Machine Learning for Solving Combinatorial Problems: Some Empirical Studies

Junchi Yan

yanjunchi@sjtu.edu.cn

Department of Computer Science and Engineering, SJTU

November 26th, 2022

Acknowledgement of Major Collaborative Students

Runzhong Wang
PhD
Class of 2019

Chang Liu
PhD
Class of 2020

Qibing Ren
MS
Class of 2020

Han Lu
PhD
Class of 2021

Ruoyu Cheng
MS
Class of 2021

https://thinklab.sjtu.edu.cn/

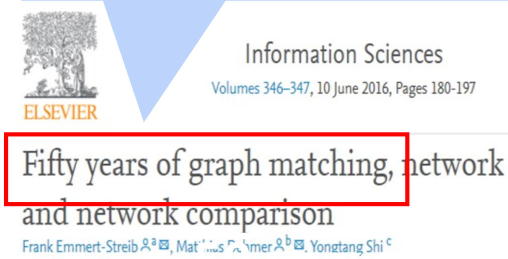
https://github.com/Thinklab-SJTU/

1. Background of Research
2. Recent Work
3. Summary and Outlook

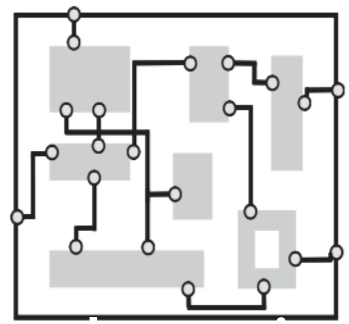
Background of Research

50 years of research on a single graph theory problem

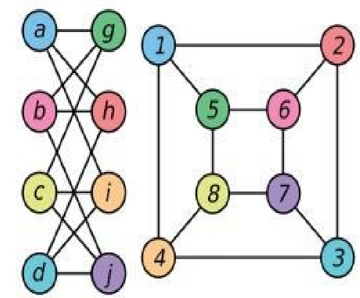
Half-century researches on classical graph matching problem


ELSEVIER
Information Sciences
Volumes 346–347, 10 June 2016, Pages 180-197
Fifty years of graph matching, network and network comparison
Frank Emmert-Streib, Matthias Dehmer, Yongtang Shi

PCB Placement

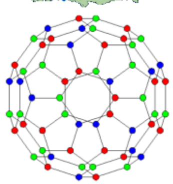


Graph Construction and Optimization

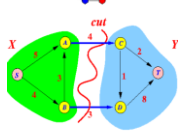


More graph theory problem challenges in real-world scenarios

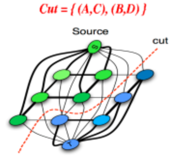
TSP



GCP



MFP



MCP

IBM CPLEX

GUROBI OPTIMIZATION

ILOG Changing the rules of business

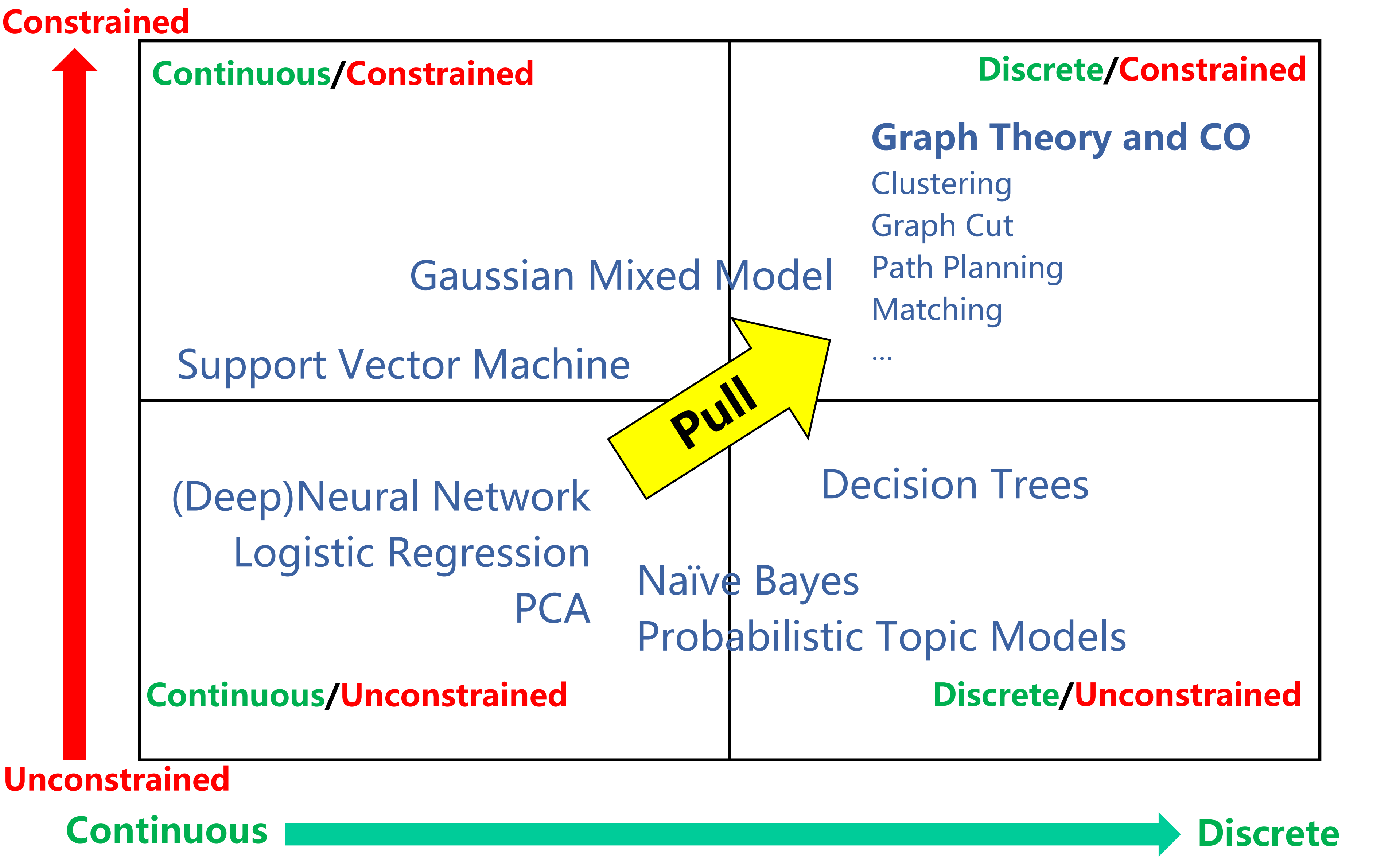
mosek

Robert Bixby
Prof. at Rice

Jerome Chailloux
Asst. Prof. at IP Paris

Erling Andersen
Asst. Prof. at SDU

Background of Research



Background of Research

Perceptual Problems

Matrix/ Sequence

Continuous Unconstrained

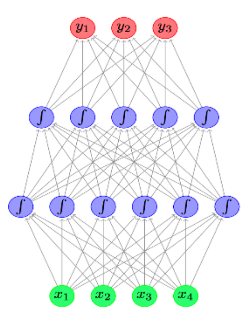
Data Form

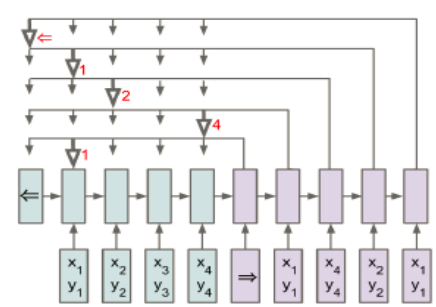
Problem Setting

Graph Theory Problems

Graph Structure

Discrete Constrained





General NN

Methodology Innovation

CO Models

Background of Research

Classical Comfort Zone

- Moderate Size
- Relative Stability
- Classical Problem
- Exact Certainty

Classical Solvers | **Machine Learning**

Open Scenarios

- Large Size — Volume
- Rapid Change — Velocity
- Multiple Forms — Variety
- More Uncertainty — Veracity

- CVPR’22 Best Paper: Learning to Solve Hard Minimal Problems
- EJOR’21 A Survey by Prof. Bengio: Machine Learning for CO
- NSF makes $20 million investment in Optimization-focused AI Research Institute

1. Background of Research
2. Recent Work
3. Summary and Outlook

1) Graph Matching

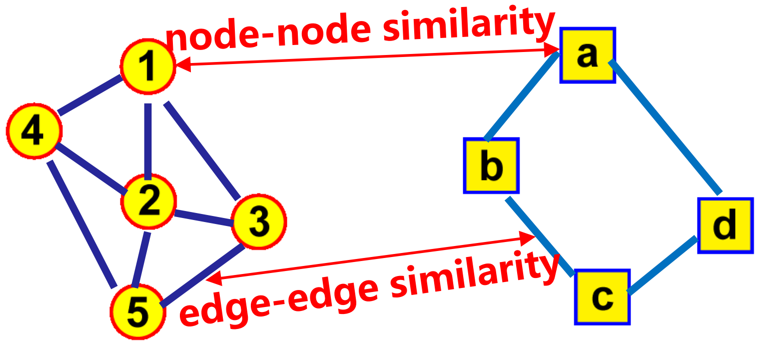
2) Generality

3) Robustness

4) Graph Application

Classical Solution to Graph Matching Problem

- **NP-hard GM Problem:**



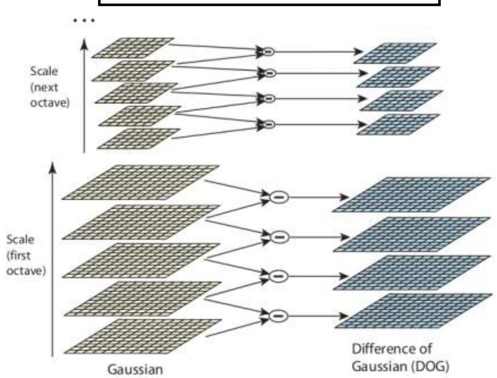
$$\max_{\mathbf{X}} \quad \mathrm{vec}(\mathbf{X})^T \; \mathbf{K} \; \mathrm{vec}(\mathbf{X})$$

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

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

- **Classical GM Pipeline:**

SIFT feature extractor → compute node, edge similarity → $\mathbf{K}$ → classical algorithm approximate solution → $\mathbf{X}$ → matching result

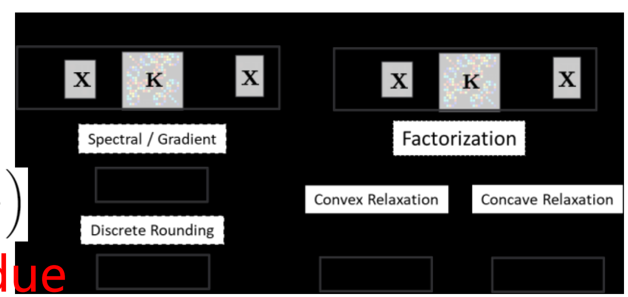


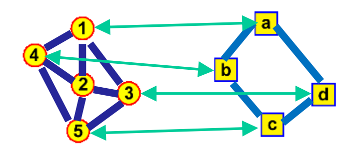
(limited representation capability of SIFT)

Fixed method, e.g. Gaussian Kernel Function

$$\mathbf{K}_{ia,jb} = \exp\left(\frac{(\mathbf{f}_{ij} - \mathbf{f}_{ab})^2}{\sigma^2}\right)$$

(limited capacity due to fixed similarity)





(limited performance of classical algorithm)

Learning GM by Graph Embedding Model ICCV19/TPAMI20

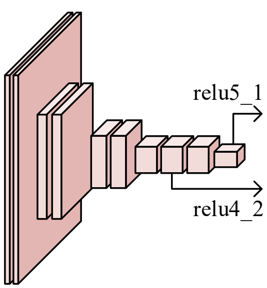
- **Classical GM Pipeline:**

SIFT feature extractor → compute node, edge similarity → classical algorithm approximate solution → matching result

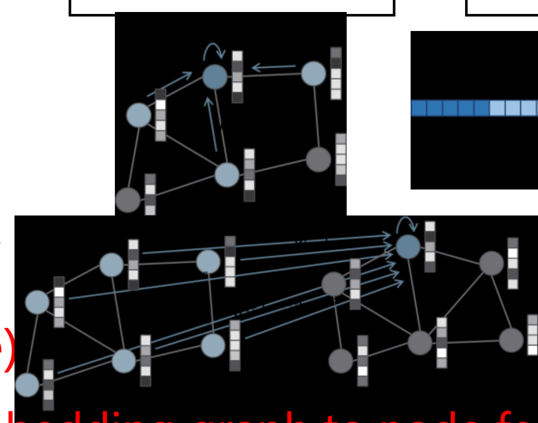
(limited representation capability of SIFT) (limited capacity due to fixed similarity) (limited performance of classical algorithm)

- **Deep Graph Embedding GM Pipeline:**

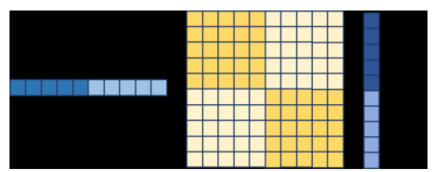
CNN feature extractor → intra-+cross-graph GNN embedding → learn node similarity → Sinkhorn algorithm → matching result



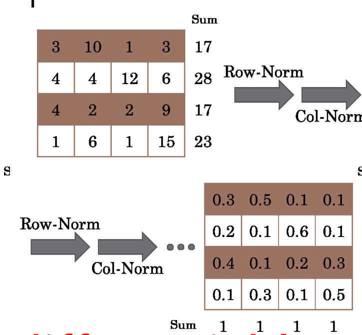
(robust to noise)



(embedding graph to node features, complexity reduced)



(learning weighted similarity)



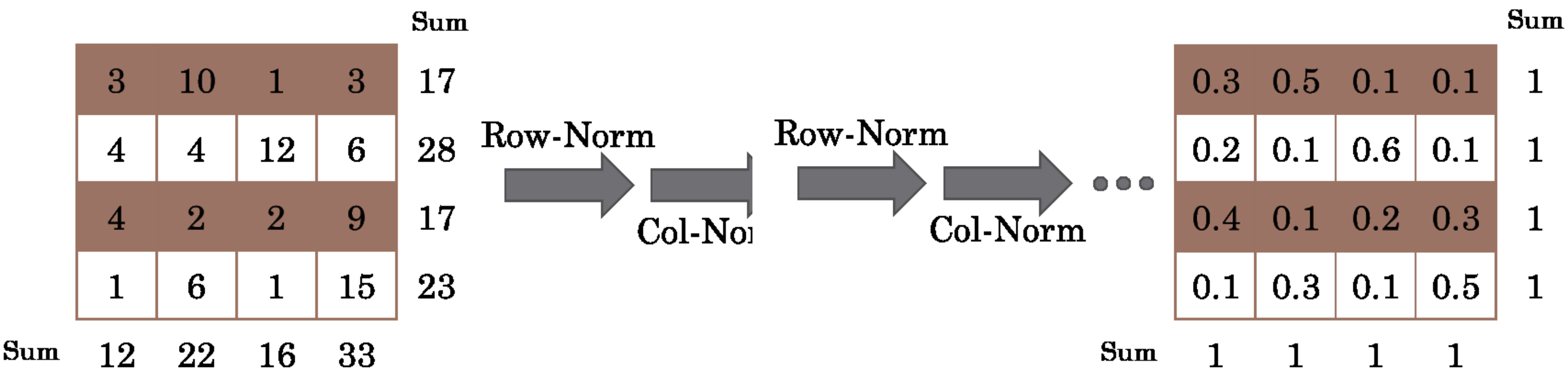
(differentiable exact solution)

GitHub repo QR code

[1] Combinatorial Learning of Robust Deep Graph Matching: an Embedding based Approach, **TPAMI** 2020
[2] Learning Combinatorial Embedding Networks for Deep Graph Matching, **ICCV** 2019

Learning GM by Graph Embedding Model ICCV19/TPAMI20

- **Sinkhorn:** differentiable, exact linear assignment algorithm



- **How to invoke:** pip install pygmtools
  (already support numpy, pytorch, paddle, jittor; will support tensorflow, mindspore)

```
>>> import torch
>>> import pygmtools as pygm
>>> pygm.BACKEND = 'pytorch'
>>> np.random.seed(0) # 2-dimensional (non-batched) input
>>> s_2d = torch.from_numpy(np.random.rand(5, 5))
>>> s_2d tensor([[0.5488, 0.7152, 0.6028, 0.5449, 0.4237],
                 [0.6459, 0.4376, 0.8918, 0.9637, 0.3834],
                 [0.7917, 0.5289, 0.5680, 0.9256, 0.0710],
                 [0.0871, 0.0202, 0.8326, 0.7782, 0.8700],
                 [0.9786, 0.7992, 0.4615, 0.7805, 0.1183]])
```

```
>>> x = pygm.sinkhorn(s_2d)
>>> x tensor([[0.1888, 0.2499, 0.1920, 0.1603, 0.2089],
              [0.1895, 0.1724, 0.2335, 0.2219, 0.1827],
              [0.2371, 0.2043, 0.1827, 0.2311, 0.1447],
              [0.1173, 0.1230, 0.2382, 0.1996, 0.3219],
              [0.2673, 0.2504, 0.1536, 0.1869, 0.1418]])
>>> print('row_sum:', x.sum(1), 'col_sum:', x.sum(0))
row_sum: tensor([1.0000, 1.0000, 1.0000, 1.0000, 1.0000])
col_sum: tensor([1.0000, 1.0000, 1.0000, 1.0000, 1.0000])
```

Learning GM by Graph Embedding Model ICCV19/TPAMI20

- **Permutation Loss:** The matching problem can be considered as a binary classification problem for each element

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

cross-entropy

1.	0	0	0
0	0	1.	0
0	1.	0	0
0	0	0	1.

ground truth S^{gt}

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

- Compared with the regression based offset loss used in the past, the permutation loss better portrays the combinatorial optimization nature of graph matching

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

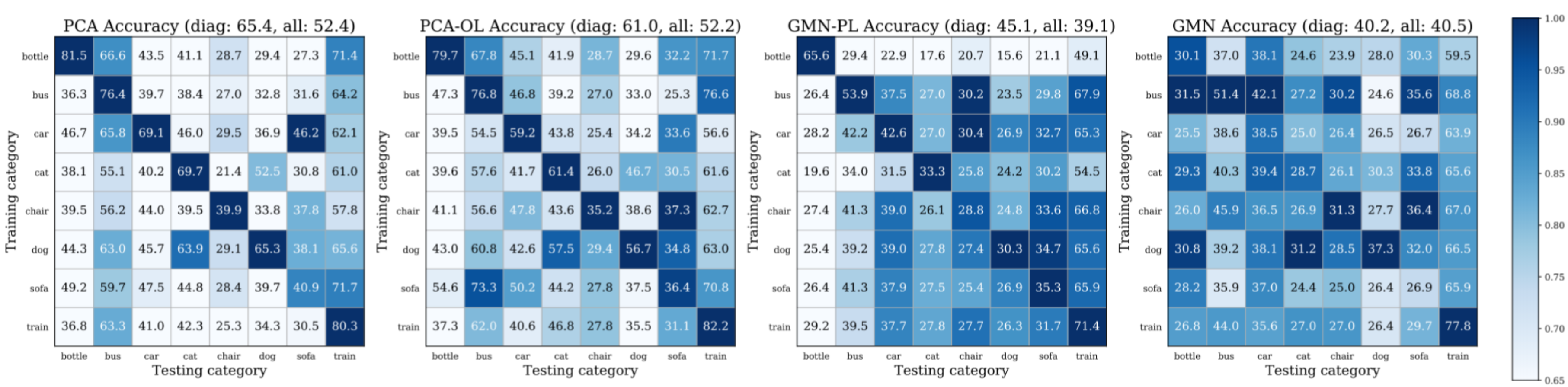
Learning GM by Graph Embedding Model ICCV19/TPAMI20

- **Matching results on PascalVOC:**

Permutation Loss>Offset Loss，Intra-+Cross-graph GNN>Intra-graph GNN>Classical GM

Model	CNN	GM Formulation	Loss Func	Matching Acc
GMN	VGG16	Classical GM（Zanfir et al. CVPR 2018）	Offs Loss	55.3
GMN-PL	VGG16	Classical GM（Zanfir et al. CVPR 2018）	Perm Loss	57.9
PIA-GM-OL	VGG16	Intra-graph GNN	Offs Loss	61.6
PIA-GM	VGG16	Intra-graph GNN	Perm Loss	63.0
PCA-GM	VGG16	**Intra-+Cross-graph GNN**	Perm Loss	**63.8**

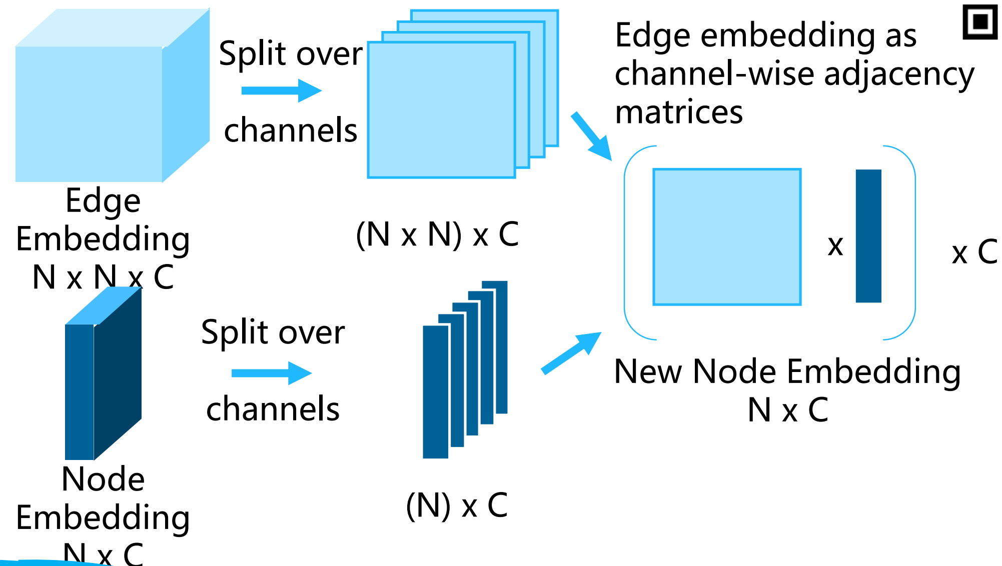
- **The model has the capability to transfer across categories:**



Improvement on Graph Embedding and Loss Function ICLR20

Learning deep graph matching with channel-independent embedding and Hungarian attention, **ICLR** 2020

- **Improve Graph Embedding Module:** Simulate multi-head attention, propose a Channel Independent Embedding (CIE) method

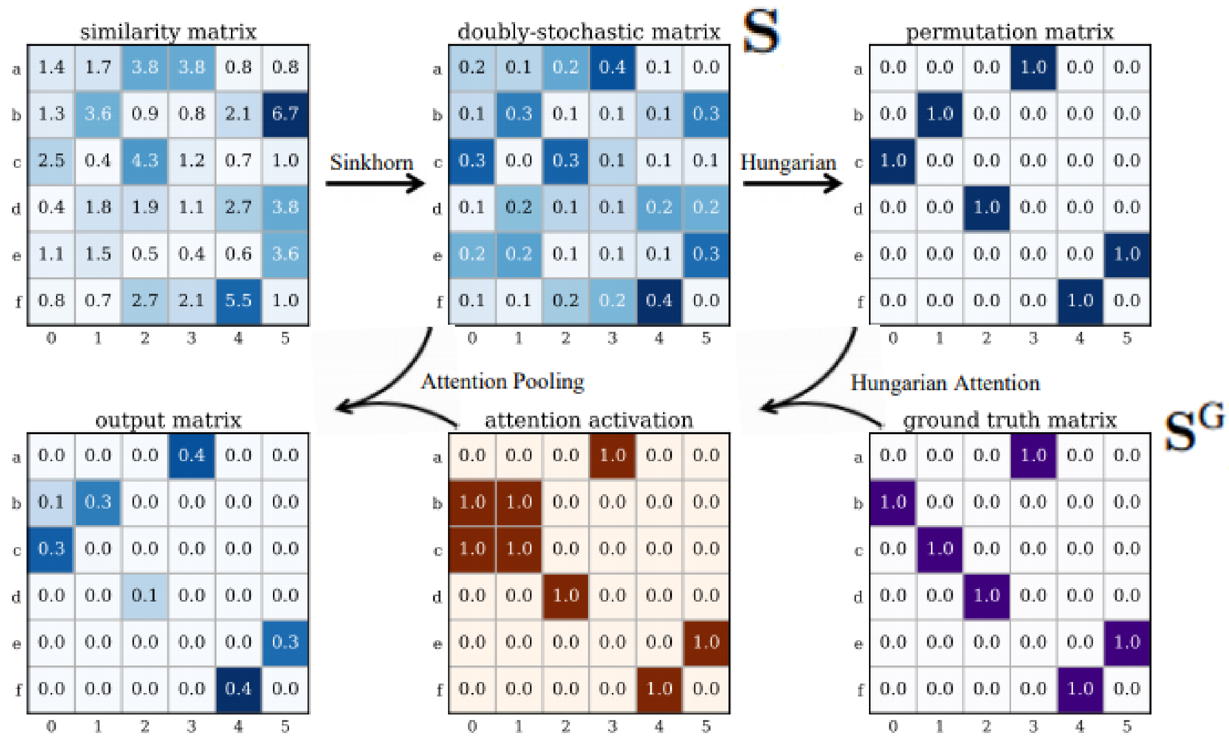


- **Experiment:** Under control variates, **CIE outperforms other GNN structures**

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	Ave
GMN-D	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-P	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
GAT-P	46.4	60.5	60.9	51.8	79.0	70.9	62.7	70.1	39.7	63.9	66.2	63.8	65.8	62.8	39.5	82.0	66.9	50.1	78.5	90.3	63.6
GAT-H	47.2	61.6	63.2	53.3	79.7	70.1	65.3	70.5	38.4	64.7	62.9	65.1	66.2	62.5	41.1	78.8	67.1	61.6	81.4	91.0	64.6
EPN-P	47.6	65.2	62.2	52.7	77.8	69.5	63.4	69.6	37.8	62.8	63.6	63.9	64.6	61.9	39.9	80.5	66.7	45.5	77.6	90.6	63.2
PIA-D	39.7	57.7	58.6	47.2	74.0	74.3	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-P	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-P	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
PCA-H	49.8	60.7	63.9	52.6	79.8	72.5	63.8	71.2	38.4	62.5	71.7	65.4	66.6	62.5	40.5	84.7	66.1	47.9	80.5	91.1	64.6
PCA+-P	46.6	61.0	62.3	53.9	78.2	72.5	64.4	70.5	39.0	63.5	**74.8**	65.2	65.0	61.6	40.8	83.2	67.1	50.5	79.6	91.6	64.6
CIE_2-P	50.9	65.5	68.0	57.0	81.0	75.9	70.3	73.4	**41.1**	66.7	53.2	68.3	68.4	63.5	45.3	84.8	69.7	57.2	79.8	91.6	66.9
CIE_2-H	51.2	68.4	69.5	57.3	82.5	73.5	69.5	74.0	40.3	67.8	60.0	69.7	70.3	65.1	44.7	86.9	70.7	57.3	84.2	92.2	67.4
CIE_1-P	**52.1**	**69.4**	69.9	**58.9**	80.6	76.3	**71.0**	**74.2**	**41.1**	68.0	60.4	69.7	70.7	65.1	**46.1**	85.1	**70.4**	61.6	80.7	91.7	68.1
CIE_1-H	51.2	69.2	**70.1**	55.0	**82.8**	72.8	69.0	**74.2**	39.6	**68.8**	71.8	**70.0**	**71.8**	**66.8**	44.8	**85.2**	69.9	**65.4**	85.2	**92.4**	**68.9**

Improvement on Graph Embedding and Loss Function ICLR20

- **Improve Loss Function:** Permutation loss requires the output to be 0/1, which may cause overfitting
- Propose **Hungarian Attention**, focusing on inconsistent matches after Hungarian
- **Experimental results:** mitigating overfitting and improving test set performance



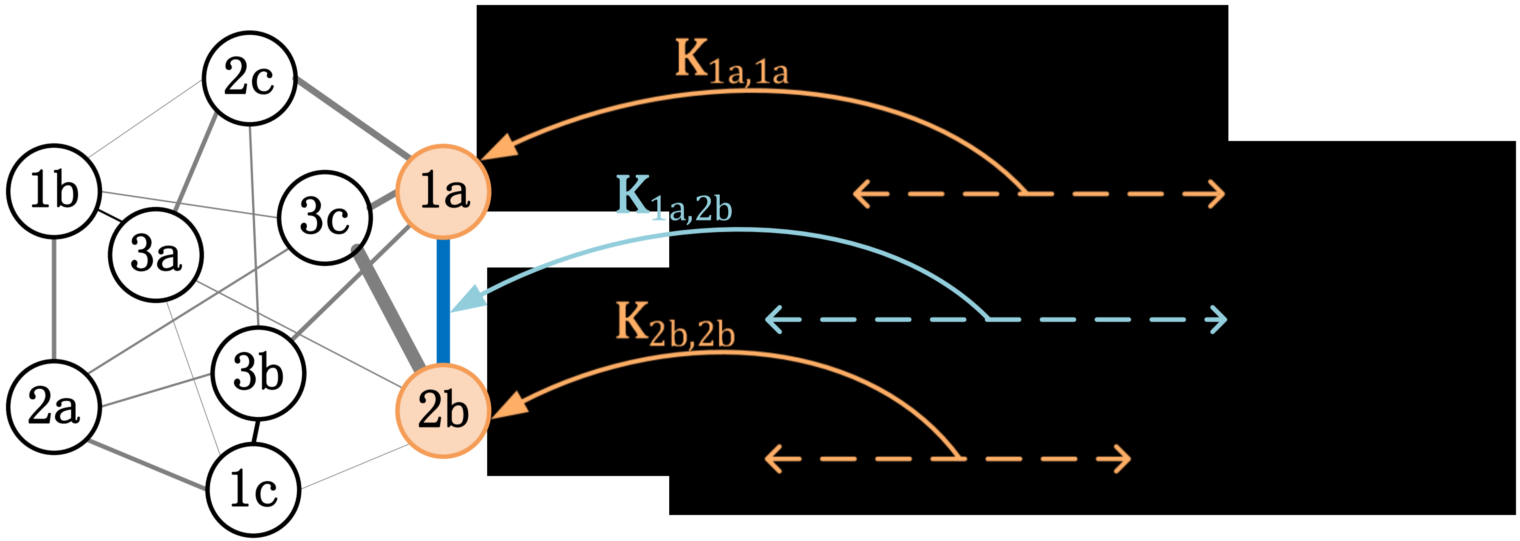
$$\mathcal{H}_{\mathrm{CE}} = -\sum_{i \in \mathcal{G}_1, j \in \mathcal{G}_2} \mathbf{Z}_{ij} \left(\mathbf{S}^{\mathrm{G}}_{ij} \log \mathbf{S}_{ij} + \left(1 - \mathbf{S}^{\mathrm{G}}_{ij}\right) \log \left(1 - \mathbf{S}_{ij}\right) \right)$$

$$\mathbf{Z} = \mathrm{Atten}\left(\mathrm{Hungarian}(\mathbf{S}), \mathbf{S}^{\mathrm{G}}\right)$$

	Training Acc	Test Acc
Perm. Loss	**88.2**	63.8
Hung. Attn	78.7	**68.9**

Learning GM Solvers TPAMI22

- **GM is equivalent to node classification on an association graph:**



Association Graph Graph Matching Problem

- **Node 1 matches node a ⟷ 1a=1 on association graph**

- **Therefore, GM solvers==node classifier on association graph**

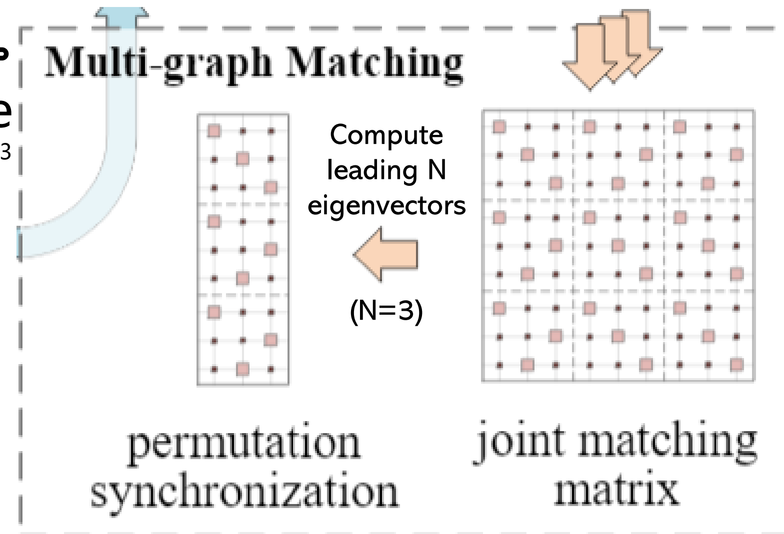
- **Naturally, GNN that excel in node classification can serve as graph matching solvers!**

Learning GM Solvers TPAMI22

Extend to Multi-graph Matching

Adopt permutation synchronization technique

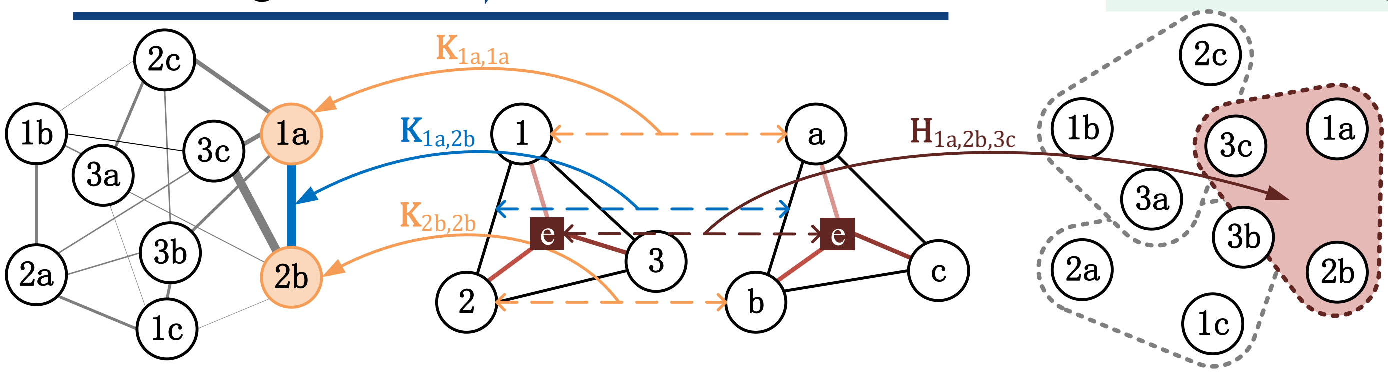
Pachauriy et al., Solving the multi-way matching problem by permutation synchronization, in NIPS 2013



MGM Test Dataset	Matching Acc
NGMv2 (2GM)	**97.5**
NHGMv2 (HGM)	97.8
NMGMv2 (MGM)	98.2

Extend to Hyper Graph Matching

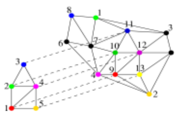
Neural Graph Matching (NGM) GM	Neural **Hyper** Graph Matching (NHGM) **H**GM
1/2 order features	**high-order** features
association graph	association **hyper**graph
update features along edges	update features along **hyper**edges
matching acc 80.1	matching acc **80.7**



(a) association graph (b) graph matching (c) association hypergraph

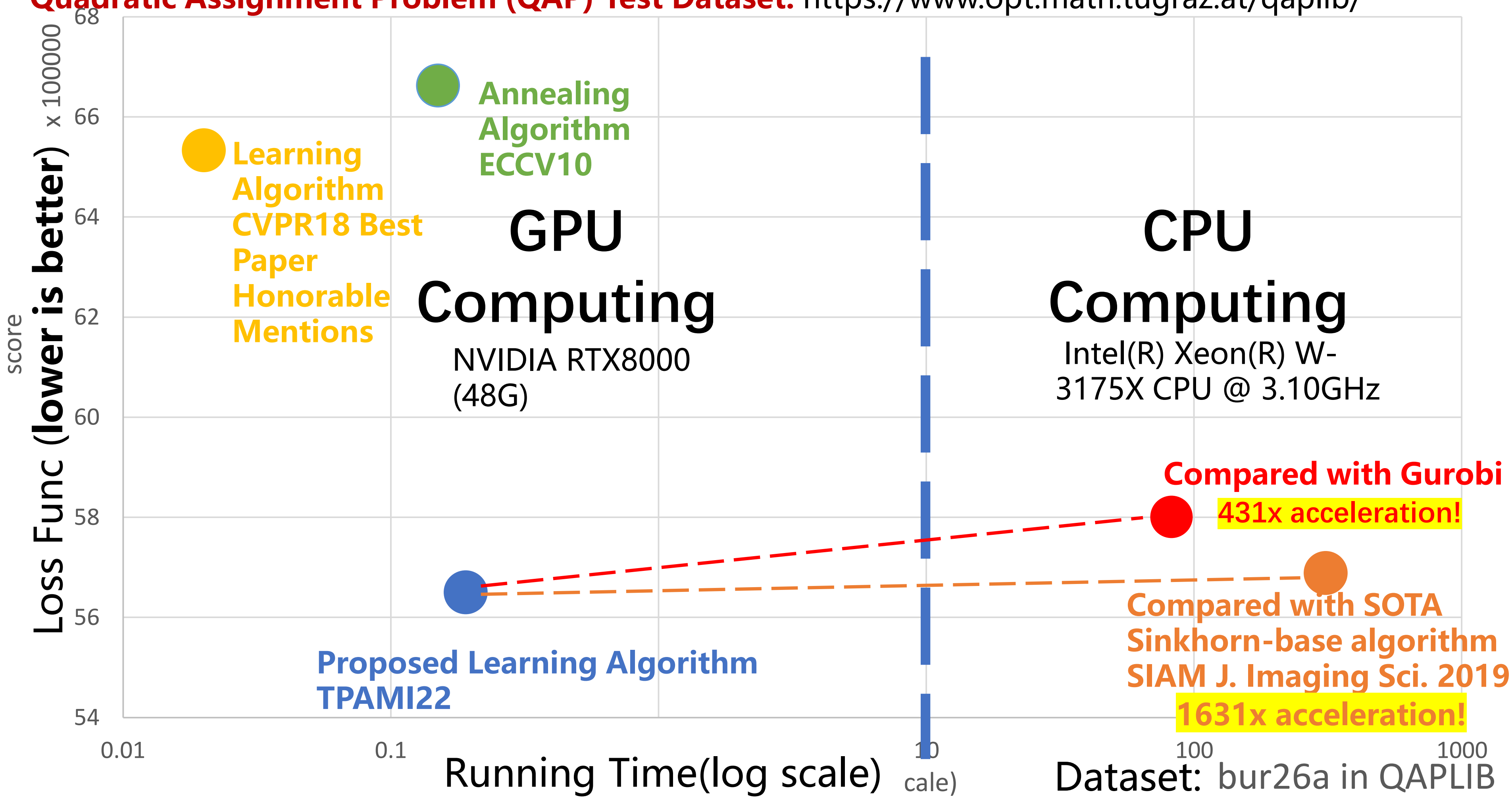
Learning GM Solvers TPAMI21

Neural Graph Matching Network: Learning Lawler’ s Quadratic Assignment Problem with Extensions to Hypergraph and Multi-graph matching, **TPAMI** 2021



QAP $\max_{x} \; x K x^T$ → Assoc Graph Construction → GNN → Node Classifier → Double Stochastic Iteration → Solu-tion

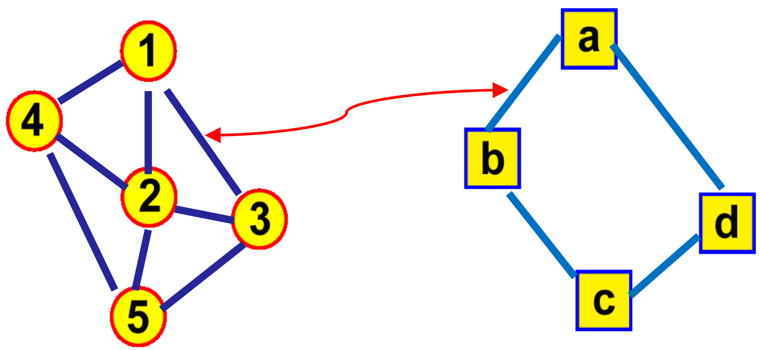
Quadratic Assignment Problem (QAP) Test Dataset: https://www.opt.math.tugraz.at/qaplib/



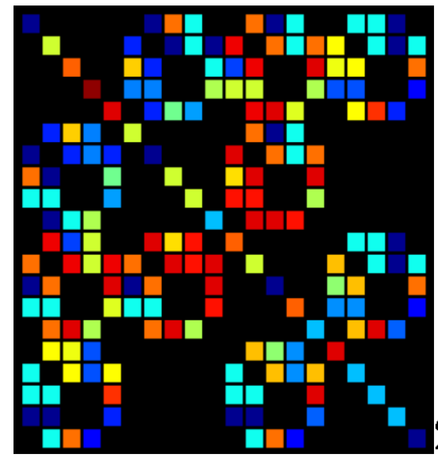
Dataset: bur26a in QAPLIB

Self-Supervised Learning for GM ECCV22

Self-supervised Learning of Visual Graph Matching, **ECCV** 2022



$\mathbf{K}$ *(Leordeanu & Hebert, 2005)*



20×20

$\Uparrow$

$\boxed{5 \cdot 4}$

node-node **similarity**

+

edge-edge **similarity**

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

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

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

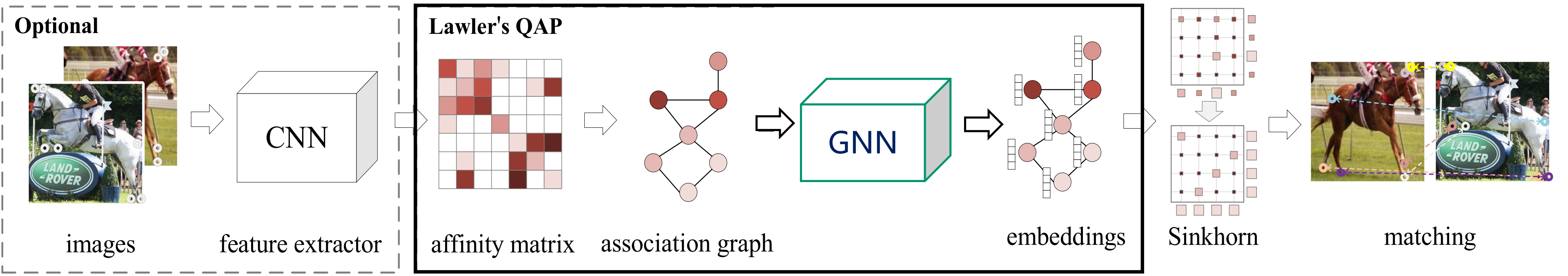
NP-hard

Generally referred to as
Lawler’s Quadratic Assignment Problem
(**Lawler’s QAP**)

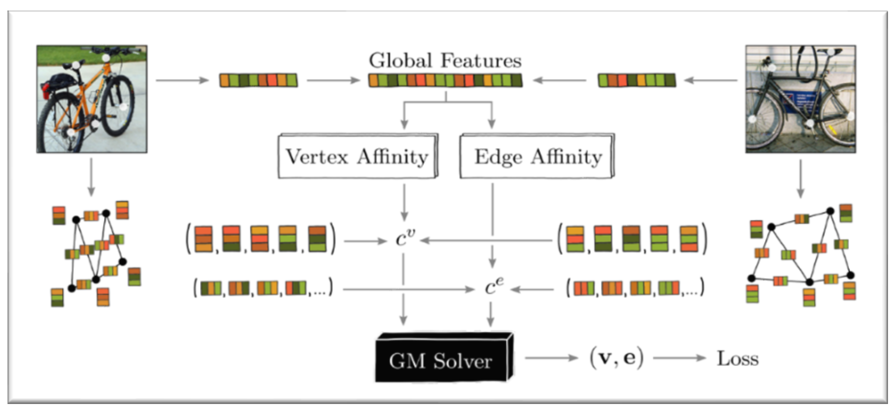
GitHub repo
QR code

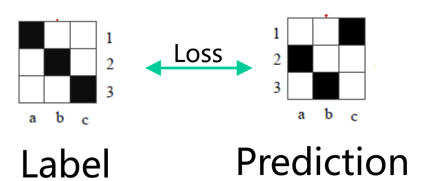
Code available at “https://github.com/Thinklab-SJTU/ThinkMatch-SCGM”

Self-Supervised Learning for GM ECCV22



Existing Deep GM Models:
Require ground truth node correspondence as labels for supervised learning

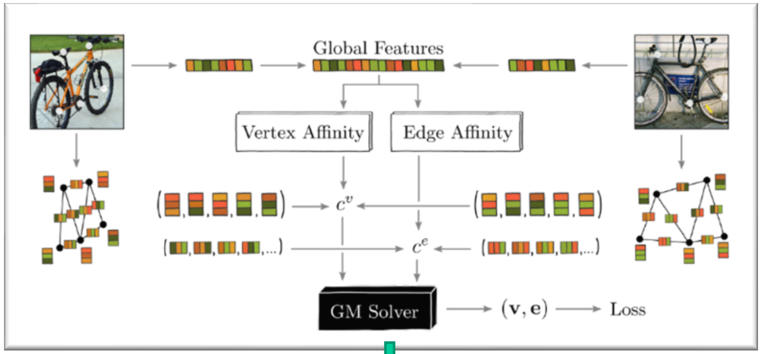


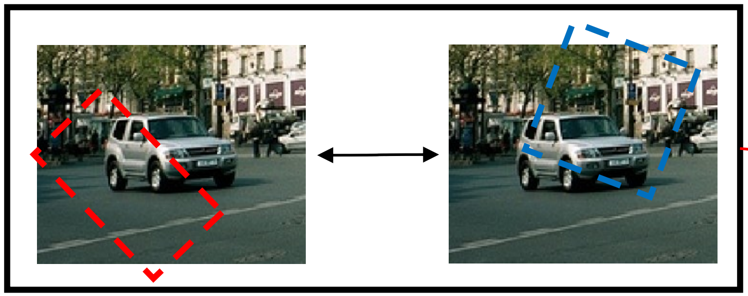


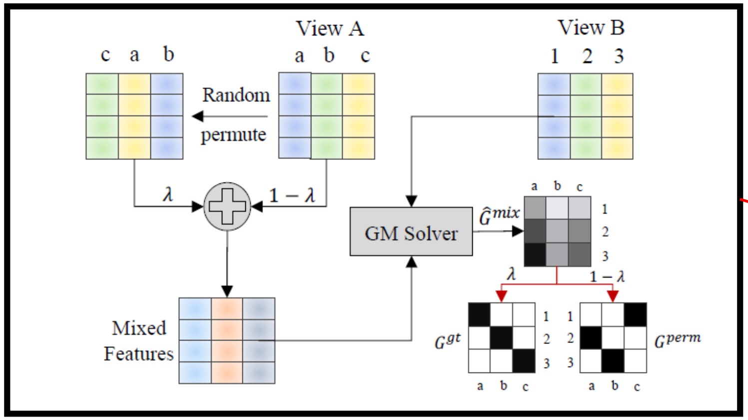
- Rolínek, Michal, et al. "Deep graph matching via blackbox differentiation of combinatorial solvers." ECCV. Springer, Cham, 2020.

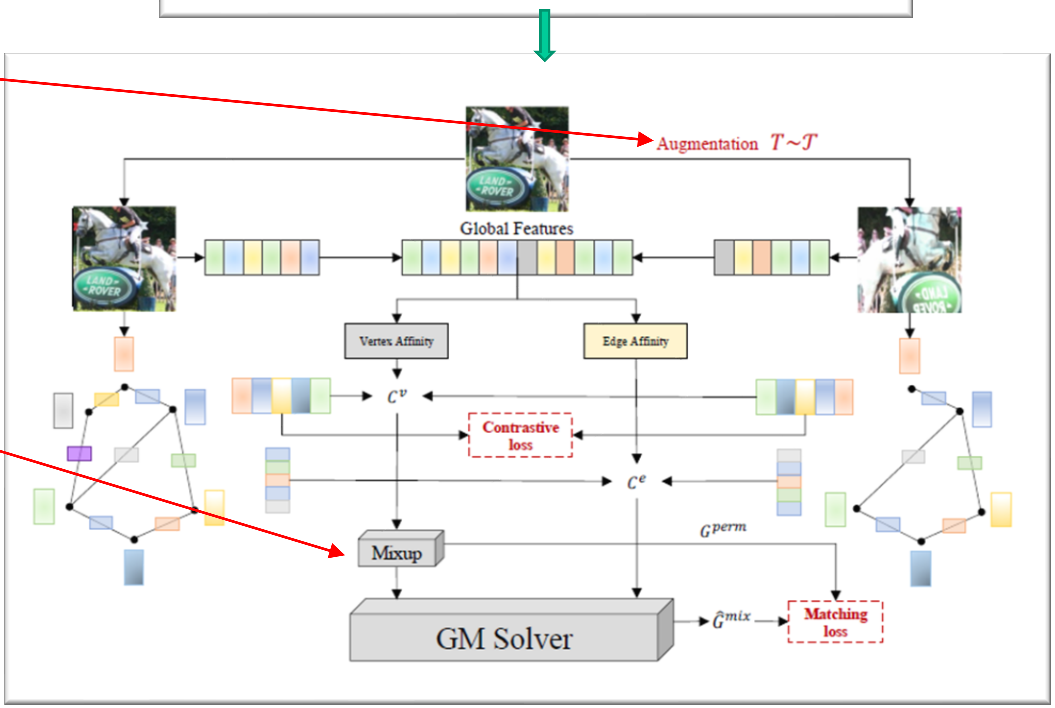
Self-Supervised Learning for GM ECCV22

- Self-supervised Learning for Graph Matching (**SCGM**)
  - Two-stage Data Augmentation
  - Contrastive Learning on Node Layers





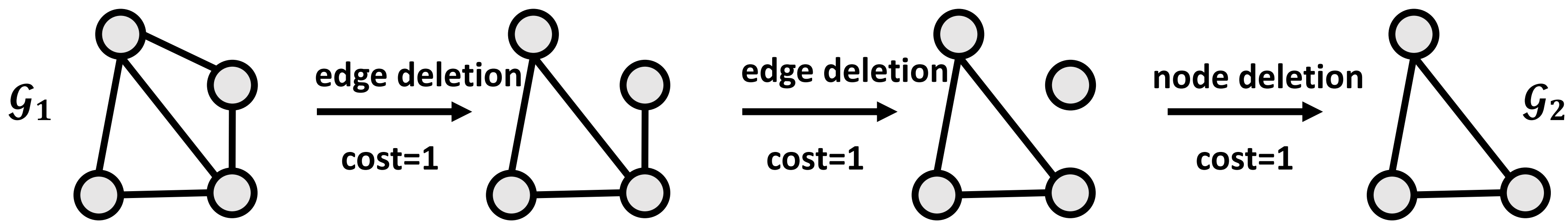




Solving Graph Edit Distance and Edit Path CVPR21

Combinatorial Learning of Graph Edit Distance via Dynamic Embedding, **CVPR** 2021

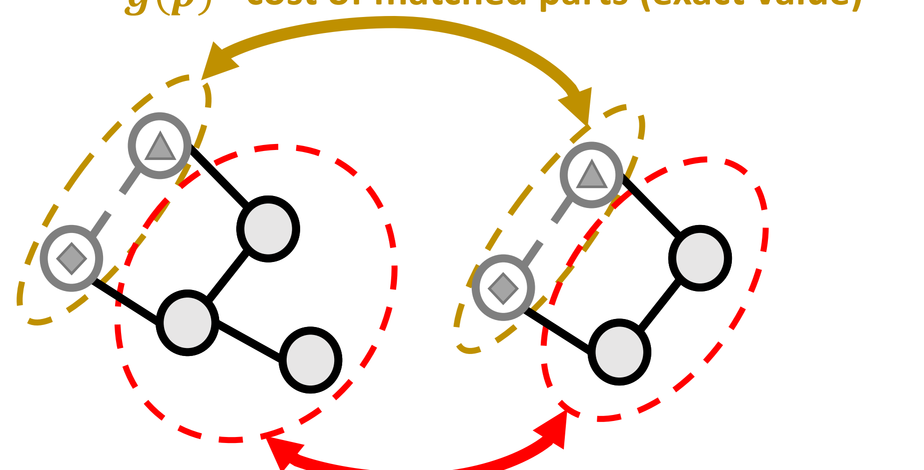
Graph Edit Distance(GED):



GED$(\mathcal{G}_1, \mathcal{G}_2)$ = 3

Classical A* Algorithm——$g(p)$ and $h(p)$:

$g(p)$ cost of matched parts (exact value)

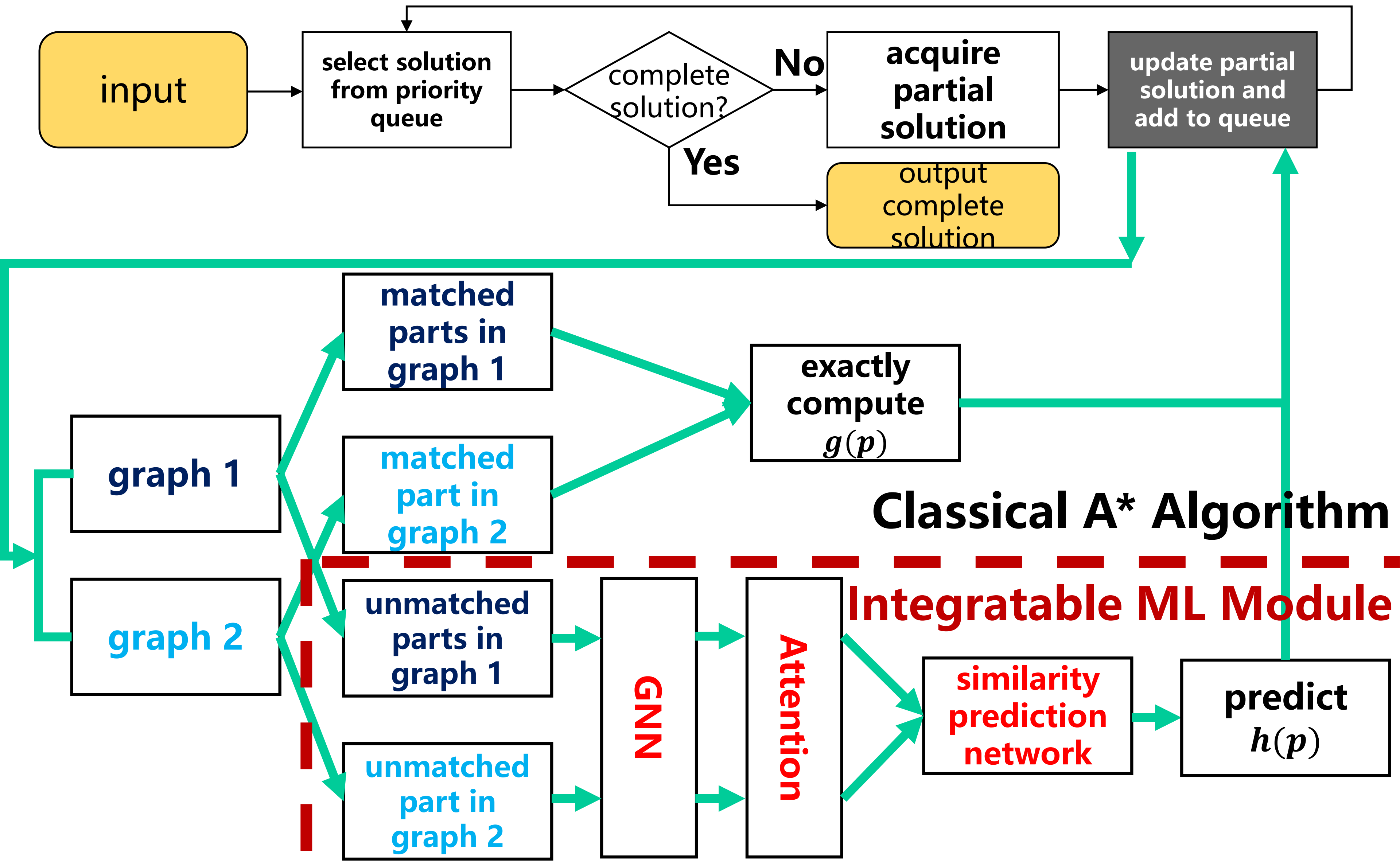


$h(p)$ cost of unmatched parts (predicted value)

GitHub repo QR code

Solving Graph Edit Distance and Edit Path CVPR21

Combinatorial Learning of Graph Edit Distance via Dynamic Embedding, **CVPR** 2021



Solving Graph Edit Distance and Edit Path CVPR21

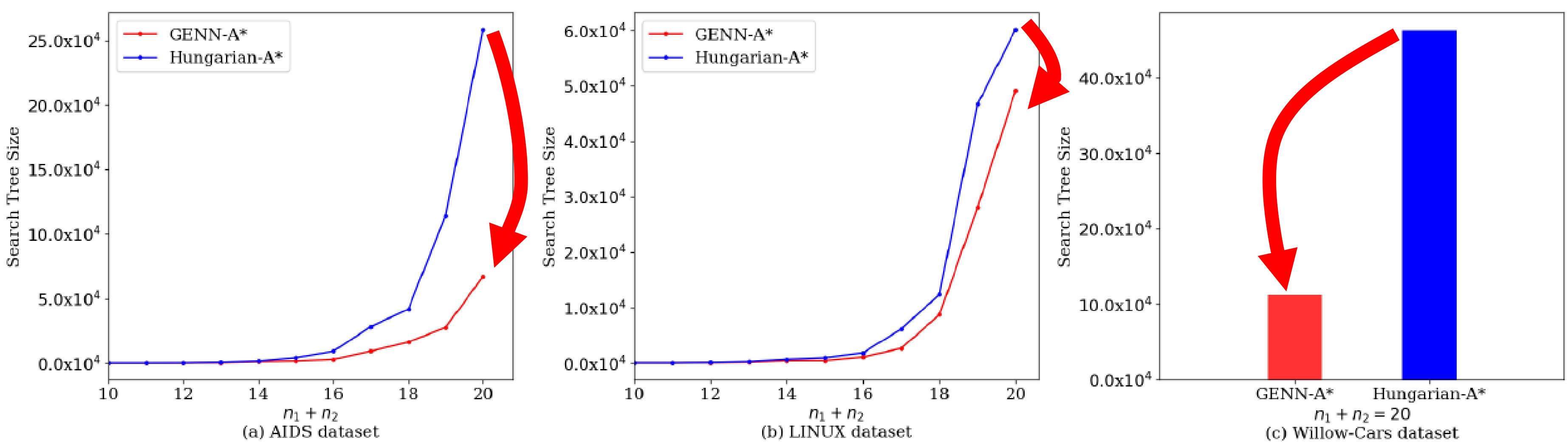
Combinatorial Learning of Graph Edit Distance via Dynamic Embedding, **CVPR** 2021

Accuracy Metrics for GED on 3 Real-world Datasets

Method	Edit Path	AIDS			LINUX			Willow-Cars		
		mse ($\times 10^{-3}$)	ρ	p@10	mse ($\times 10^{-3}$)	ρ	p@10	mse ($\times 10^{-3}$)	ρ	p@10
SimGNN [3]	×	1.189	0.843	0.421	1.509	0.939	0.942	-	-	-
GMN [26]	×	1.886	0.751	0.401	1.027	0.933	0.833	-	-	-
GraphSim [4]	×	**0.787**	0.874	0.534	**0.058**	**0.981**	**0.992**	-	-	-
GENN (ours)	×	1.618	**0.901**	**0.880**	0.438	0.955	0.527	-	-	-
Beam Search [20]	✓	12.090	0.609	0.481	9.268	0.827	**0.973**	1.820	0.815	0.725
Hungarian [31]	✓	25.296	0.510	0.360	29.805	0.638	0.913	29.936	0.553	0.650
VJ [13]	✓	29.157	0.517	0.310	63.863	0.581	0.287	45.781	0.438	0.512
GENN-A* (ours)	✓	**0.635**	**0.959**	**0.871**	**0.324**	**0.991**	0.962	**0.599**	**0.928**	**0.938**

The integratable algorithm preserves the high accuracy of classical solvers

Achieve high efficiency with machine learning algorithm



Classical Algorithm vs Integratable Algorithm: Size of Search Trees

Solving Combinatorial Optimization over Graphs by a General Bi-level ML Framework NeurIPS21

A Bi-level Framework for Learning to Solve Combinatorial Optimization over Graphs, **NeurIPS** 2021

For CO problems over graphs, current formulation is

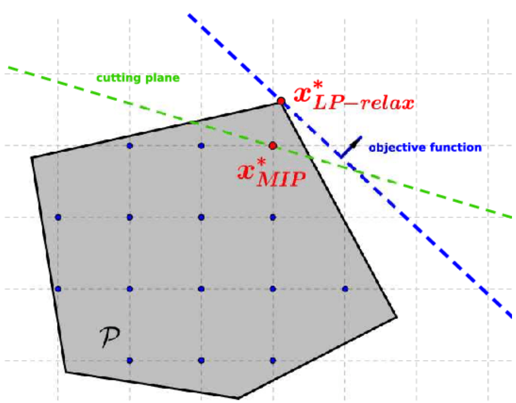
$$\min_{\boxed{\mathbf{x}}} \boxed{f(\mathbf{x}|\mathcal{G})} \qquad s.t. \quad \boxed{h_i(\mathbf{x},\mathcal{G}) \leq 0, \text{for } i = 1...I}$$

Decision Variable | **Objective Function** | **Constraints**

Existing papers use reinforcement learning modeling : **Action Sequence** | **Reward** | **Action Feasible Domain**

However
- Larger scale, longer **action sequence** → **Sparse reward**, hard to converge
- Assume adequate model capacity to learn $\mathcal{G} \rightarrow \mathbf{x}$ → NP-hard problem, **hard to devise model**

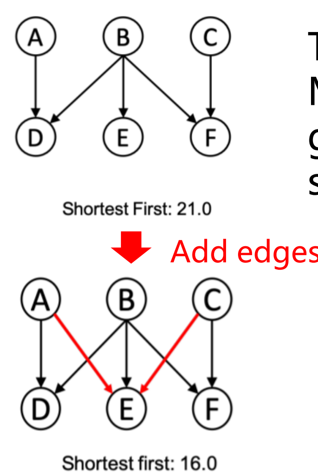
Resort to the classic idea: Modifying the original problem to aid problem solving



Adding cutting planes for integer programming

Node id	Resource	Runtime
A	0.5	6.0
B	0.5	5.0
C	0.5	6.0
D	0.5	5.0
E	0.5	5.0
F	0.5	5.0

Total Resource Limit: 1.0



This paper: Modifying graph structure

GitHub repo QR code

Solving Combinatorial Optimization over Graphs by a General Bi-level ML Framework NeurIPS21

Propose a Bi-level Optimization Formulation:

Upper-level: Adopt a reinforcement learning agent to adaptively modify the graphs

$\mathcal{G}'$

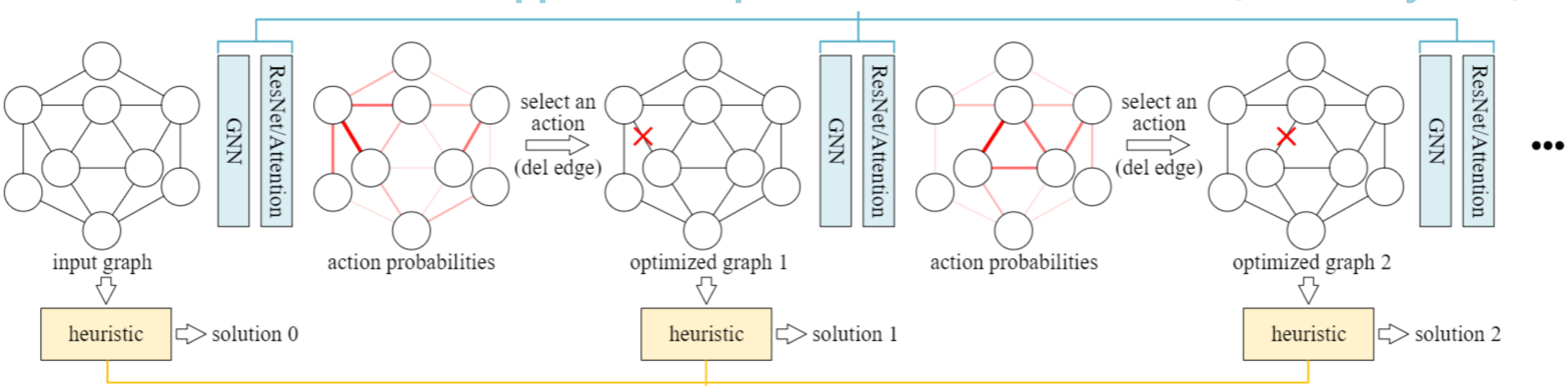
$$\min_{\mathbf{x}',\mathcal{G}'} f(\mathbf{x}'|\mathcal{G}) \qquad s.t. \quad H_j(\mathcal{G}',\mathcal{G}) \le 0, \text{for } j = 1...J$$

$$\mathbf{x}' \in \arg\min_{\mathbf{x}'} \{ f(\mathbf{x}'|\mathcal{G}') : h_i(\mathbf{x}',\mathcal{G}') \le 0, \text{for } i = 1... \mathbf{X}'$$

Lower-level: Optimize decision variables by heuristics

Bi-level Framework: When the upper-level RL modifies graph structure, the lower-level heuristic is invoked

Upper-level Optimizer: RL action network(trained by PPO)

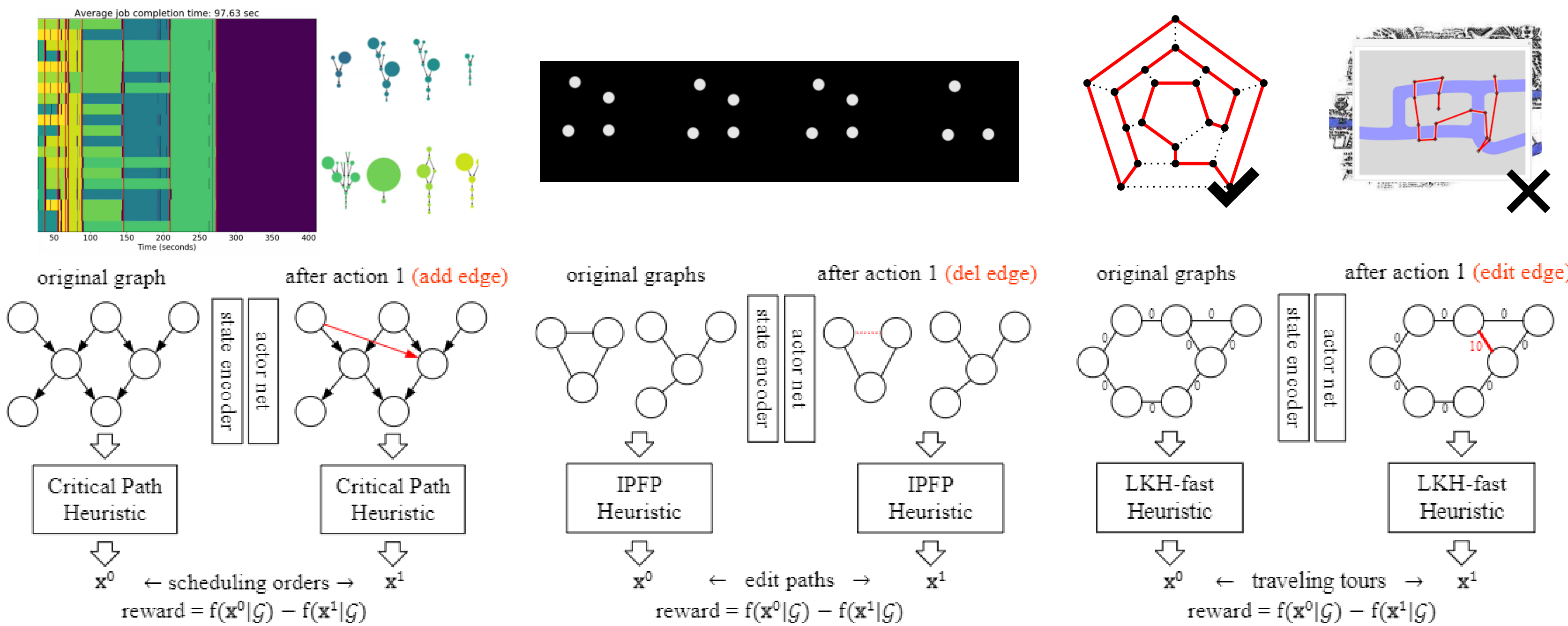


Lower-level Optimizer: Heuristic algorithms

Solving Combinatorial Optimization over Graphs by a General Bi-level ML Framework NeurIPS21

A General Framework for Different Graph Theory Problems

(a) DAG Scheduling (b) GED Problem (c) Hamiltonian Cycle



DAG Sche Time TPC-H Dataset	Custo-mized	Gen-eral	Improv-ements
50 DAGs	9821	8906	9.3%
100 DAGs	16914	15193	10.2%
150 DAGs	24429	22371	8.4%

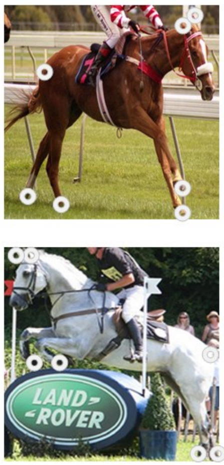
GED AIDS Dataset	Custo-mized	Gen-eral	Improv-ements
20-30 nodes	37.4	29.1	22.2%
30-50 nodes	70.4	61.1	13.2%
50+ nodes	101.9	77	24.4%

Hamiltonian Cycle Accuracy FHCP Dataset	Custo-mized	Gen-eral	Improv-ements
500-600 nodes	20	25	25%

Appearance and Structure Aware Robust Deep Visual Graph Matching CVPR22

Appearance and structure aware robust deep visual graph matching: Attack, defense and beyond, **CVPR** 2022

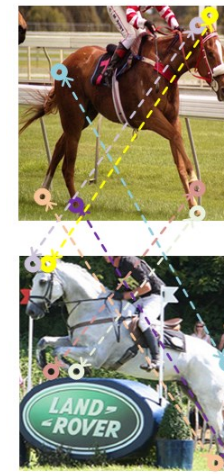
- Research Problem: Robust Decision for Deep Visual GM in Adversarial Attack Contexts
- Deep Visual GM Pipeline (Wang, TPAMI 2021, Rolinek, ECCV 2020):



Multi-graph with keypoints

→ keypoints feature extractor → multi-graph construction → GM solver →

VGG16 with SplineConv | Delaunay Triangulation | GNN or Blackbox Solvers



Similarity between multi-graphs

- Challenge 1: Existing adversarial attack algorithms for graph structures are not feasible for MGM
  - Adding or deleting nodes will degrade matching accuracy
  - Adding or deleting edges will be reverted in multi-graph construction
- Challenge 2: Existing adversarial defense algorithms on a single graph are not feasible for MGM
  - Learn discriminative features between nodes on a single graph
  - Learn correspondences between multiple graphs for MGM

GitHub repo QR code

Github Code: https://github.com/Thinklab-SJTU/robustMatch

Appearance and Structure Aware Robust Deep Visual Graph Matching CVPR22

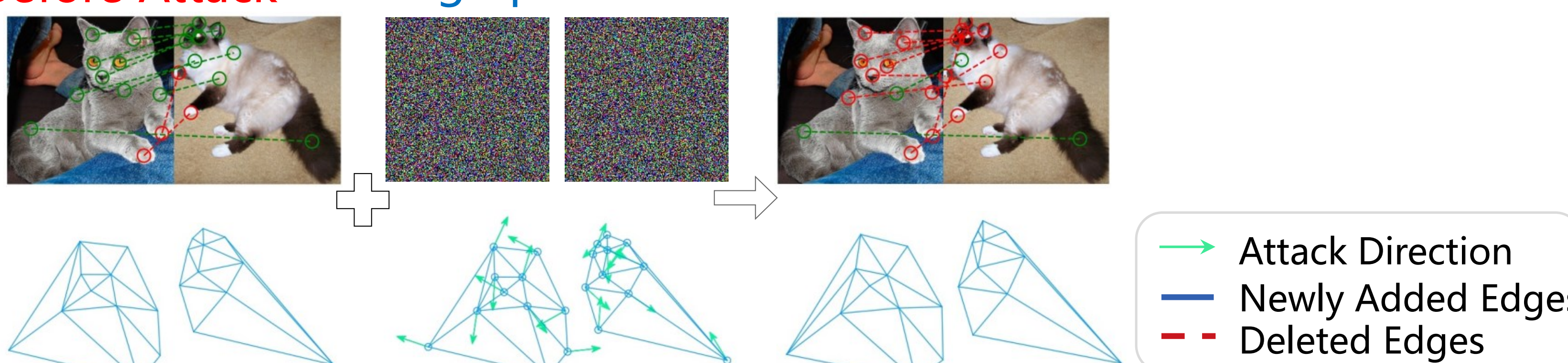
- Attack Strategy: Locality attack by perturbing keypoint localities and pixel attack by perturbing image pixel values
  - Bi-level Constrained Optimization Problem:
    - c, z refers to keypoint localities, features, respectively
    - ϵ_c, ϵ_z refers to perturbation budget, unavailable to attack

$$\max_{c',z'} \max_{G'} \mathrm{L}(\mathrm{f}(c',z',G'),y)$$

$$s.t.\ \mathrm{d}_\infty(c',c) \le \epsilon_c, \mathrm{d}_\infty(z',z) \le \epsilon_z$$

  - Impact of Keypoint Locality Attack on Models:
    - Influence the extraction of keypoint features in the graph
    - Determine the connectivity between keypoints (edge addition or deletion)

Before Attack | Multi-graph Pixel Attack | After Attack



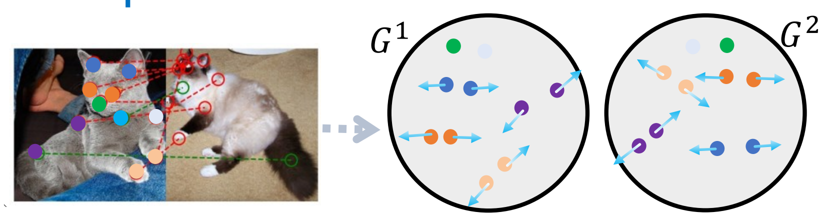
Matched: 9/11 | Multi-graph Locality Attack | Matched: 2/11

Appearance and Structure Aware Robust Deep Visual Graph Matching CVPR22

- Defense Strategy: Vulnerability of appearance-similar keypoints in embedding space and explicit constraints
  - Appearance-similar keypoints are vulnerable to attack
    - ☐ Similar shape, similar texture, symmetrical structure

After attack

  - Actively attack to discover appearance-similar keypoints during training and expand their distances in embedding space

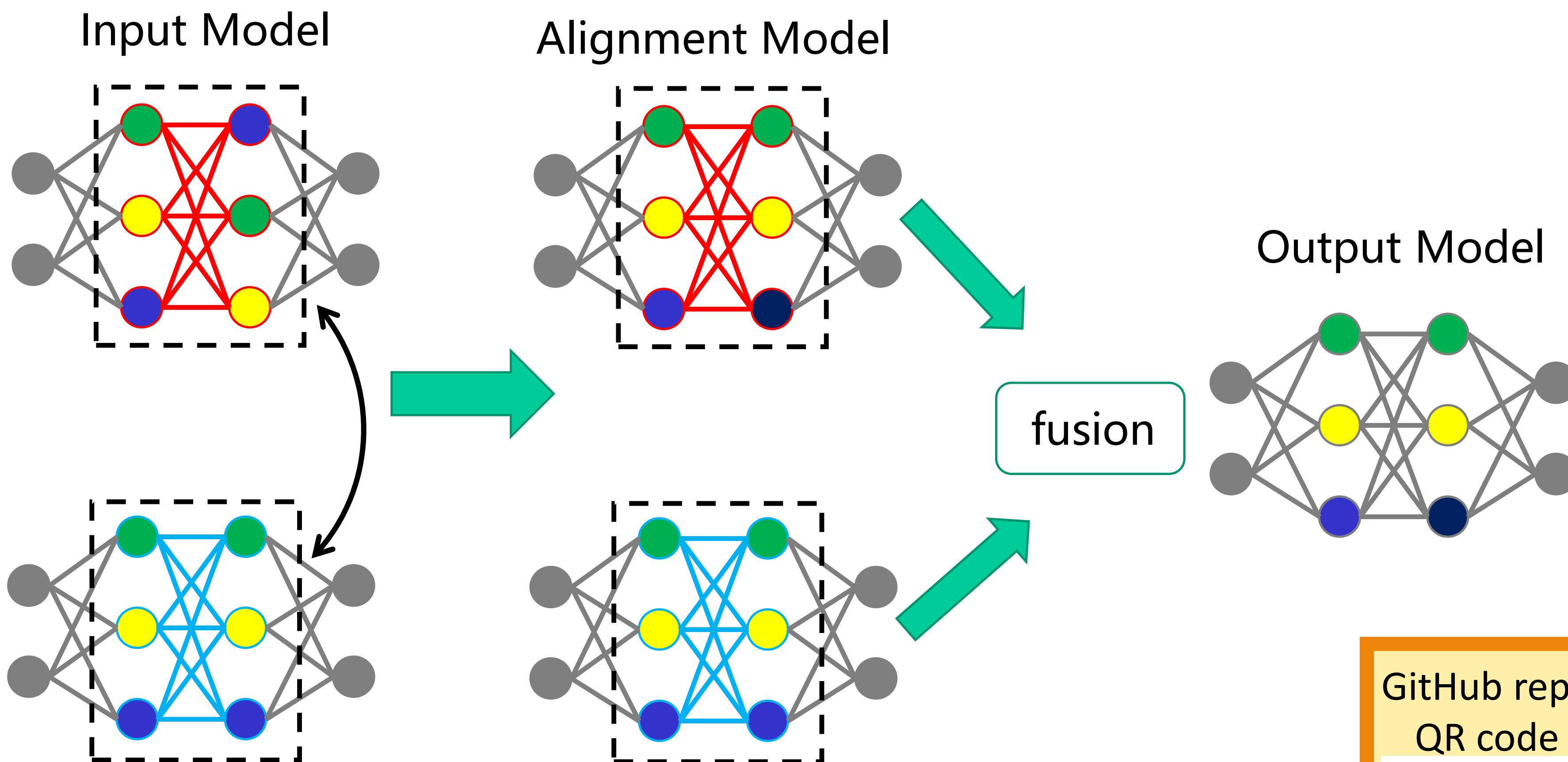


  - Combined with adversarial training, the adversarial samples generated by the attack are received as input to further improve the robustness

adversarial samples generation → deep GM module → predicted matching matrix / ground truth → Appearance Aware Regularizer / Cross Entropy Loss

Graph Matching Based Model Fusion ICML22

Deep Neural Network Fusion via Graph Matching with Applications to Model Ensemble and Federated Learning, **ICML**22



Code available at “https://github.com/Thinklab-SJTU/GAMF”

GitHub repo
QR code

Graph Matching Based Model Fusion ICML22

1. Model Ensemble

   – Prediction-based Model Ensemble: Need to maintain all individual models

   – Fusion-based Model Ensemble: Need to maintain only one model

2. Federated Learning

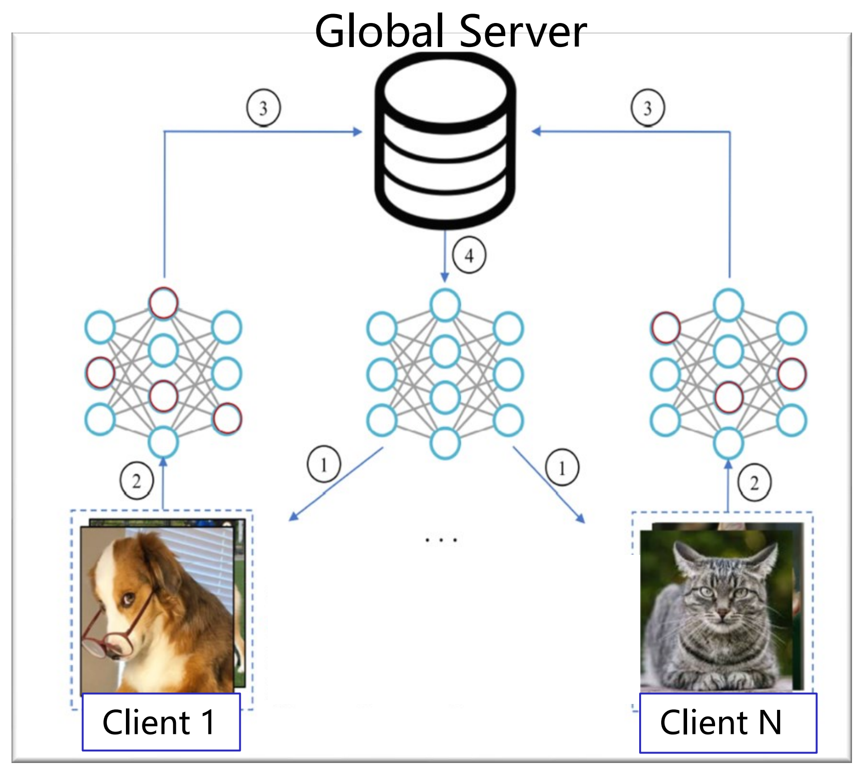
   – FL Pipeline：

     1) Global server sends the global model to each local client

     2) Each client train the local model with their own datasets

     3) Local clients send the local model back to global server

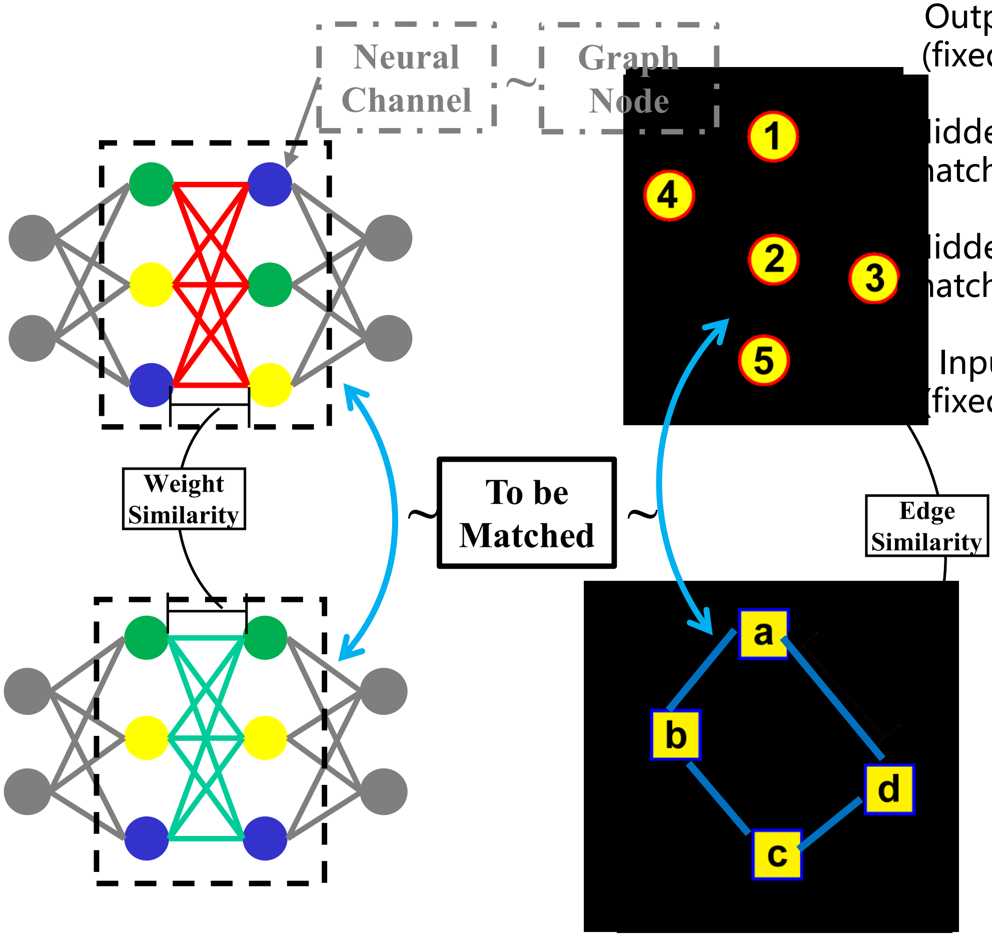
     4) **Global server gathers all local models and merge them into a shared global model**

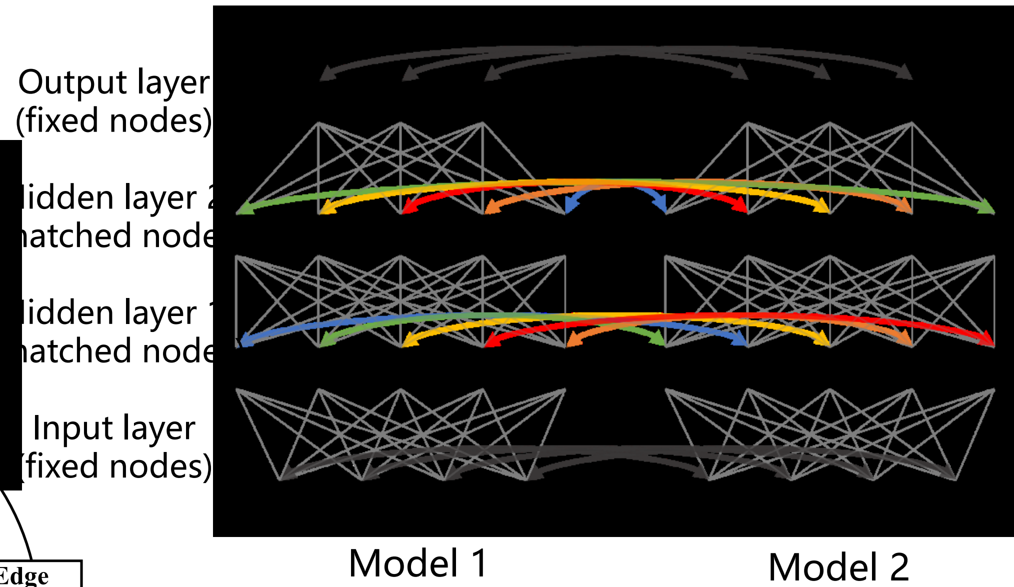


   – Efficiently aggregate local models by Model Fusion

- Li, Q., He, B., and Song, D. Model-contrastive federated learning. CVPR, 2021.

Graph Matching Based Model Fusion ICML22

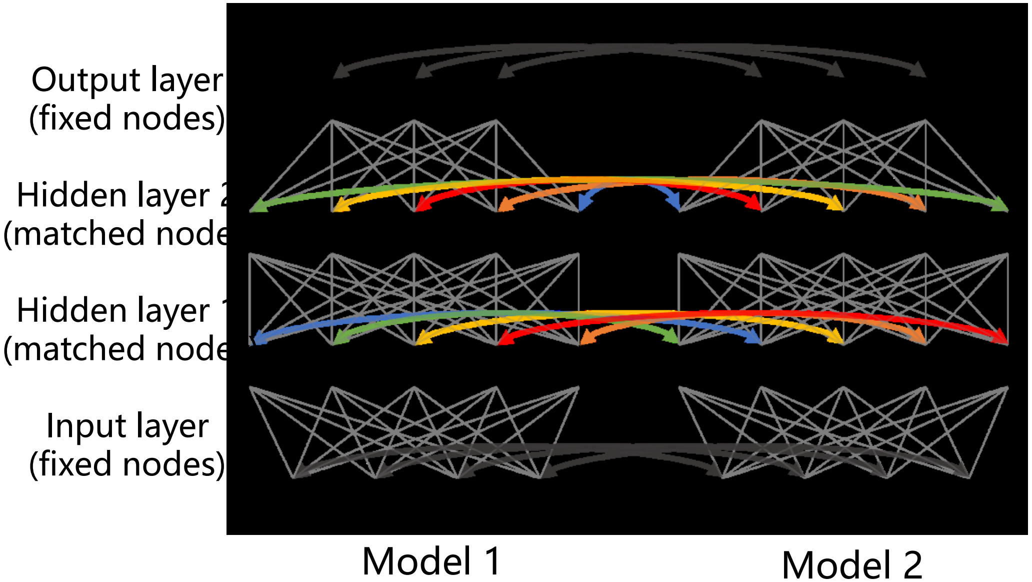


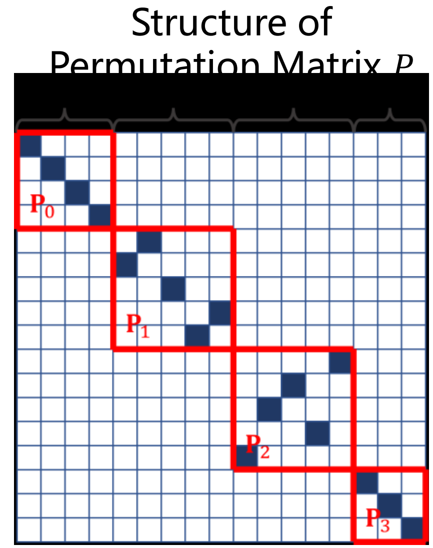


Challenge: Problem Scale

- Model Fusion: Large scale of common NN, with up to 1024 channels each layer and a total number of channels exceeding 10000
- Graph Matching：Less than 100 keypoints in a graph in commonly used dataset, which differs significantly from the requirements of Model Fusion

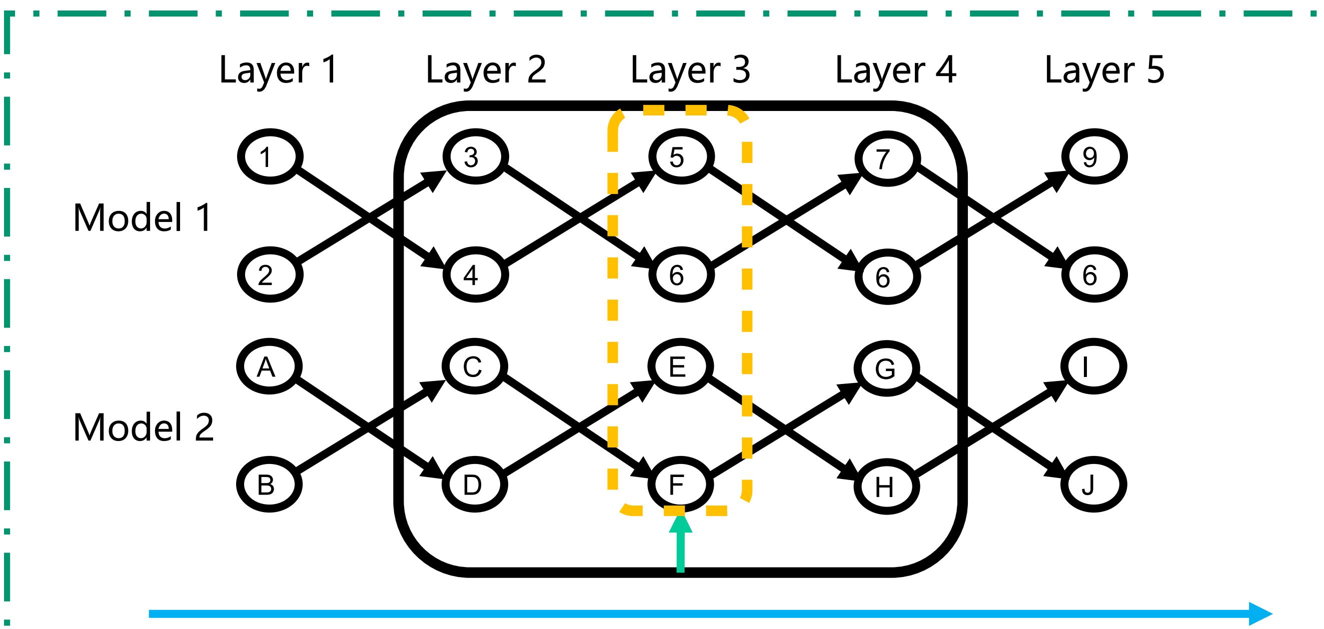
Graph Matching Based Model Fusion ICML22





Graduated Assignment

- Select 3 adjacent layers at a time
- Fix front and back layers
- Update the permutation matrix of the middle layer
- Iterate until convergence



Graph Optimization Problem of Placement and Routing

Background

Explore solving Placement and Routing via Machine Learning, as an alternative to classical algorithms

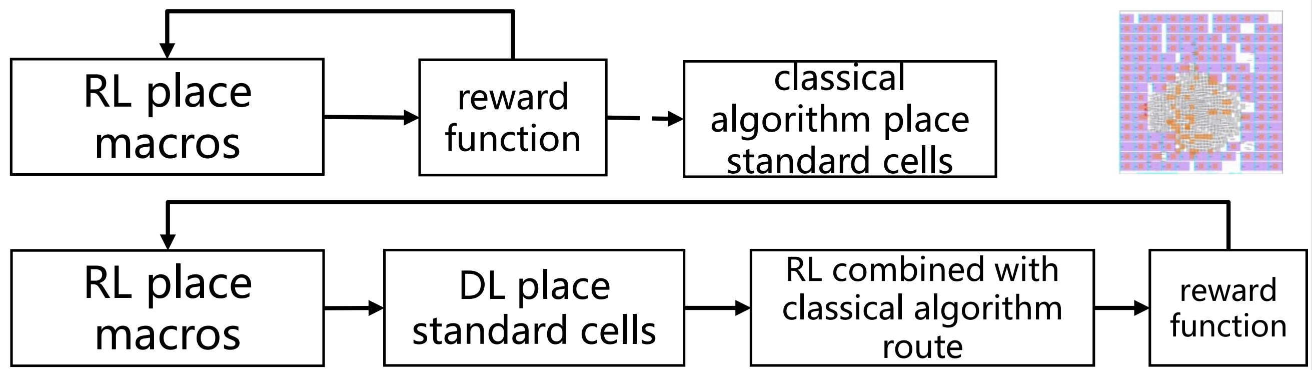
On Joint Learning for Solving Placement and Routing in Chip Design, NeurIPS21

Propose a Cyclic Placement and Routing Model

Google

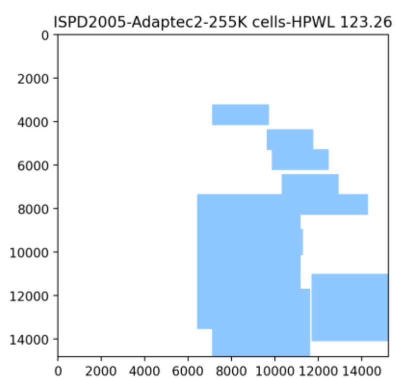
nature

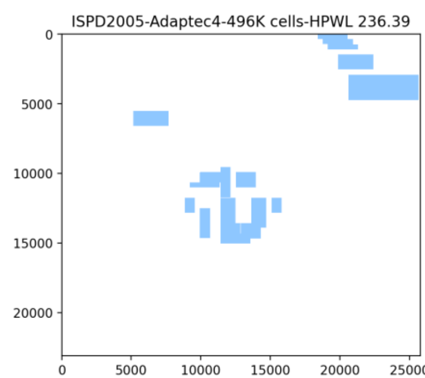
DeepPR (NeurIPS21)

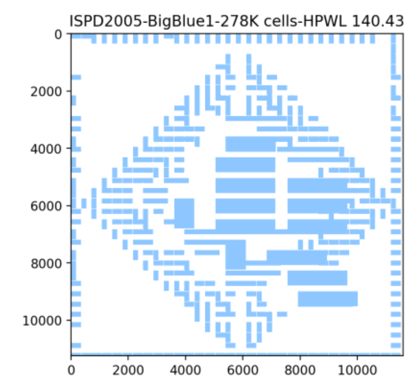


Cheng & Yan, On Joint Learning for Solving Placement and Routing in Chip Design, NeurIPS'21

ISPD 2005 Dataset wire-length 8%↓







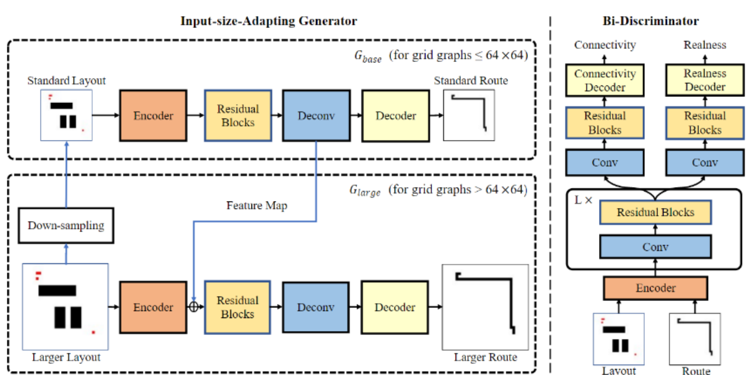
Graph Optimization Problem of Placement and Routing

The Policy-gradient Placement and Generative Routing Neural Networks for Chip Design, NeurIPS21

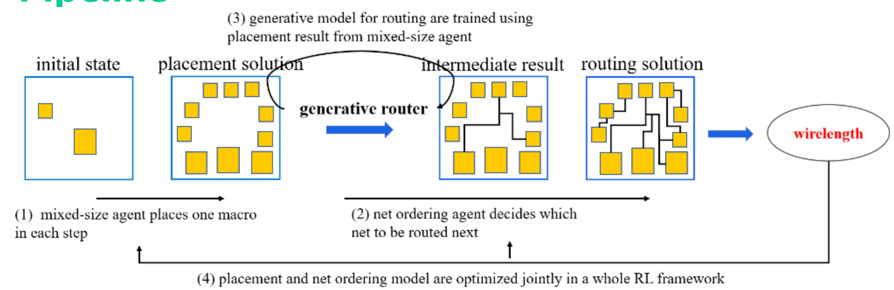
Formulation of Mixed-size Placement

- The key elements of the Markov Decision Processes (MDPs) for mixed-size placement are defined as follows:
- **State** s **:** the state representation consists of two part, global image I portrayed the layout and netlist graph H which contains detailed position of placed macros. The initial state $I_{xy} = 1$ if (x, y) has already been occupied before placement
- **Action** a **:** position (x_o, y_o) is available if all points p in the region R satisfy $I_p = 0$, where $R = \{(x,y) | |x - x_o| \leq \frac{h}{2}, |y - y_o| \leq \frac{w}{2}\}$.
- **Reward** r **:** to further control the overlap in the final placement, the reward at the end of episode is a negative weighted sum of wirelength, routing congestion and overlapping area: $R_E = -L_{wl} - \lambda_1 * L_{cg} - \lambda_2 * L_{ol}$

Architecture of Generative Routing Model



Neural Macro Placement and Routing Pipeline



- Combining the RL-based model for learning mixed-size macro placement with one-shot generative routing network to perform routing as we introduce above, we propose a pure neural pipeline for macro placement and routing.
- Inspired by EM algorithm, we first update the generative router using placement result from mixed-size agent (similar to E step), then placement and net order agents are learned jointly in a whole reinforcement learning framework to minimize wirelength calculated by trained generative model (corresponding to M step)
- The generator is composed of a basic generator for the input size of 64 × 64 or below and an extension for the input size of larger than 64 × 64. The discriminator consists of two sub-discriminators to estimate routes from validity and realness.

Graph Optimization Problem of Placement and Routing

The Policy-gradient Placement and Generative Routing Neural Networks for Chip Design, NeurIPS21

Results on Mixed-size Placement

- With only a slight increase of the total wirelength (within 1.3% difference on average), our mixed-size macro placer achieves approximately 4× reduction over DeepPlace on the overlapping area, stressing the importance of modeling macro's shape in state space.

Circuit	# Cells	# Mov.	Mixed-size technique (ours)		DeepPlace [1]	
			Wirelength ↓	Overlap Area↓	Wirelength ↓	Overlap Area↓
adaptec1	211K	514	82783826	**12606828**	80117232	66608273
adaptec2	255K	542	123307824	**19485631**	123265964	47085963
adaptec3	451K	710	232373680	**58588016**	241072304	140272759
adaptec4	496K	1309	234008876	**73075220**	236391936	169853555
bigblue1	278K	551	141020208	**2041890**	140435296	3519755
bigblue2	558K	948	144803296	**70702107**	140465488	103663199
bigblue3	1097K	1227	468632064	**39664931**	450633360	574956948
bigblue4	2177K	659	1001315712	**67794270**	951984128	87630042
ratio	-	-	1.000	1.0	0.987	3.9

Results on Routing

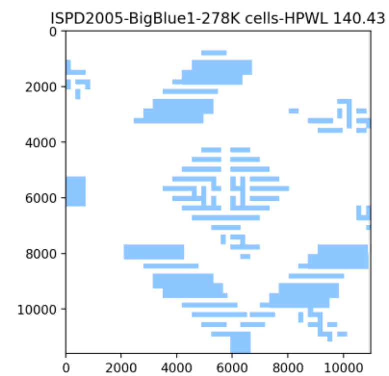
- We compare the full version with ResNet-based cGAN, as well as the pure ResNet generator. The ResNet generator outdoes the cGAN, but the bi-discriminator significantly improves the generator. Moreover, the enhanced loss improves the wirelength at the marginal expense of correctness.

Results on Overall Placement and Routing

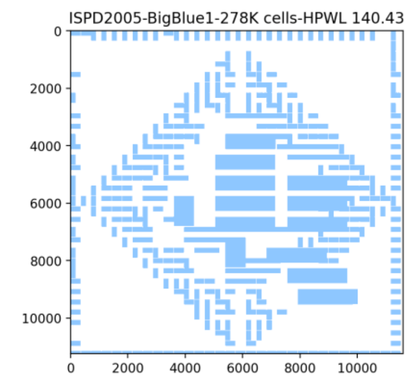
variants of our PRNet	adaptec1		adaptec3	
	WL↓	RC↓	WL↓	RC↓
RL-based Placer (i.e. DeepPlace [1])	6149	10.565	30154	62.751
RL-based Placer + GR	5940	10.464	29711	73.324
RL-based Placer + GR + NOL (full version of PRNet)	**5787**	**9.386**	**29462**	**43.207**

- We compare our PRNet with DeepPlace, along with an ablation study to verify the impact of net order learning. For all test cases, our neural placement and routing pipeline outperforms the other two methods in terms of both wirelength (WL) and routing congestion (RC). The significant difference in routing congestion without net order learning indicates that net order agent is able to arrange the sequence of routing efficiently.

our router w/ different generative models	Route-small-4		Route-small	
	CrrtR↑	WLR↓	CrrtR↑	WLR↓
CVAE*(CNN) [9]	0.414±0.020	1.179±0.033	0.397±0.008	1.042±0.006
CVAE*-cGAN(CNN)	0.557±0.065	1.292±0.108	0.439±0.021	1.315±0.015
CVAE*-bcGAN(CNN)	0.474±0.048	1.525±0.029	0.488±0.007	1.241±0.012
U-Net* [39]	0.724±0.001	3.306±0.266	0.524±0.005	1.232±0.016
cGAN(U-Net*) [29]	0.602±0.009	1.028±0.001	0.532±0.011	1.286±0.022
bcGAN(U-Net*)	0.721±0.012	1.134±0.055	0.552±0.007	1.104±0.054
ResNet [40]	0.783±0.002	1.023±0.003	0.594±0.004	1.030±0.007
cGAN(ResNet)	0.698±0.010	1.073±0.011	0.568±0.020	1.320±0.151
bcGAN(ResNet)	0.804±0.021	1.035±0.013	**0.738**±0.005	1.036±0.002
bcGAN(ResNet)+EL (full version of our router)	**0.814**±0.001	**1.010**±0.000	0.735±0.010	**1.018**±0.004



DeepPlace

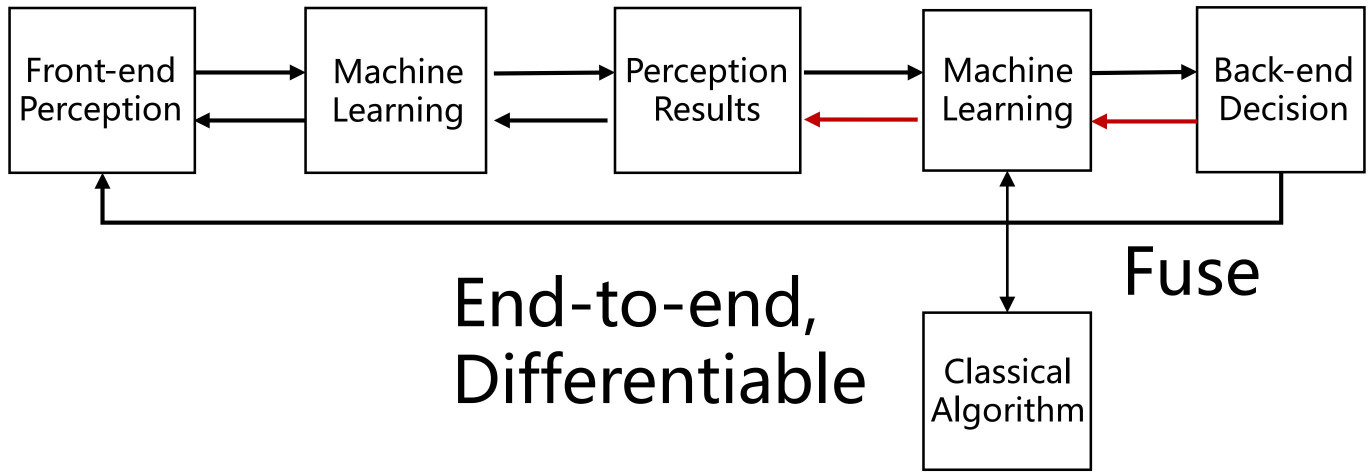


Our Mixed-size Placer

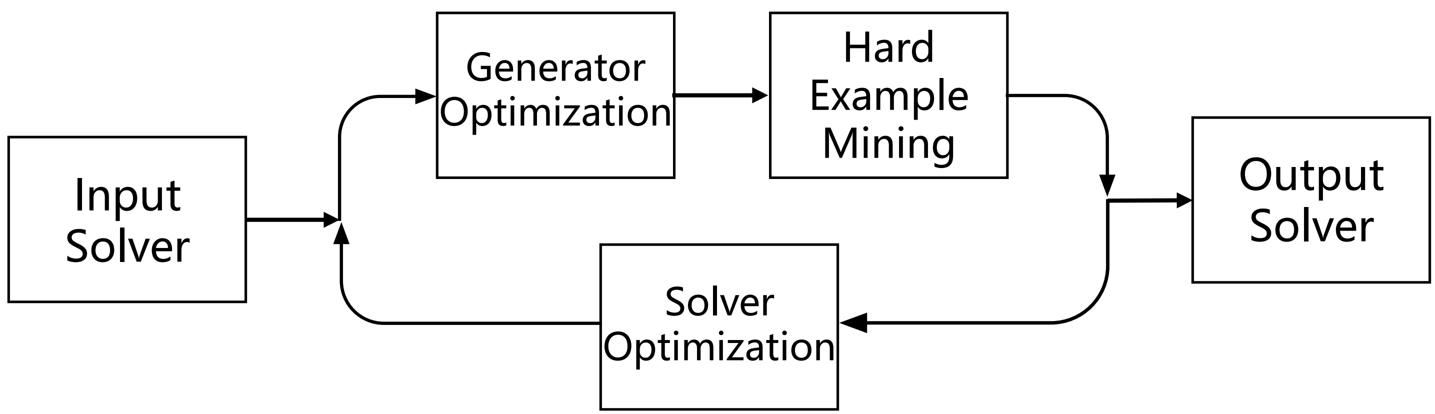
1. Background of Research
2. Recent Work
3. Summary and Outlook

Some Thoughts on Typical Paradigms

Paradigm1: Differentiable learning to improve overall front- and back-end agility



Paradigm2: Multi-task distributed self-supervised learning to improve generalizability



Thanks and Q&A

Awesome Machine Learning for Combinatorial Optimization Resources

We would like to maintain a list of resources that utilize machine learning technologies to solve combinatorial optimization problems.

We mark work contributed by Thinklab with ✨.

Maintained by members in SJTU-Thinklab: Chang Liu, Runzhong Wang, Jiayi Zhang, Zelin Zhao, Haoyu Geng, Tianzhe Wang, Wenxuan Guo, Wenjie Wu and Junchi Yan. We also thank all contributers from the community!

We are looking for post-docs interested in machine learning especially for learning combinatorial solvers, dynamic graphs, and reinforcement learning. Please send your up-to-date resume via yanjunchi AT sjtu.edu.cn.

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