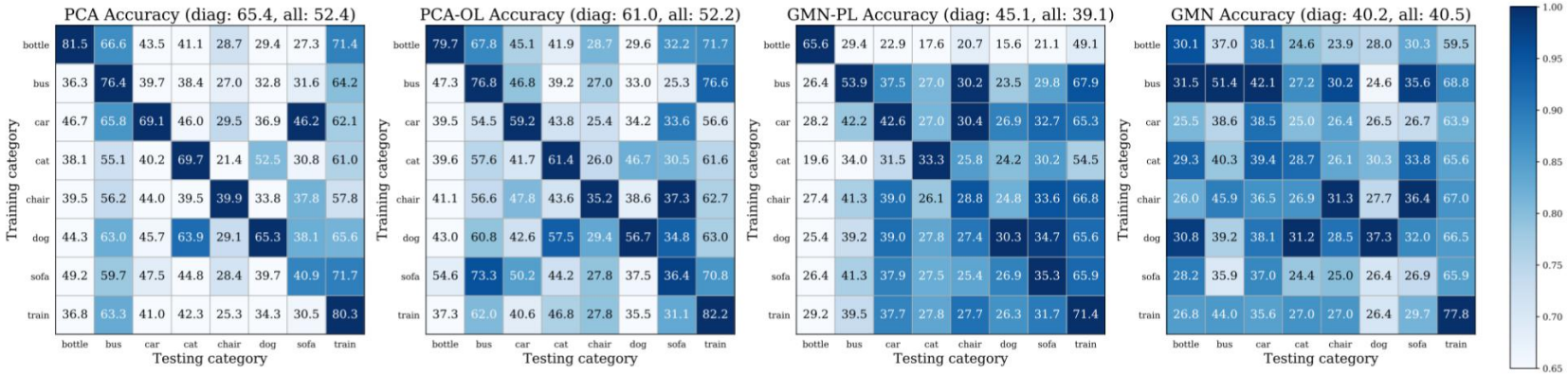
Our method outperformed non-deep learning method HARG-SSVM (Cho et al. ICCV 2013).
It also shows that the learned knowledge can be efficiently transformed among different datasets.

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	**35.5**	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	**57.7**	58.6	47.2	74.0	74.5	62.1	66.6	33.6	61.7	**65.4**	58.0	67.1	58.9	41.9	77.7	64.7	50.5	81.8	89.9	61.6
PIA	**41.5**	55.8	60.9	**51.9**	75.0	75.8	59.6	65.2	33.3	**65.9**	62.8	**62.7**	67.7	62.1	42.9	**80.2**	64.3	**59.5**	82.7	90.1	63.0
PCA	40.9	55.0	**65.8**	47.9	**76.9**	**77.9**	**63.5**	**67.4**	33.7	65.5	63.6	61.3	**68.9**	**62.8**	**44.9**	77.5	**67.4**	57.5	**86.7**	**90.9**	**63.8**

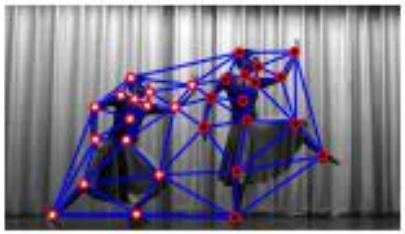
Table 2. Accuracy on Pascal VOC Keypoint. Note after replacing the offset loss by the permutation loss, the GMN-PL outperforms GMN [51] almost in all categories. While our method PIA's performance degenerates when its permutation loss is changed to offset loss.

Outline

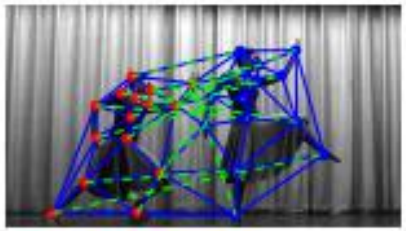
- Introduction
- Background: Learning on Graph Matching
- Embedding approach for Deep Graph Matching
- **Other techniques**
- Outlook
- Brief experience sharing

Joint matching and cuts (CVPR'18)

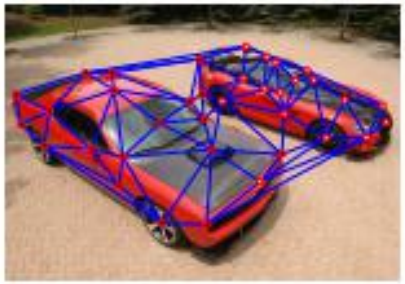
Problem formulation: Joint cuts and matching of two objects in one single image
Motivation: graph cuts aim to separate dissimilar objects, matching aims not
Limitation: the key points are assumed to be detected in prior.



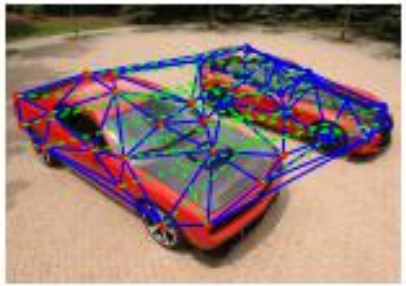
(o) CutMatch-cuts: **15/15**



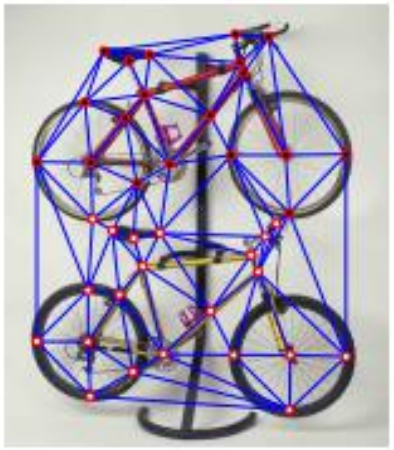
(s) CutMatch-match: **7/15**



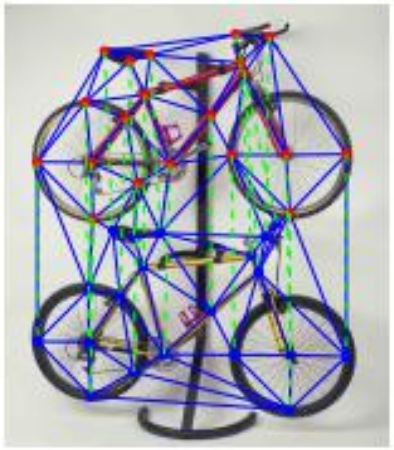
(p) CutMatch-cuts: **15/18**



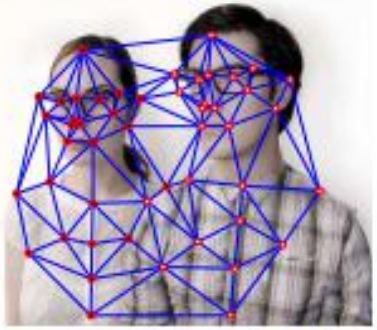
(t) CutMatch-match: **4/18**



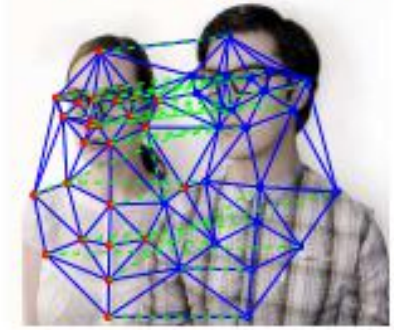
(m) CutMatch-cuts: **19/20**



(q) CutMatch-match: **20/20**



(n) CutMatch-cuts: **25/25**

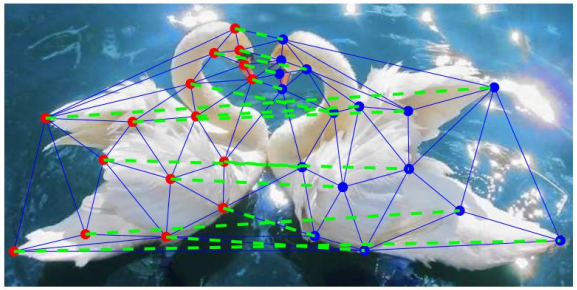


(r) CutMatch-match: **16/25**

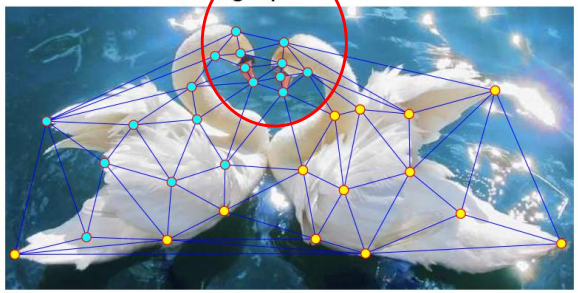
T. Yu, Junchi Yan et al, Joint Cuts and Matching of Partitions in One Graph, CVPR 2018.

Main idea (CVPR'18)

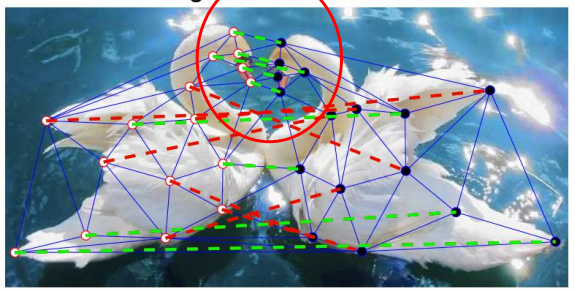
Ground-truth partitions and matchings

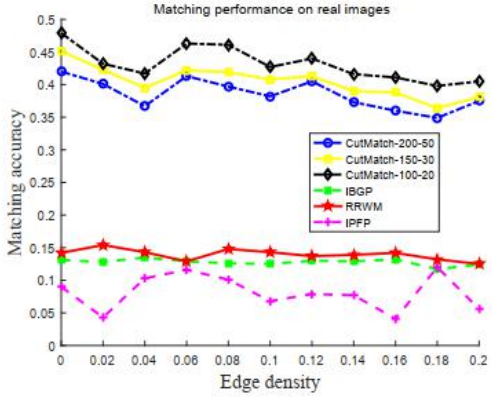


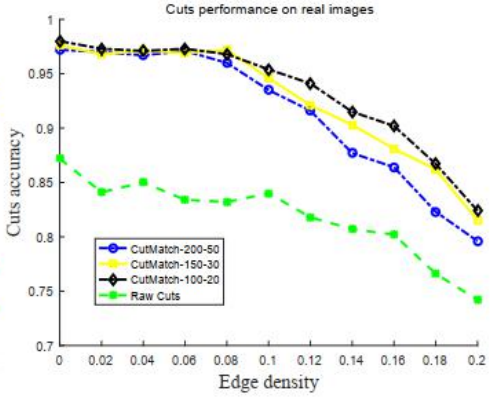
Initial graph cut result



Matching and cut with CutMatch







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

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

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

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

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

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

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

Large-scale graph matching for social networks/bioinformatics

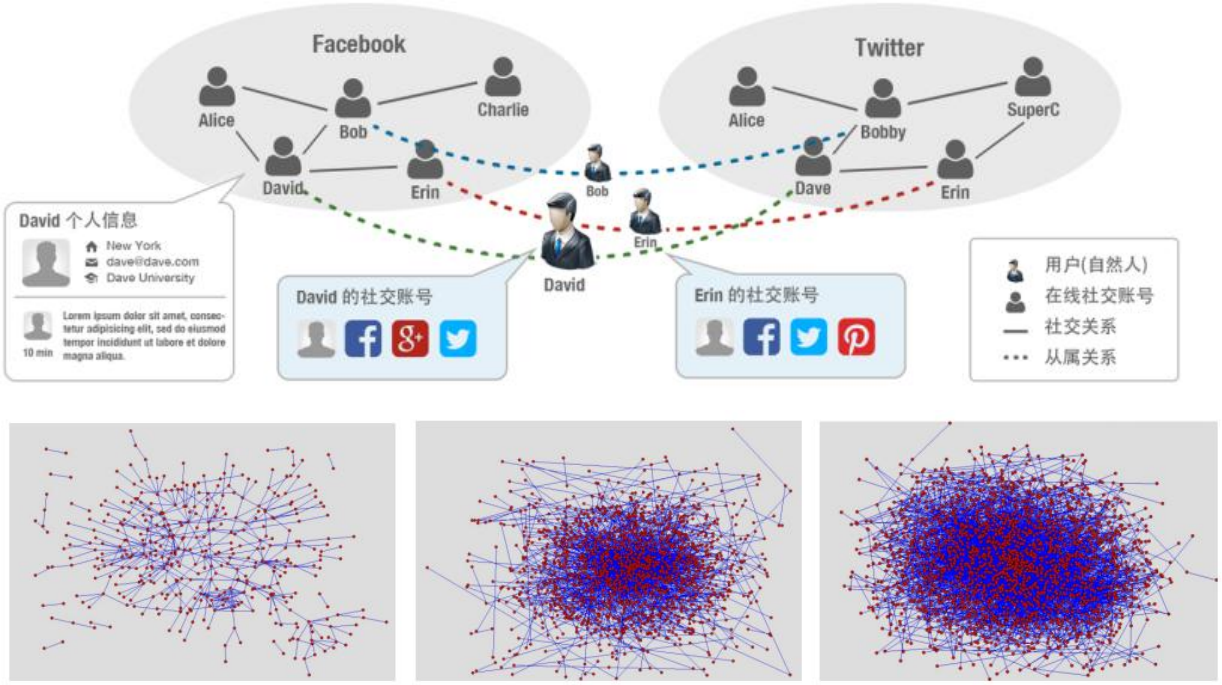


图 3-1 蛋白质相互作用网络示例图（从左至右为：果蝇、蛔虫、酵母）

Joint matching and link prediction

Problem: Joint matching and link prediction
Motivation: two can benefit each othe
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…)
……

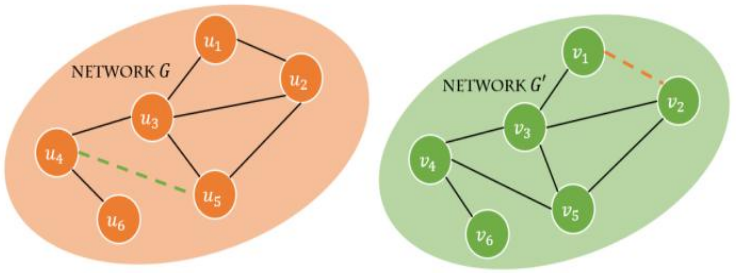


Figure 1: Illustration for how joint modeling of link prediction and network alignment can benefit to each other. Given the correspondence $v_1 = \pi(u_1), v_2 = \pi(u_2)$, it is convincing that (v_1, v_2) should be linked as (u_1, u_2) is linked. On the other hand, if (u_4, u_5) and (v_4, v_5) are not 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.

X. Du, **Junchi Yan***, H. Zha, Joint Link Prediction and Network Alignment via Cross-graph Embedding, *International Joint Conference on Artificial Intelligence (**IJCAI**), 2019*

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.

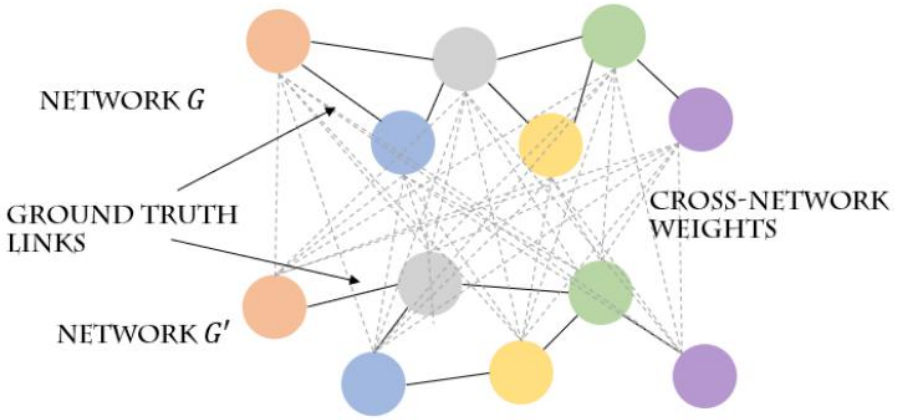


Figure 2: Weighted, undirected network $\tilde{G}$ consisting of two networks with cross-graph links among vertices in two networks.

Construct an undirected compound network

So we use a biased random walk to obtain similarity across networks.
Specifically, for each step, we will decide whether to walk on the current network or switch to another network.
For walks on the current network, it is the same as random walk.
For walks switching to another network, the walk tends to switch to the most similar ones in another network.

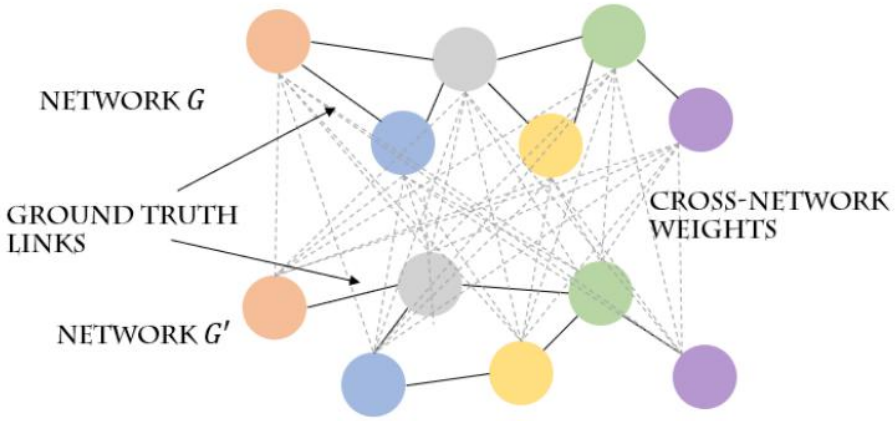
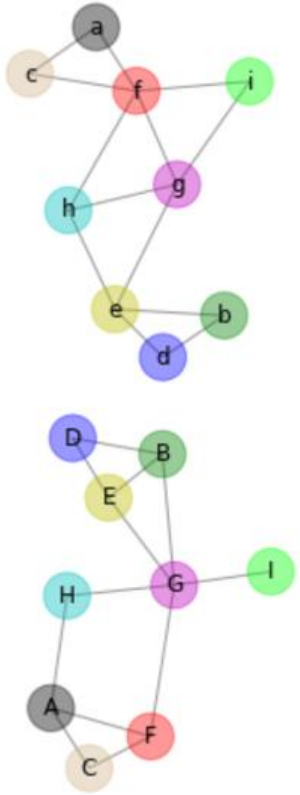
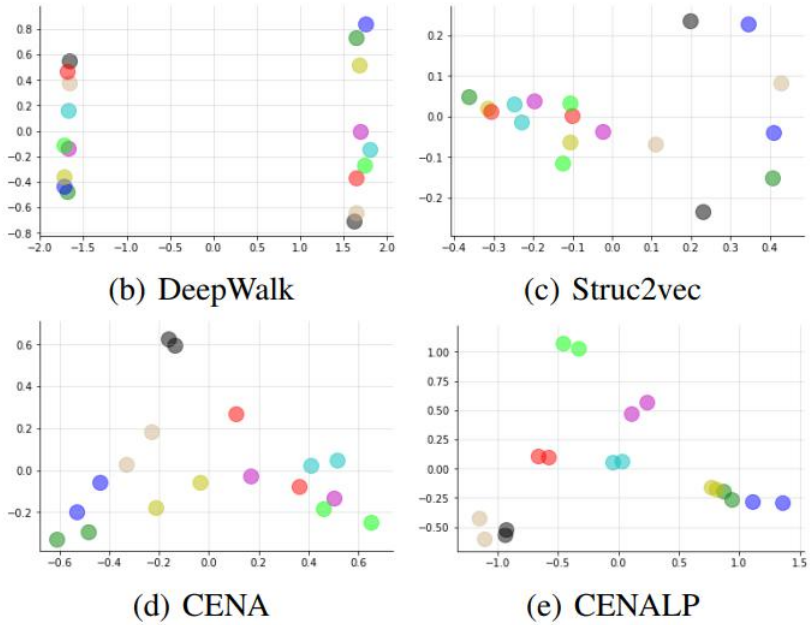


Figure 2: Weighted, undirected network $\tilde{G}$ consisting of two networks with cross-graph links among vertices in two networks.

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).

Joint Link Prediction and Network Alignment

Algorithm 3 Cross-graph Node Embedding for Joint Network Alignment and Link Prediction (CENALP)

Input: Input networks $G = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, $G' = (\mathcal{V}', \mathcal{E}', \mathcal{A}')$;
Seed vertices $\mathcal{S}$ and $\mathcal{S}'$, alignment π.

18 **while** *new vertex correspondences can be found* **do**

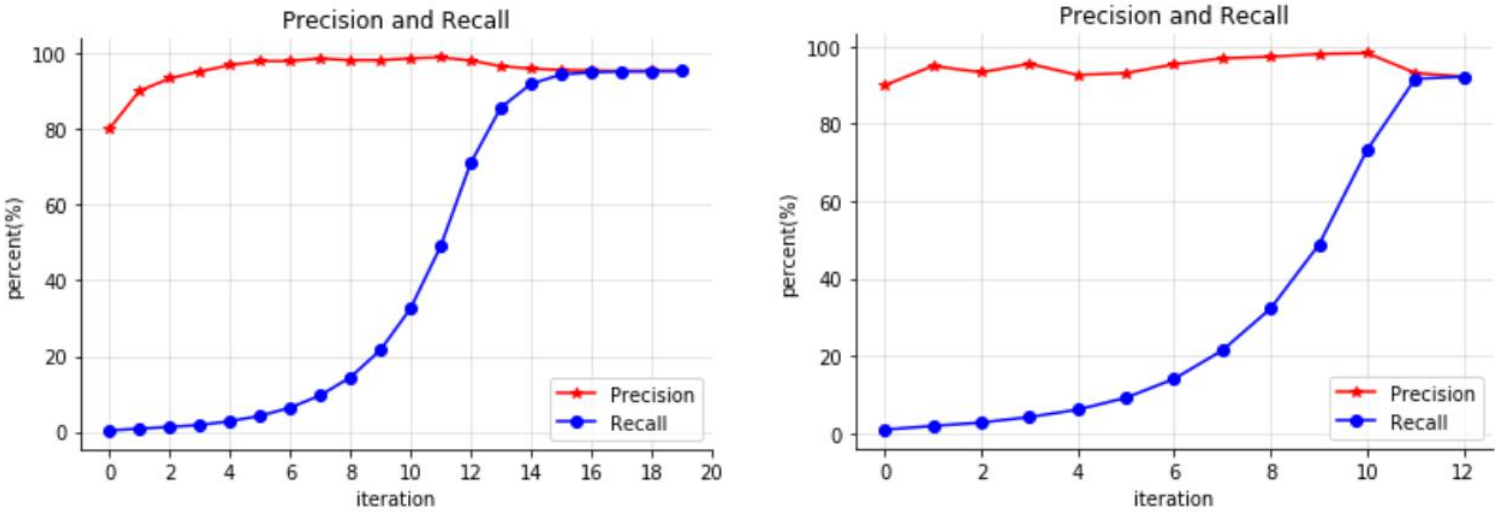
embedding — 19 Update embedding $\{\mathbf{x}_u\}_{u\in\mathcal{V}}$, $\{\mathbf{x}_v\}_{v\in\mathcal{V}'}$ by Eq. 4 to 8;
//network alignment to generate new seed vertices

network alignment — 20 Align networks and find new correspondences by Alg. 1;
//link prediction to update structures

link prediction — 21 Predict links within network for alignment by Alg. 2;

Output: Updated seed vertices $\mathcal{S}$, $\mathcal{S}'$, alignment π and node embedding $\{\mathbf{x}_u\}_{u\in\mathcal{V}}$, $\{\mathbf{x}_v\}_{v\in\mathcal{V}'}$.

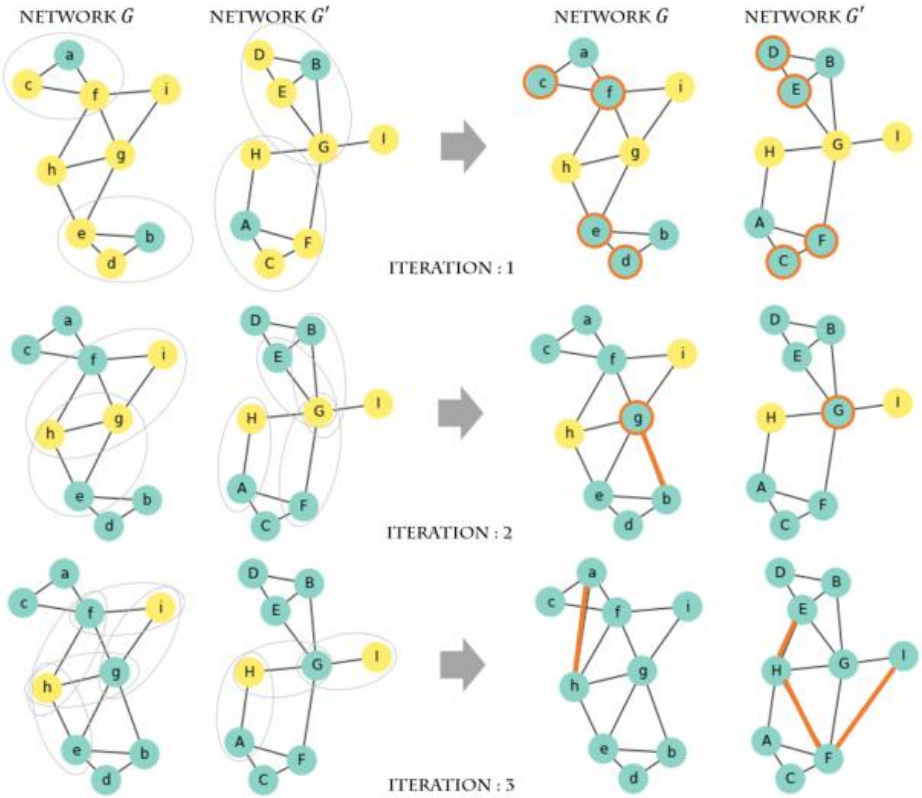
Some tricks

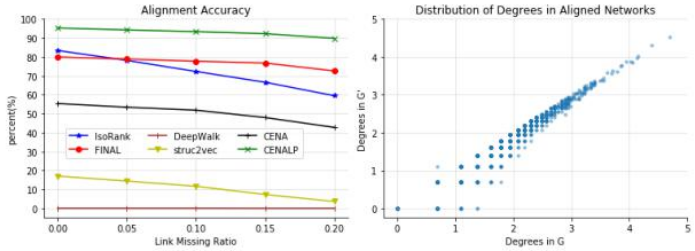


(a) DBLP (b) Facebook/Twitter

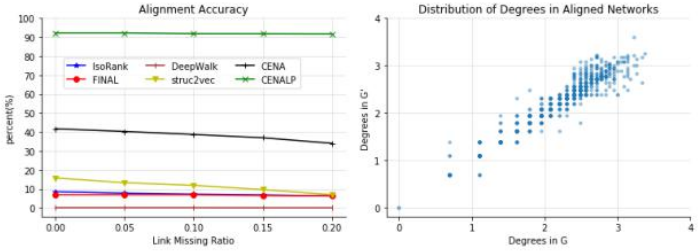
Figure 5: Precision and recall over iterations by our CENALP on DBLP and Facebook/Twitter. Note the recall grows notably.

Performance relies on distribution

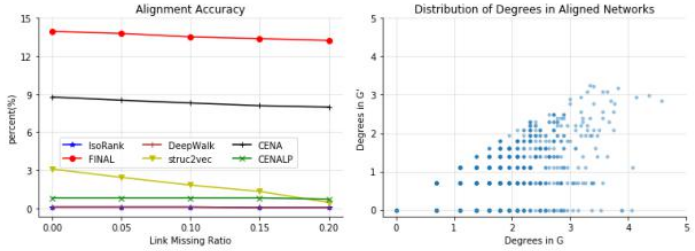




(a) DBLP and its disturbed copy



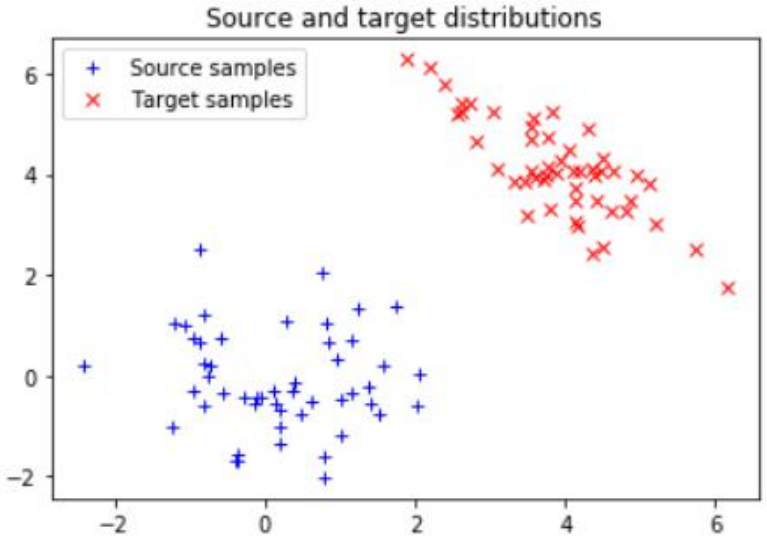
(b) Facebook/Twitter

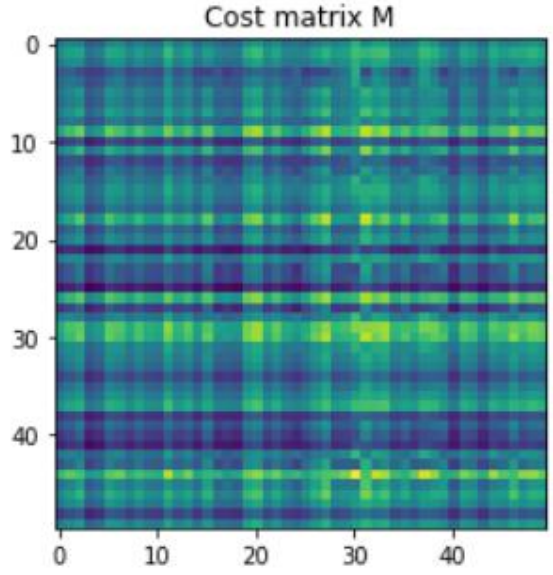


(c) Douban online/offline

Optimal transport example

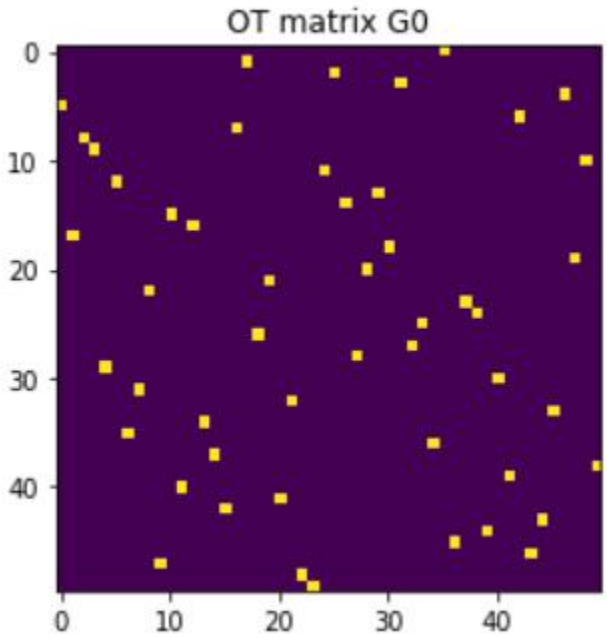
- optimal transport problem between two distributions
- cost matrix

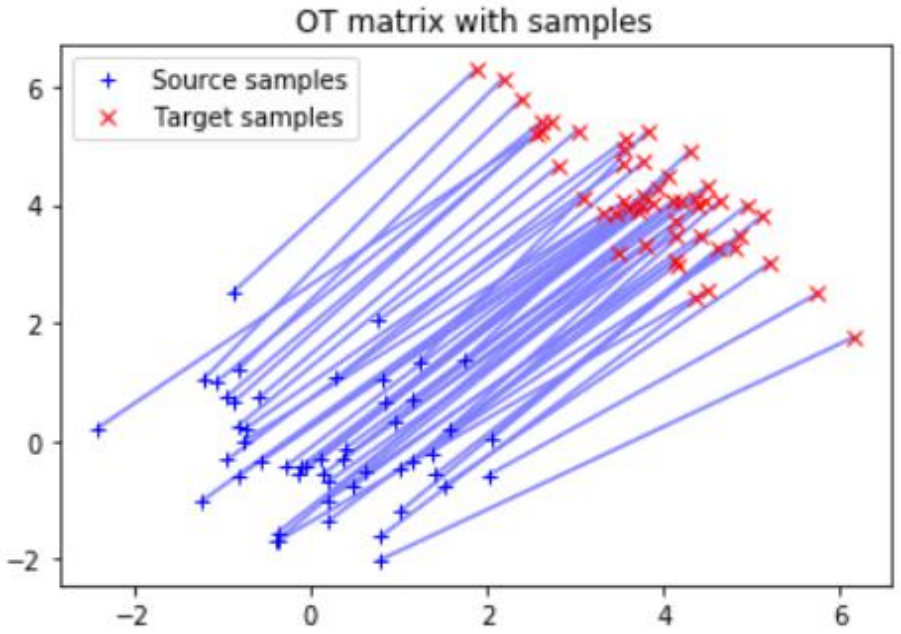




Optimal transport example

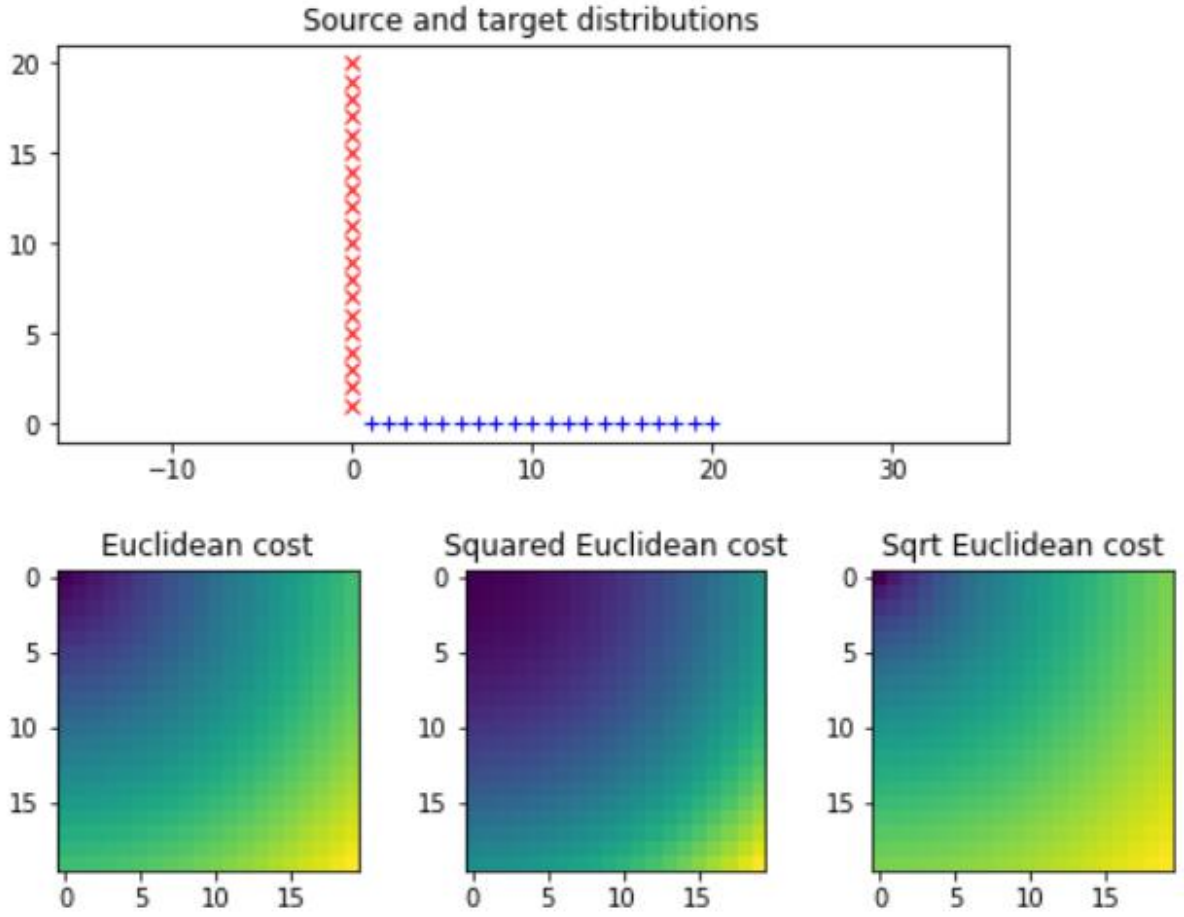
- solution





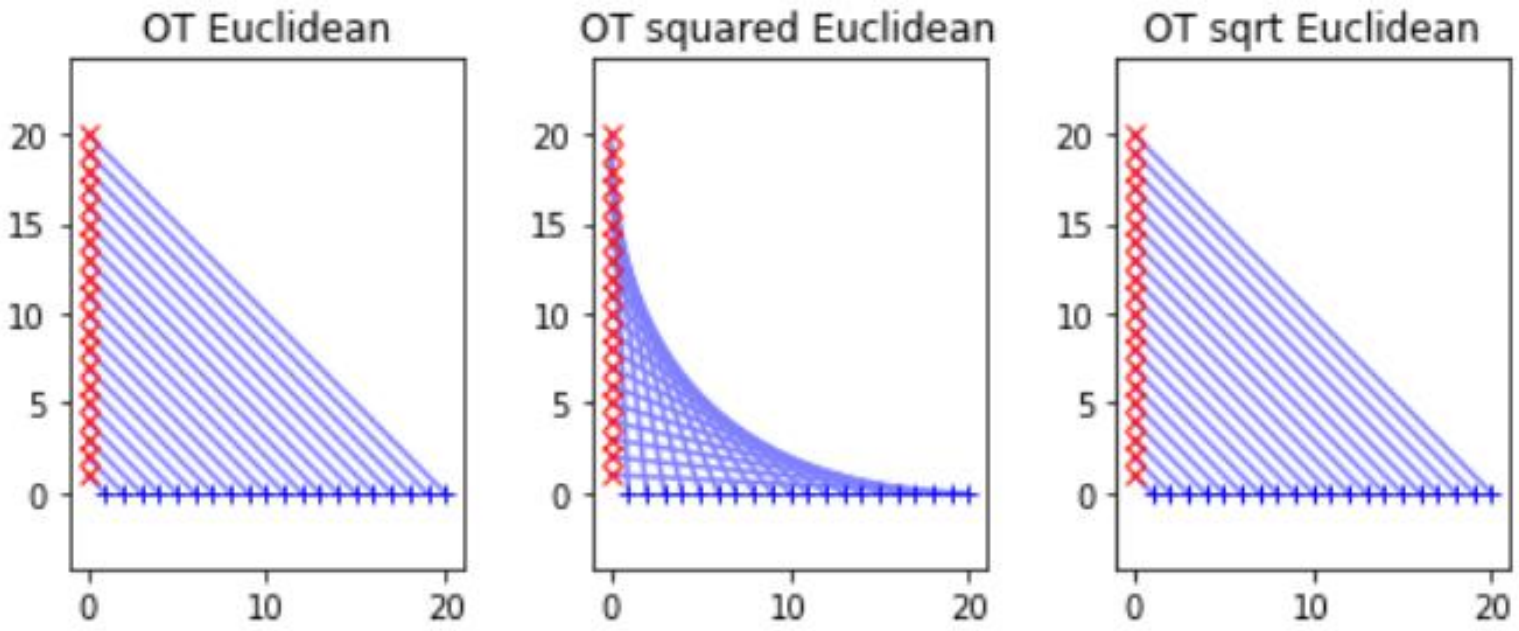
2D Optimal transport for different metrics

- Dataset1: uniform sampling



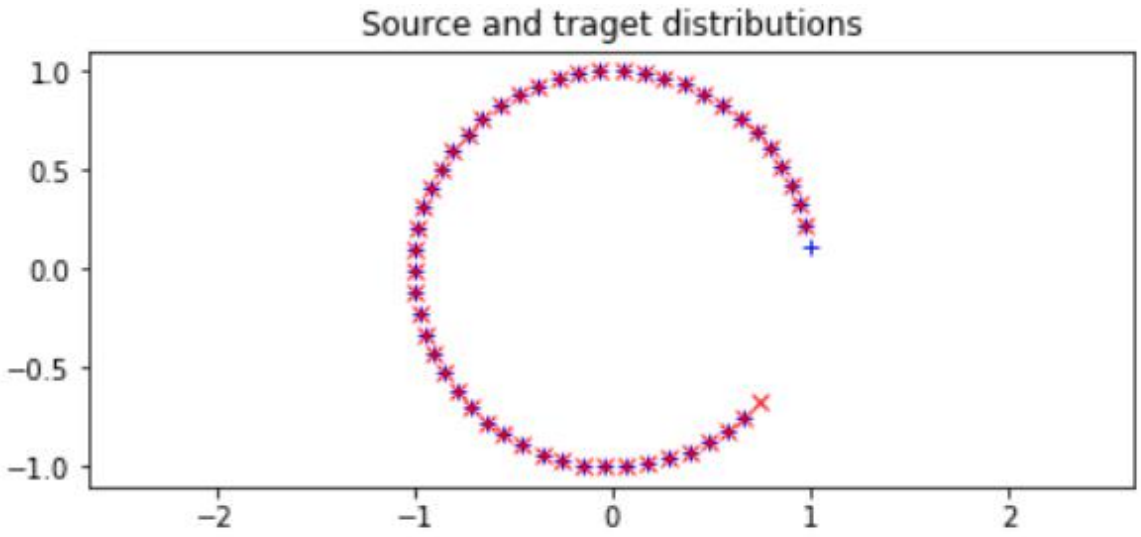
2D Optimal transport for different metrics

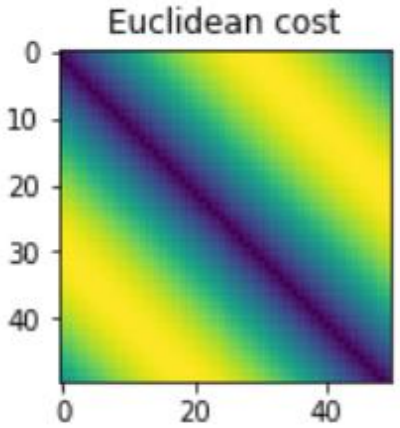
- Solution

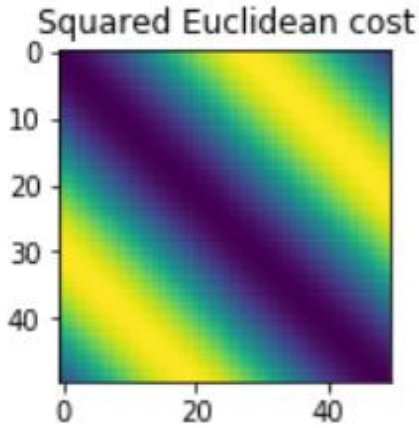


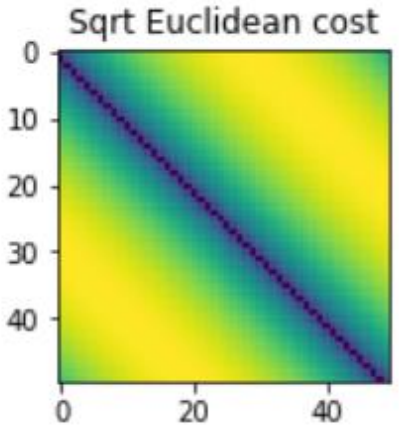
2D Optimal transport for different metrics

- Partial circle



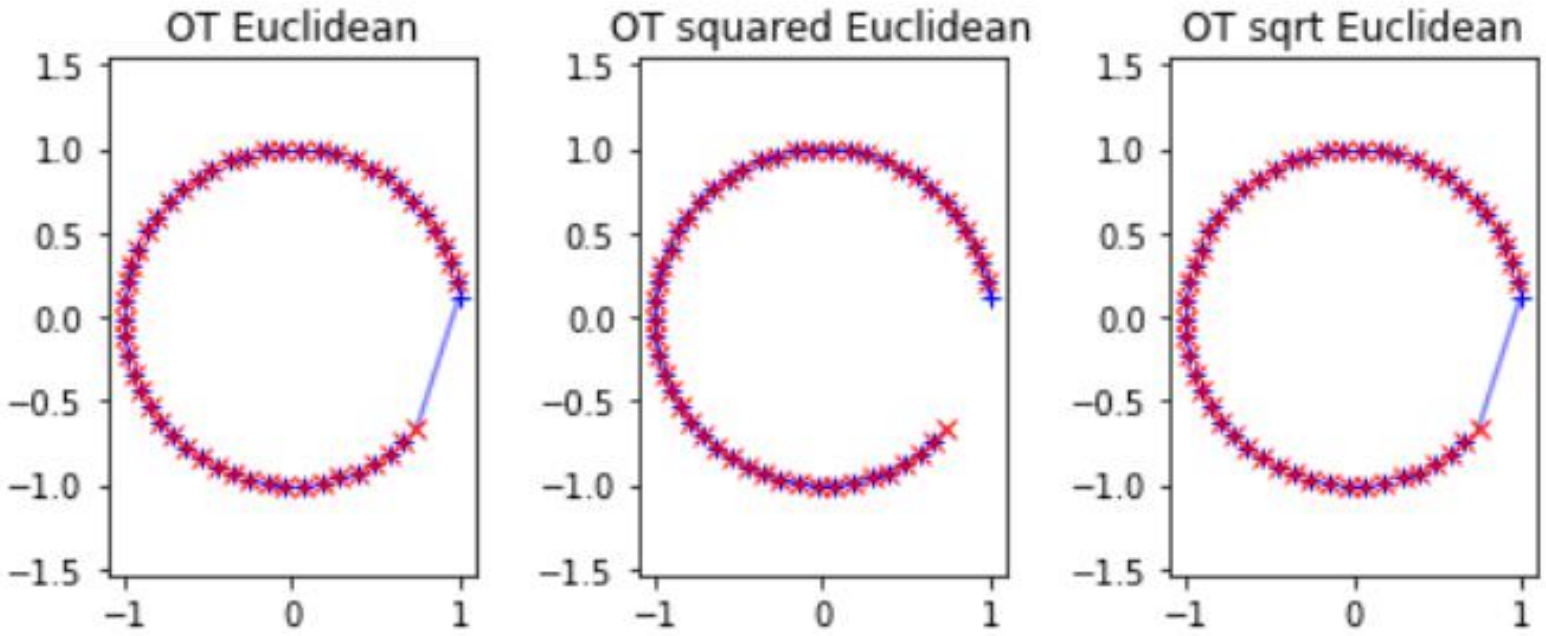






2D Optimal transport for different metrics

- Solution



Outline

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

Summary

- Introduction on graph matching
- Multiple graph matching
- Learning of graph matching
- Embedding based deep graph matching pipeline
- Other techniques
  - Joint graph matching and graph cut
  - 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. Junchi Yan, C. Li, Y. Li, G. Cao: Adaptive Discrete Hypergraph Matching. IIEEE Transactions on Cybernetics (TCYB), 2018, 48(2): 765-779

2. 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

3. 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

4. Junchi Yan, H. Xu, H. Zha, X. Yang, S. Chu. A Matrix Decomposition Perspective to Multiple Graph Matching, IEEE International Conference on Computer Vision (ICCV), 2015.

5. Junchi Yan, S. Xiao, C. Zhang, H. Zha, W. Liu, X. Yang, S. Chu: Discrete Hyper-graph Matching, CVPR 2015

6. Junchi Yan, Y. Li, W. Liu, H. Zha, X. Yang, S. Chu: Graduated Consistency-regularized Optimization for Multi-graph Matching. ECCV 2014

7. Junchi Yan, J. Wang, H. Zha, X. Yang, S. Chu: Multi-view Point Registration via Alternating Optimization. AAAI 2017

8. Junchi Yan, Y. Tian, H. Zha, X. Yang, Y. Zhang, S. Chu: Joint Optimization for Consistent Multiple Graph Matching. ICCV 2013

9. T. Yu, Junchi Yan*, J. Zhao, B. Li: Joint Cuts and Matching of Partitions in One Graph, CVPR 2018

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

11. T. Yu, Junchi Yan, Y. Wang, W. Liu, B. Li: Generalizing Graph Matching beyond Quadratic Assignment Model. NIPS 2018

12. W. Zhang, Junchi Yan, X. Wang, H. Zha: Deep extreme multi-label learning, ICMR 2018

13. Z. Ren, Junchi Yan*, B. Ni, B. Liu, H. Zha, X. Yang.. Unsupervised Deep Learning for Optical Flow Estimation. AAAI 2017.

14. R. Wang, Junchi Yan*, X. Yang, Learning Combinatorial Embedding Networks for Deep Graph Matching, ICCV 2019 Oral

Thanks!

Q&A