Graph Neural Network for Heterogeneous Graph

Ph.D candidate in Computational Science and Engineering
Yonsei Univ.

Jin-Duk Park

Reading group material

What is Heterogeneous Graph (HG)?

Definition of HG

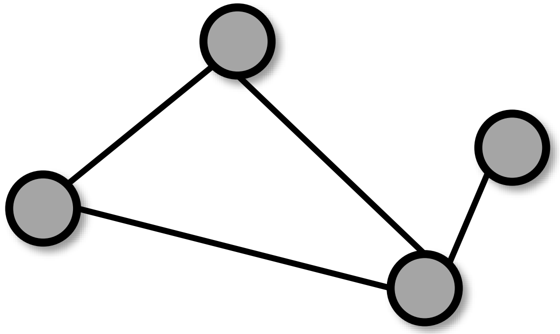
$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$

$$\pi : \mathcal{V} \rightarrow \mathcal{A} \qquad \psi : \mathcal{E} \rightarrow \mathcal{R}$$

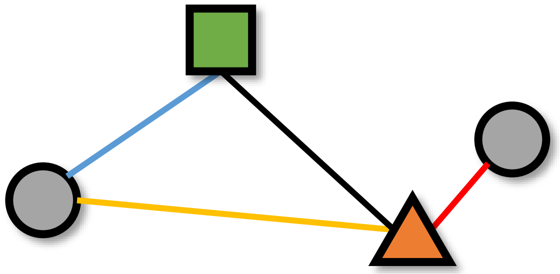
(node type mapping) (edge type mapping)

$$\boxed{|\mathcal{A}| + |\mathcal{R}| > 2}$$

Standard (homogeneous) Graph

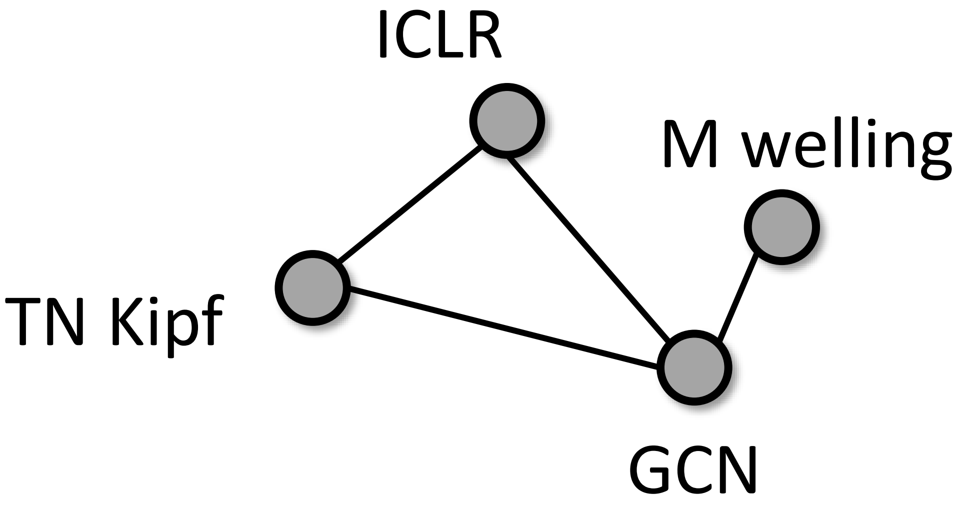


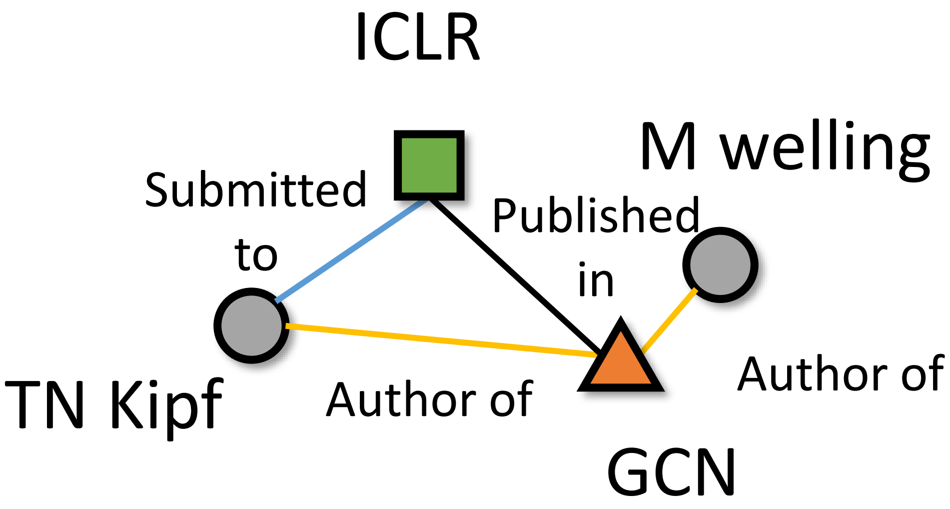
Heterogeneous Graph



Why HG?

Expressiveness

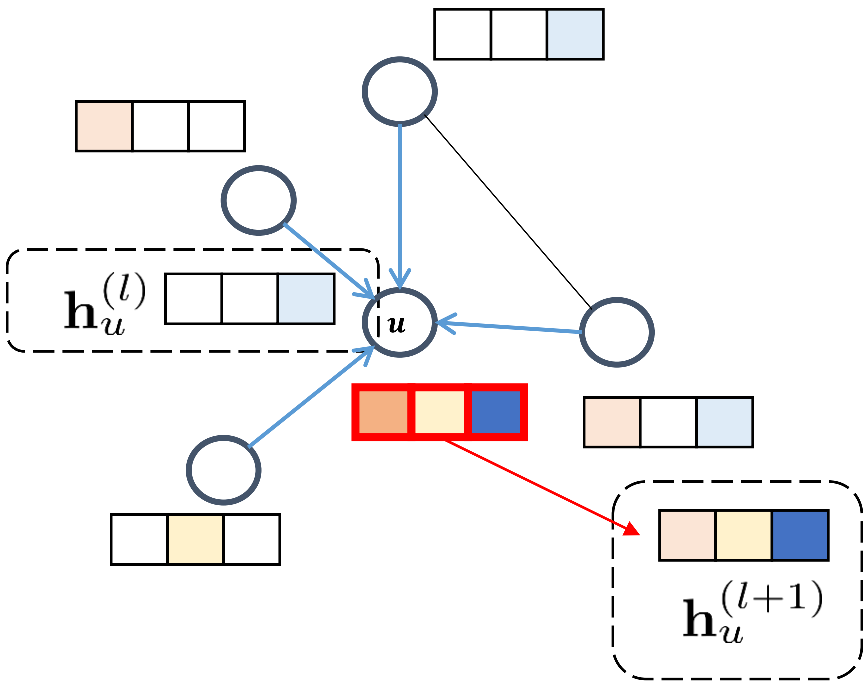




It can further express rich semantics
within different types of nodes and its relations

Graph Neural Network (GNN)

GNNs



Active GNN research:
High expressiveness in graph

Natural question:
How to carry out GNNs in HG?

Representative work (1)

Heterogeneous Graph Attention Network

Xiao Wang, Houye Ji
Beijing University of Posts and Telecommunications
Beijing, China
{xiaowang,jhy1993}@bupt.edu.cn

Chuan Shi*, Bai Wang
Beijing University of Posts and Telecommunications
Beijing, China
{shichuan,wangbai}@bupt.edu.cn

Peng Cui, P. Yu
Tsinghua University
Beijing, China
{cuip,psyu}@tsinghua.edu.cn

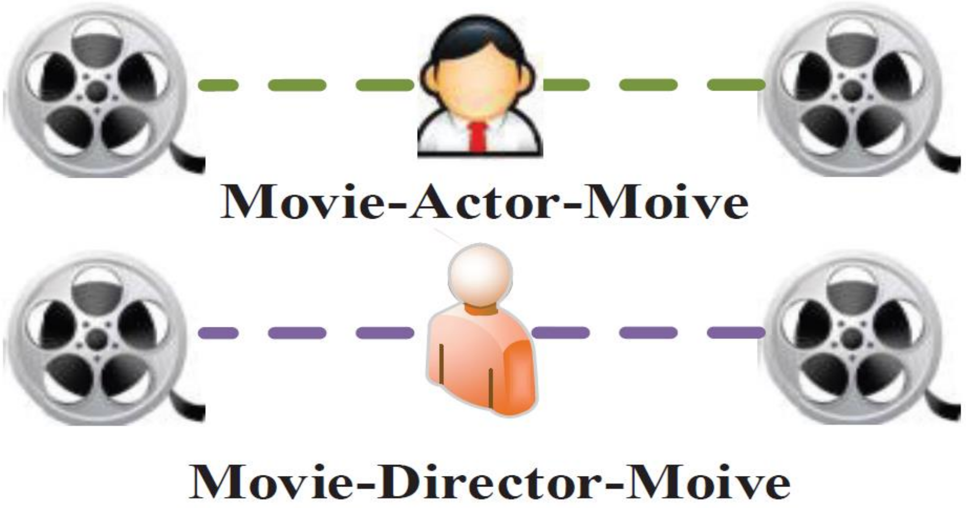
Yanfang Ye
West Virginia University
WV, USA
yanfang.ye@mail.wvu.edu

-The first attempt of GNN on HG, WWW'19

What is Meta-Path?

Meta-path

- User-defined path
- Different meta-paths reveal different semantics



e.g.)
M-A-M
M-D-M

What is Meta-Path?

Meta-path

Formally,

- Adjacency matrix of HG:

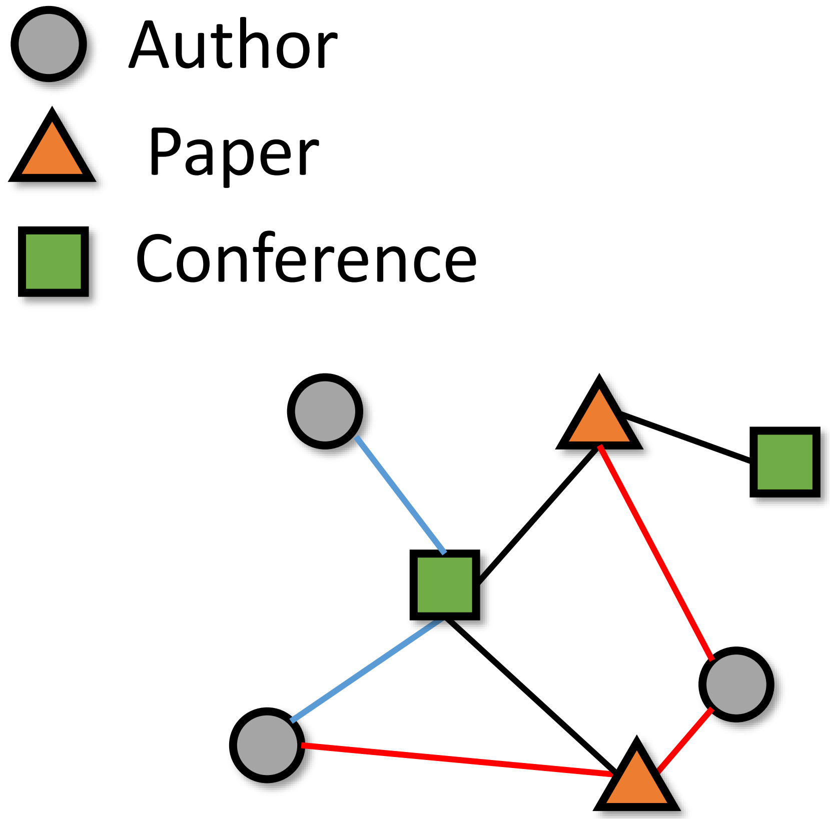
$$\{A_k\}_{k=1}^{|\mathcal{R}|}$$

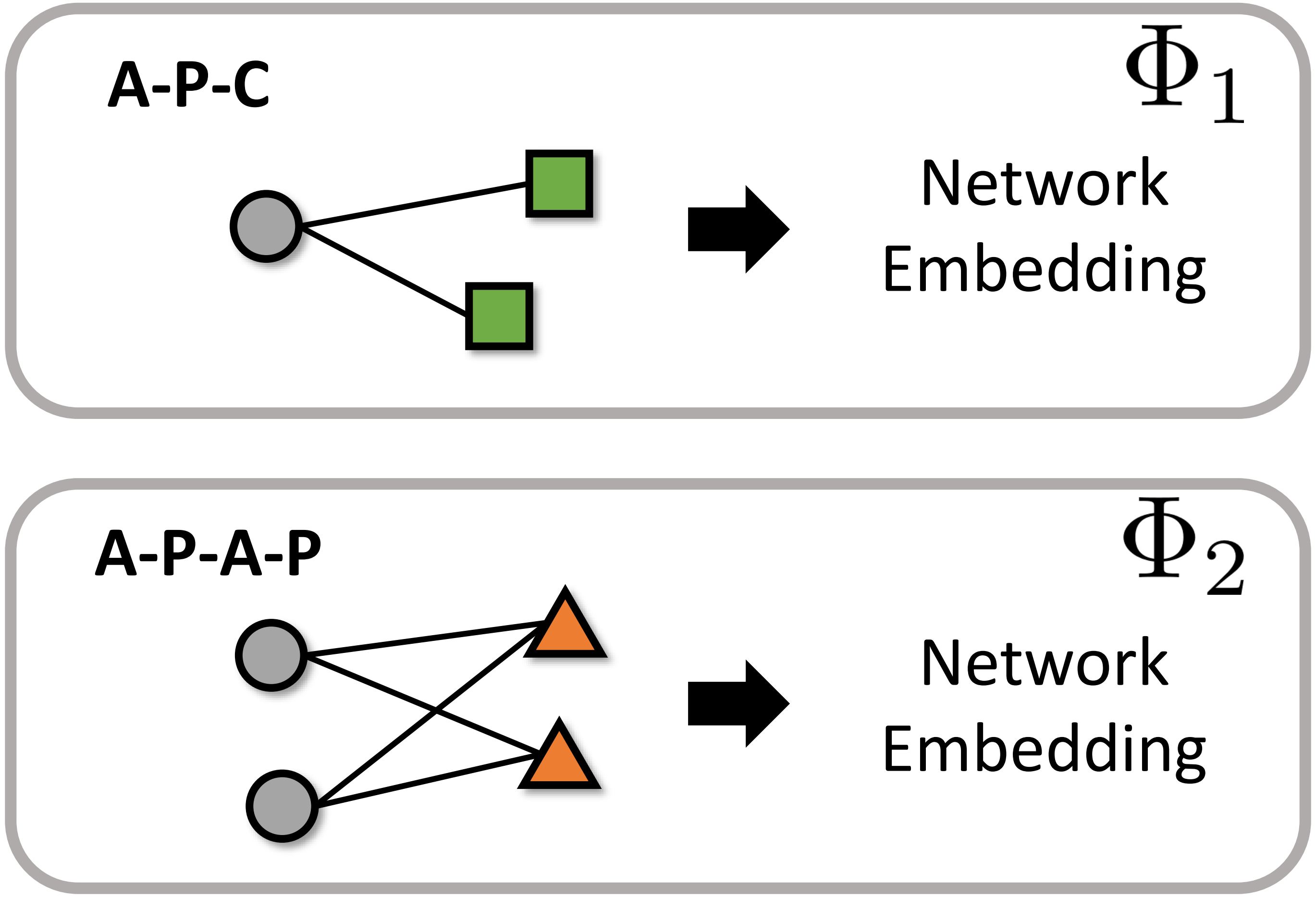
- Meta-path via matrix multiplication

$$A_{PAC} = A_{PA}A_{AC}$$

What is Meta-Path?

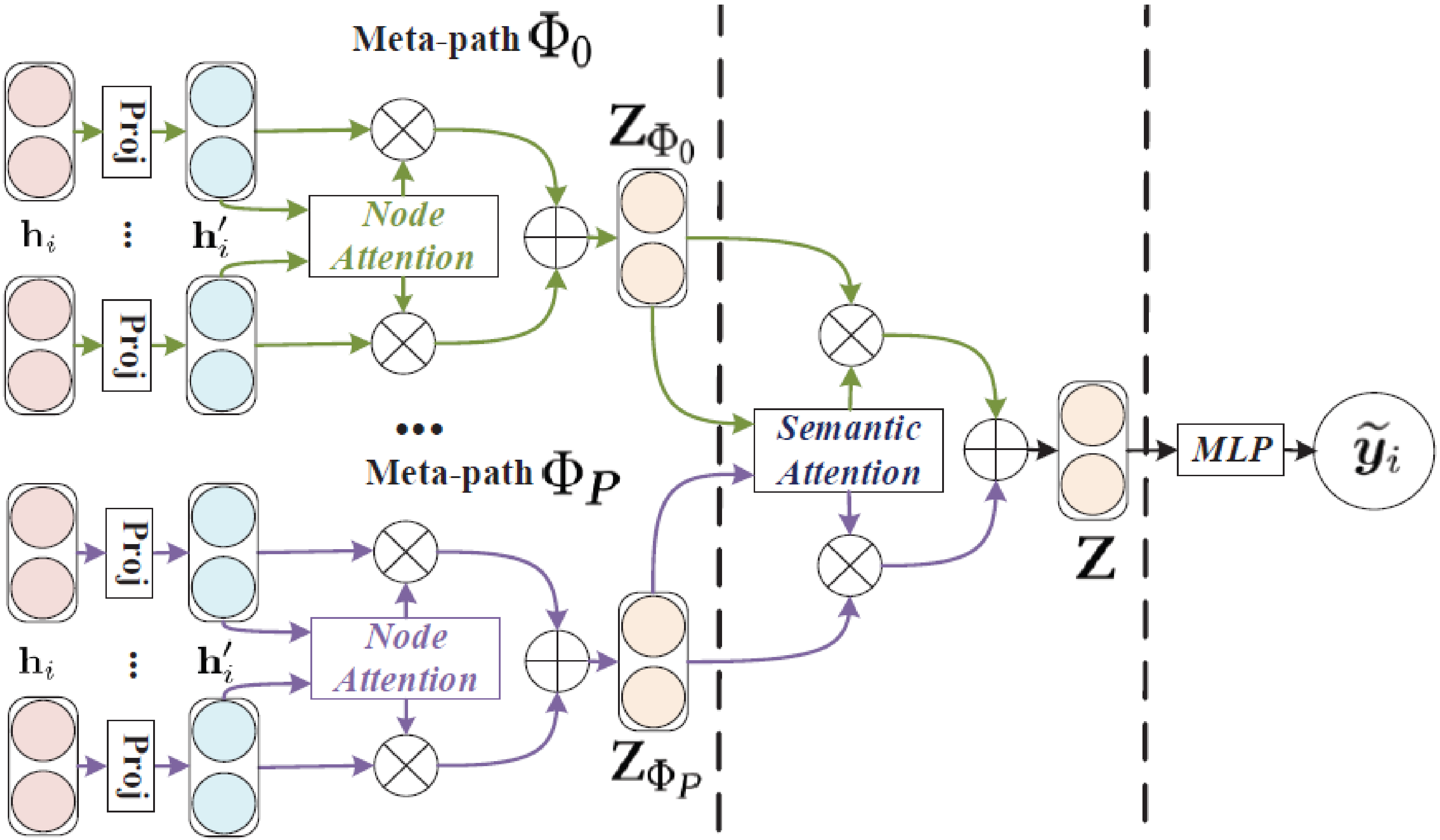
Meta-path for HG embedding





Components of HAN

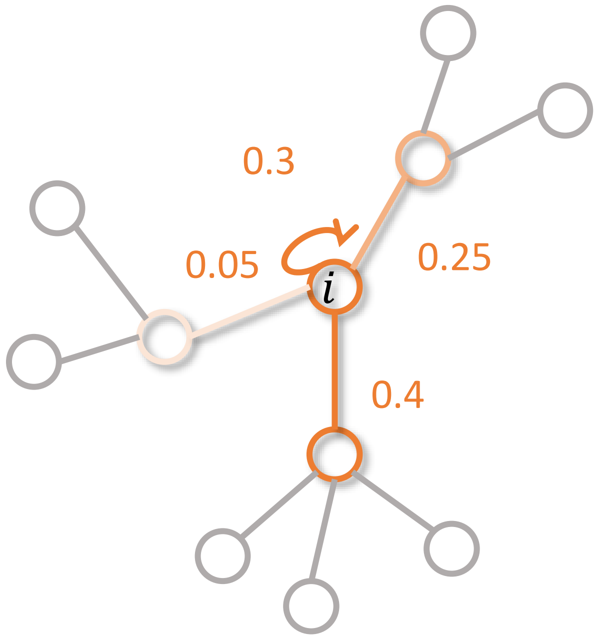
Overview of HAN



(a) Node-Level Attention (b) Semantic-Level Attention (c) Prediction

A Short Summary of GAT

GAT overview



Step 1. Masked self-attention

Only compute attention coefficients for nodes in the neighbors

$$j \in \mathcal{N}_i$$

$$\alpha_{ij} = \frac{\exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

Normalize with softmax

Node Level Attention

1. Mapping to same feature space

$$\mathbf{h}'_i = \mathbf{M}_{\phi_i} \cdot \mathbf{h}_i$$

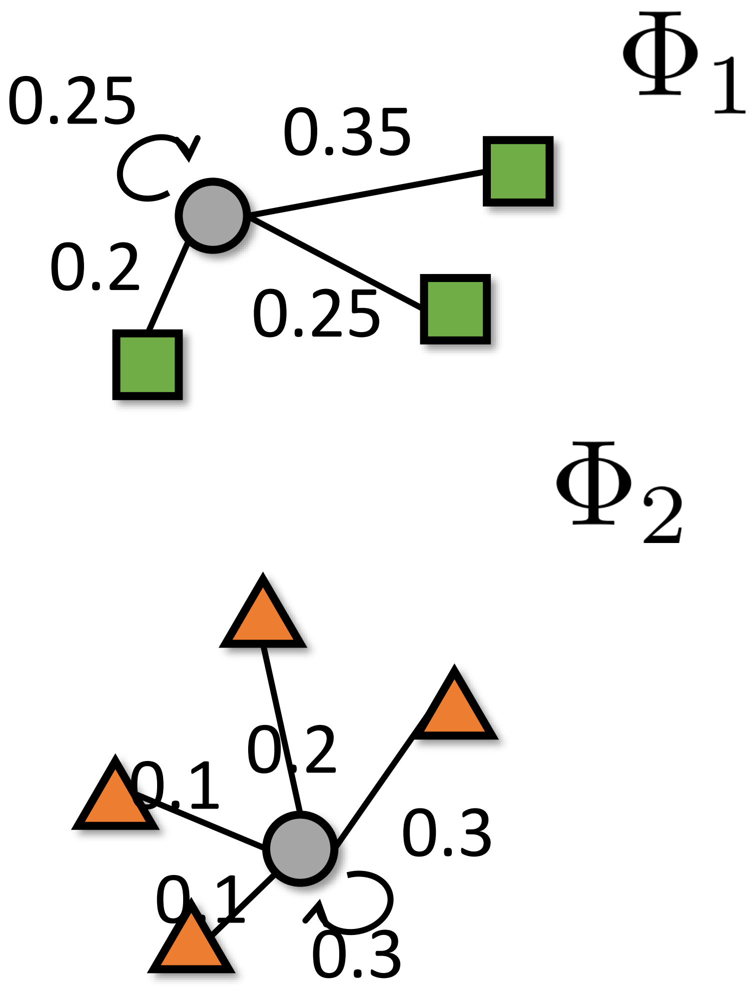
2. Masked Attention

Meta-path-specific

$$\alpha_{ij}^{\Phi} = softmax_j(e_{ij}^{\Phi}) = \frac{\exp\left(\sigma\left(\boxed{\mathbf{a}_{\Phi}^{\mathrm{T}}} \cdot [\mathbf{h}'_i \| \mathbf{h}'_j]\right)\right)}{\sum_{\boxed{k \in \mathcal{N}_i^{\Phi}}} \exp\left(\sigma(\mathbf{a}_{\Phi}^{\mathrm{T}} \cdot [\mathbf{h}'_i \| \mathbf{h}'_k])\right)}$$

Meta-path-based neighbors!

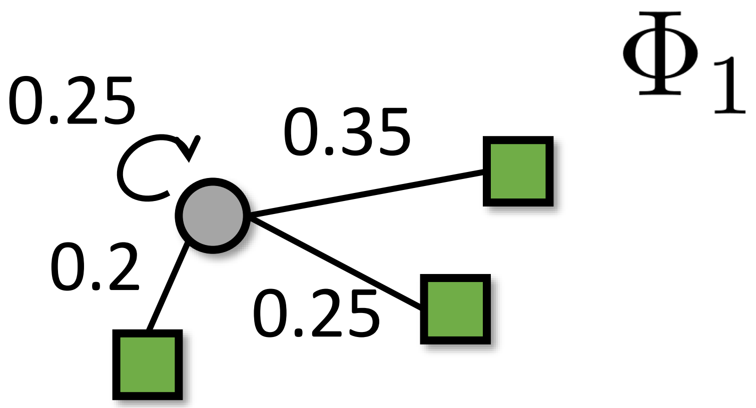
(including self)



Node Level Attention

1. Mapping to same feature space

$$\mathbf{h}'_i = \mathbf{M}_{\phi_i} \cdot \mathbf{h}_i$$



2. Masked Attention

Meta-path-

$$\alpha_{ij}^{\Phi} = softmax_j(e_{ij}^{\Phi}) = \frac{\exp\left(\sigma\left(\boxed{\mathbf{a}_{\Phi}^{\mathrm{T}}}\right.\right.}{\sum_{\boxed{k \in \mathcal{N}_i^{\Phi}}} \exp\left(\sigma\right.}$$

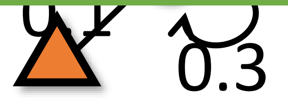
We can get

(# of the meta-path)

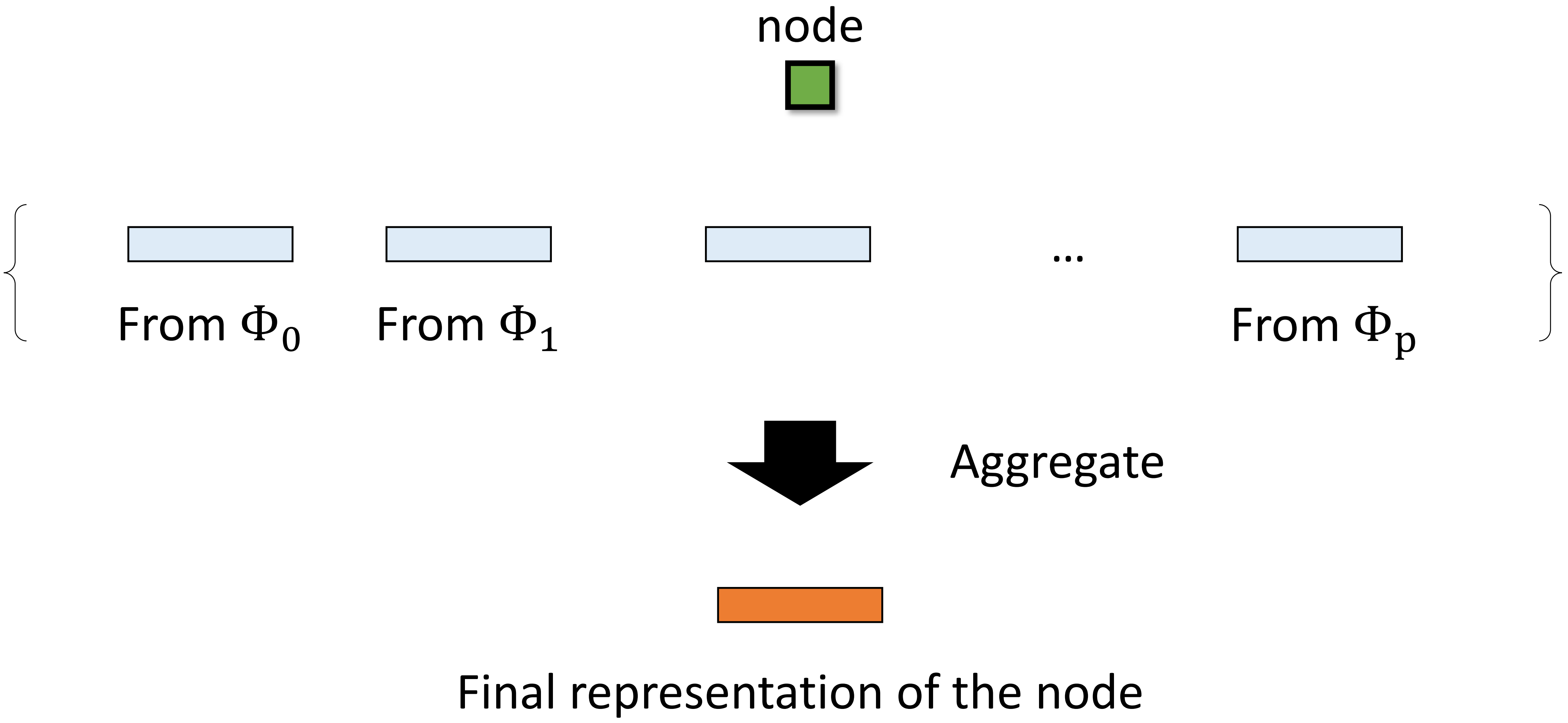
representations for each node

Meta-path-based neighbo

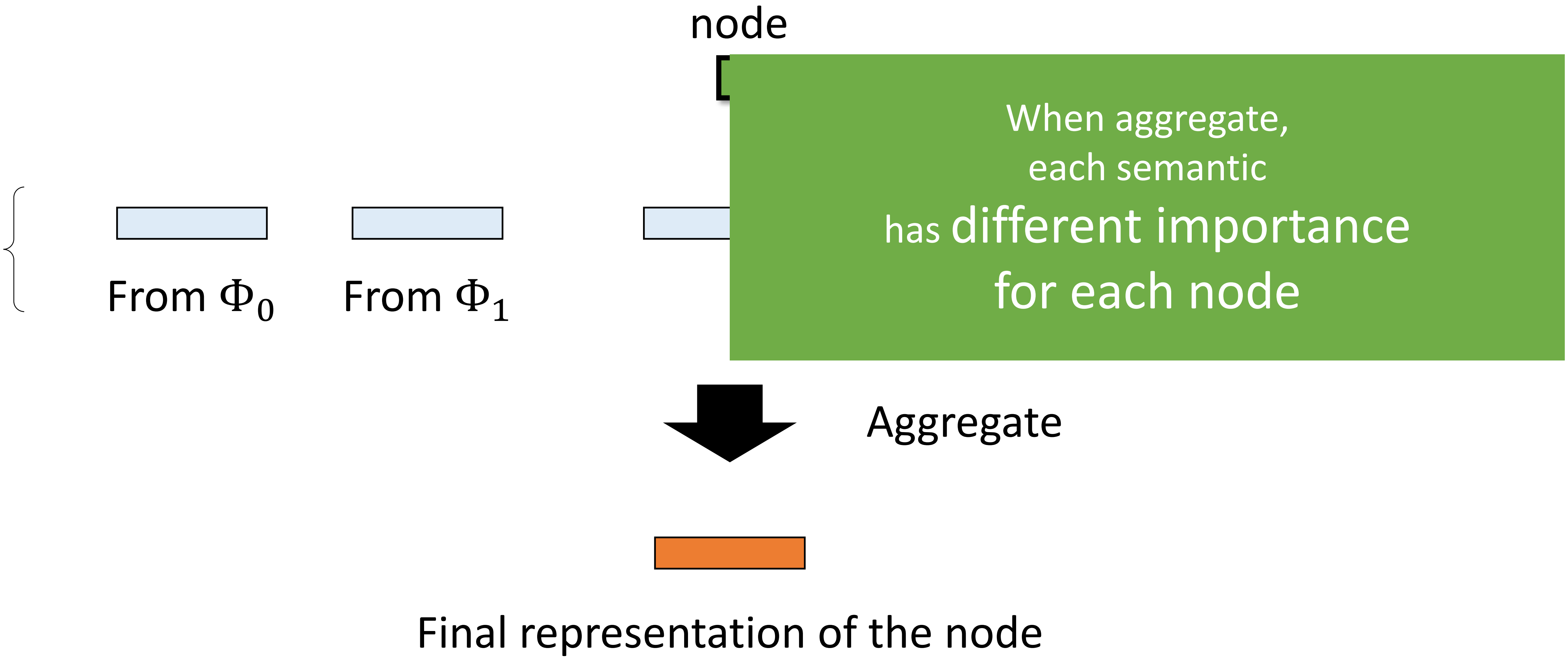
(including self)



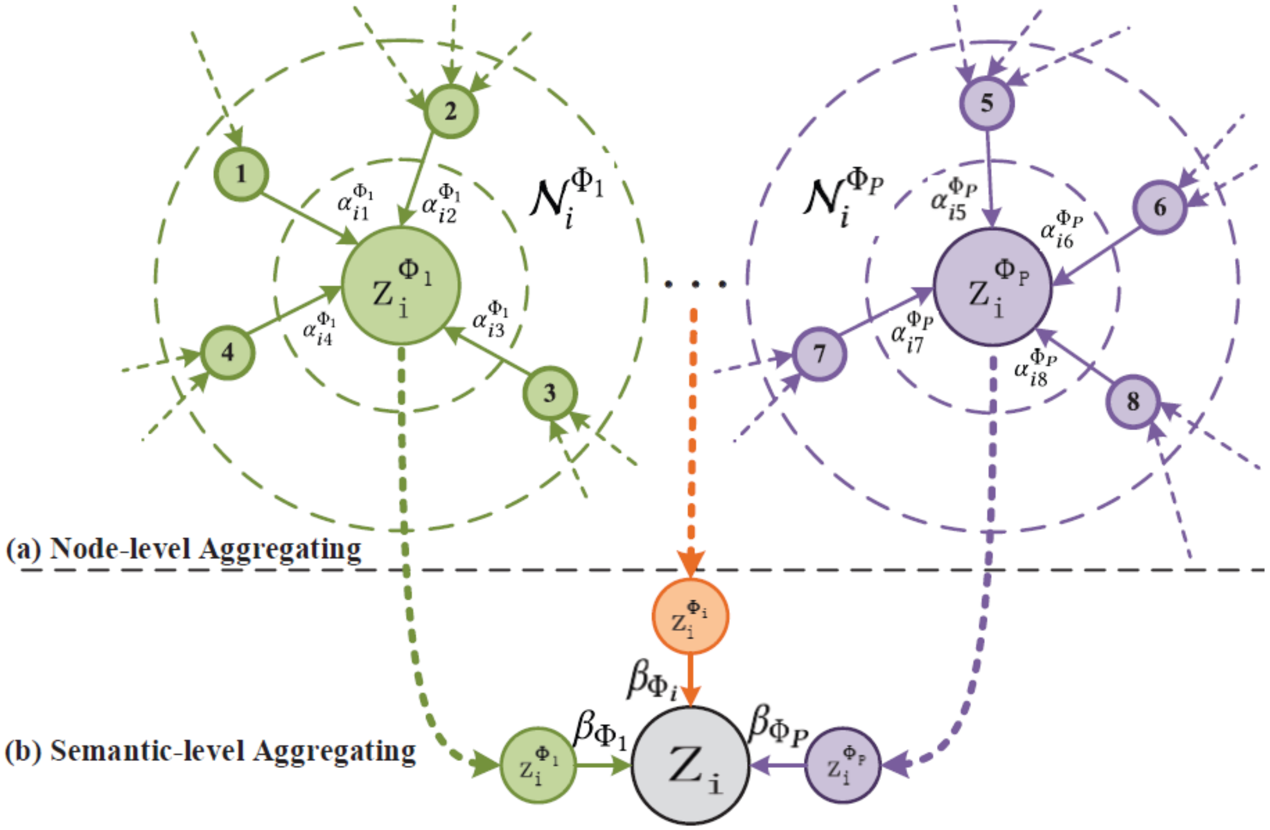
Semantic Level Attention



Semantic Level Attention

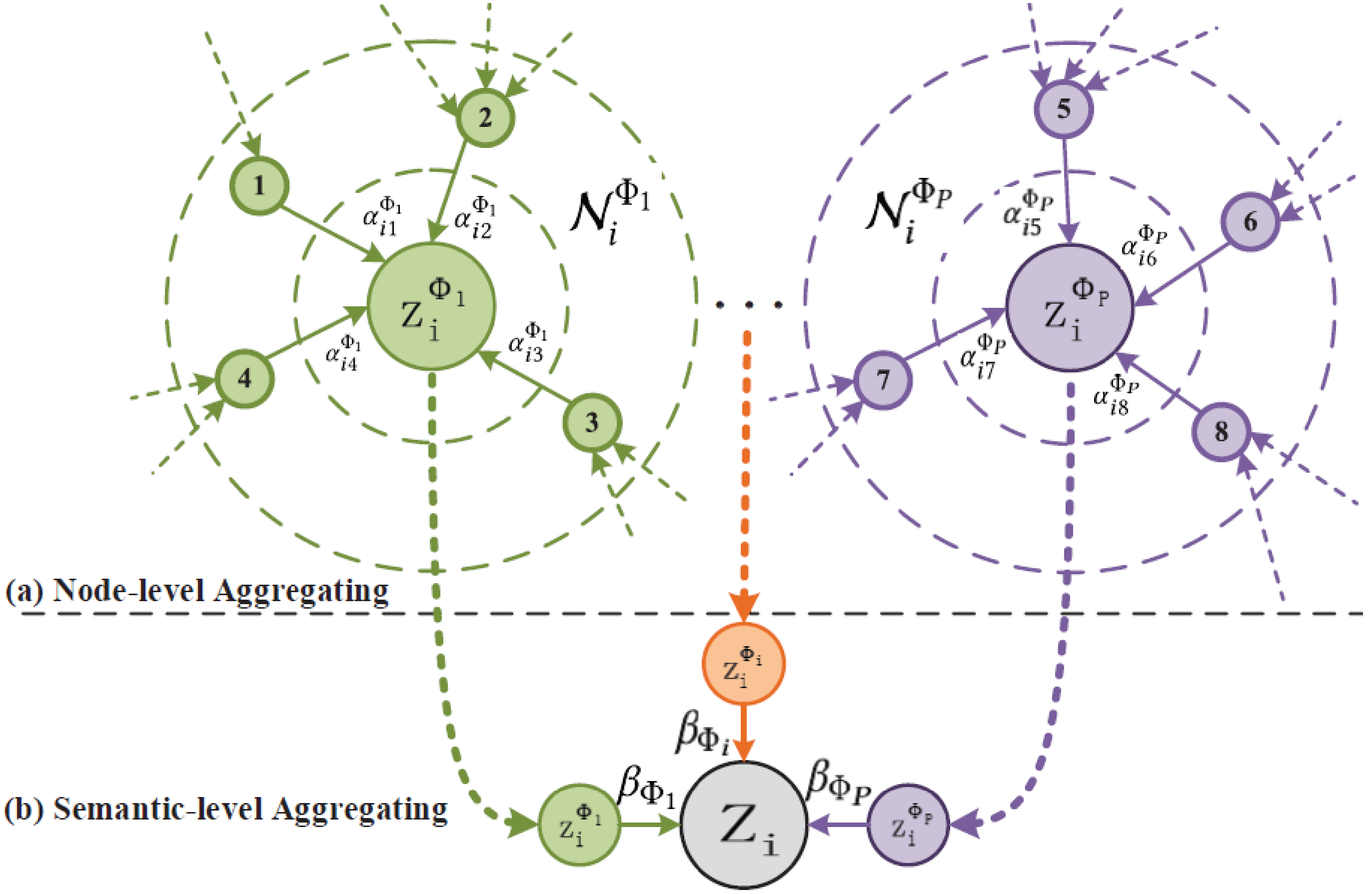


Semantic Level Attention



$$(\beta_{\Phi_0}, \beta_{\Phi_1}, \ldots, \beta_{\Phi_P}) = att_{sem}(\mathbf{Z}_{\Phi_0}, \mathbf{Z}_{\Phi_1}, \ldots, \mathbf{Z}_{\Phi_P})$$

Semantic Level Attention



1-layer MLP

$$w_{\Phi_i} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \mathbf{q}^{\mathrm{T}} \cdot \boxed{\tanh(\mathbf{W} \cdot \mathbf{z}_i^{\Phi} + \mathbf{b})}$$

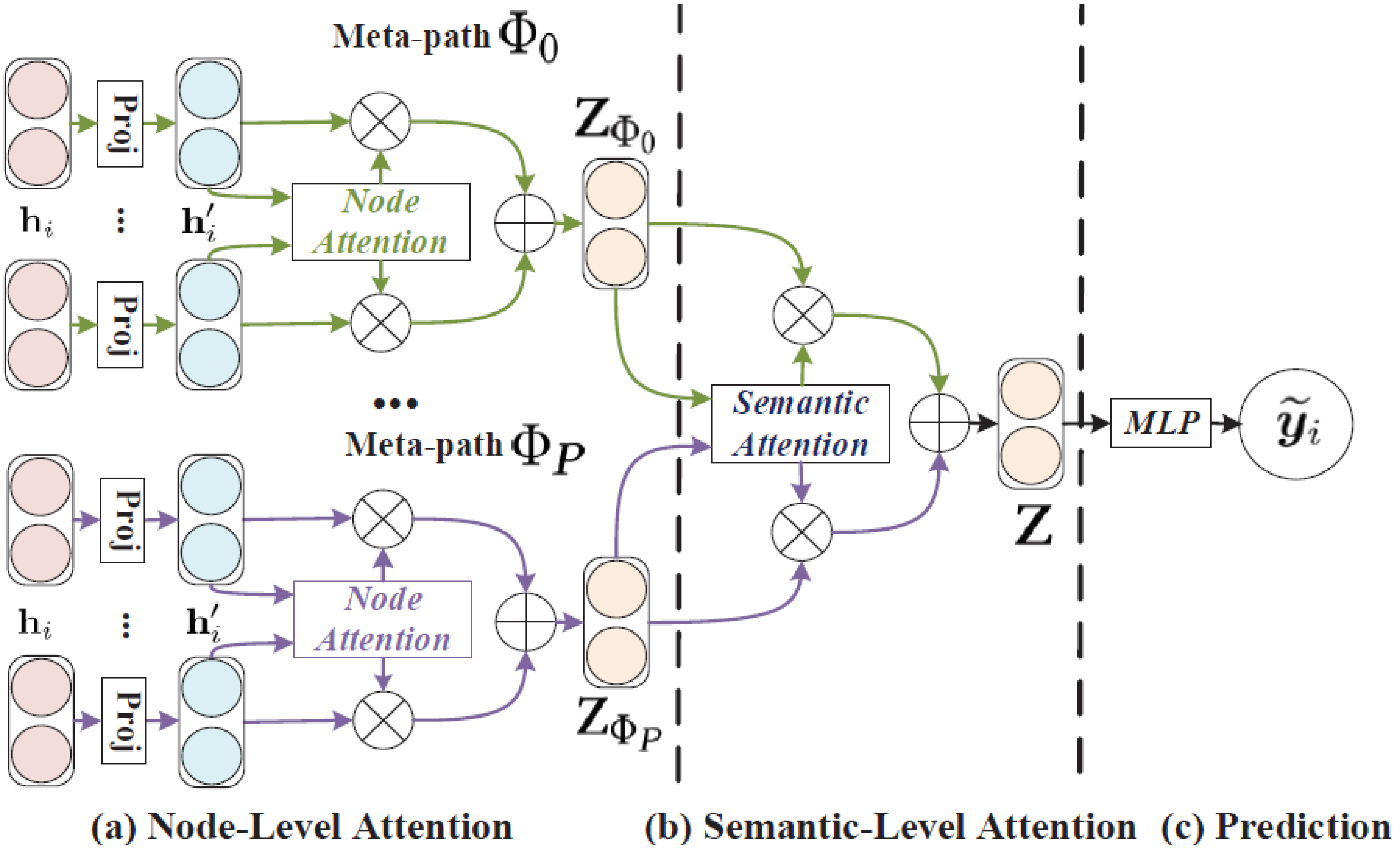
q: semantic level attention vector

$$\beta_{\Phi_i} = \frac{\exp(w_{\Phi_i})}{\sum_{i=1}^{P} \exp(w_{\Phi_i})}$$

$$\mathbf{Z} = \sum_{i=1}^{P} \beta_{\Phi_i} \cdot \mathbf{Z}_{\Phi_i}$$

Final representation of the node

HAN


Meta-path Φ_0
Proj
Proj
$\mathbf{h}_i$
$\mathbf{h}'_i$
Node Attention
$\mathbf{Z}_{\Phi_0}$
Meta-path Φ_P
Proj
Proj
$\mathbf{h}_i$
$\mathbf{h}'_i$
Node Attention
$\mathbf{Z}_{\Phi_P}$
Semantic Attention
$\mathbf{Z}$
MLP
$\tilde{y}_i$
(a) Node-Level Attention
(b) Semantic-Level Attention
(c) Prediction

Graph Transformer Networks

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang* , Hyunwoo J. Kim*
Department of Computer Science and Engineering
Korea University
{ysj5419, minbyuljeong, raehyun, kangj, hyunwoojkim}@korea.ac.kr

NIPS'2019

Motivation of Graph Transformer Networks (GTN)

Conventional work:

Needs handcrafted meta-path

Not robust,
Lack of generality

In IMDB:
MAM, MDM

In ACM:
PAP,PSP

In DBLP:
APA, APCPA, APTPA

Motivation of Graph Transformer Networks (GTN)

Conventional work:

Needs handcrafted meta-path

⬇

Not robust,
Lack of generality

In IMDB:
MAM, MDM

In ACM:
PAP,PSP

Natural question:

How to automatically find the meta-path?

Recall Meta-path

Meta-path

Formally,

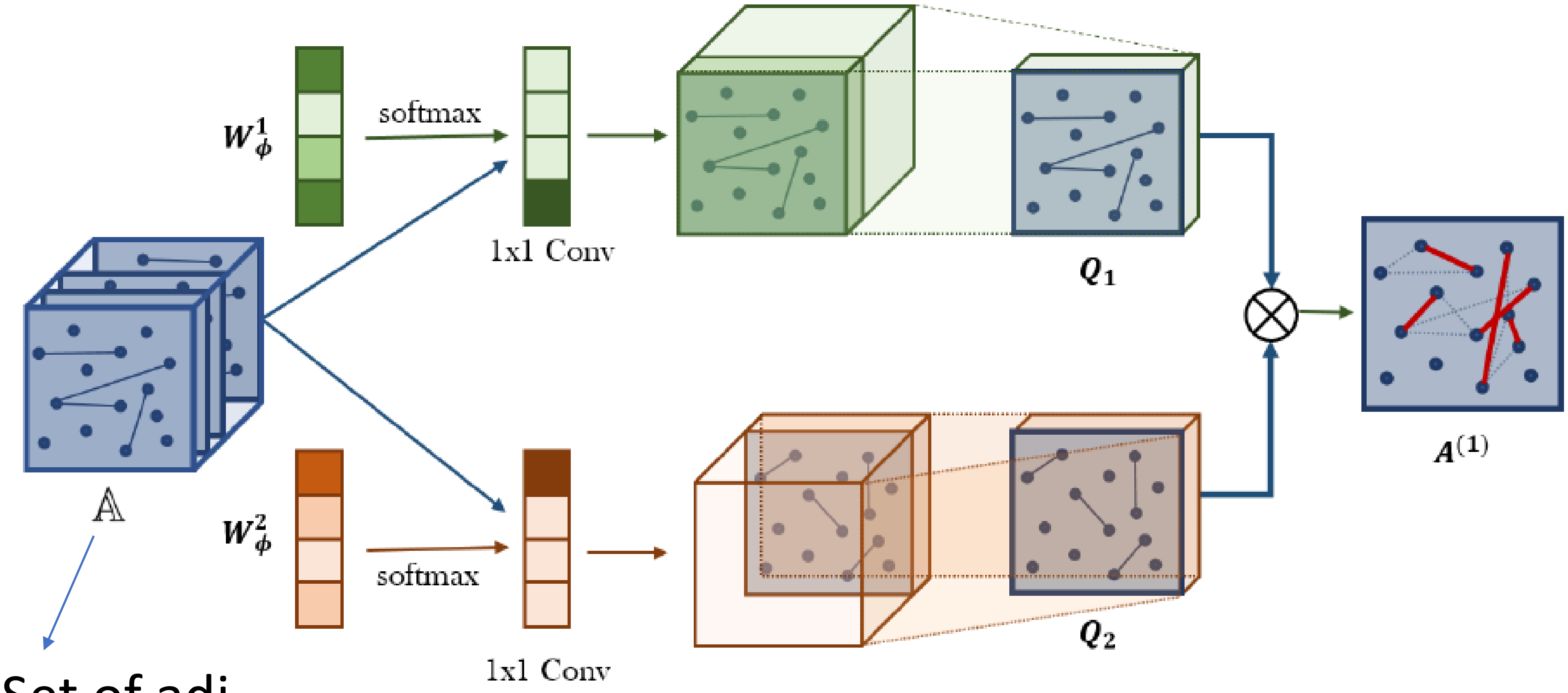
- Adjacency matrix of HG:

$$\{A_k\}_{k=1}^{|\mathcal{R}|}$$

- Meta-path via **matrix multiplication**

$$A_{PAC} = A_{PA} A_{AC}$$

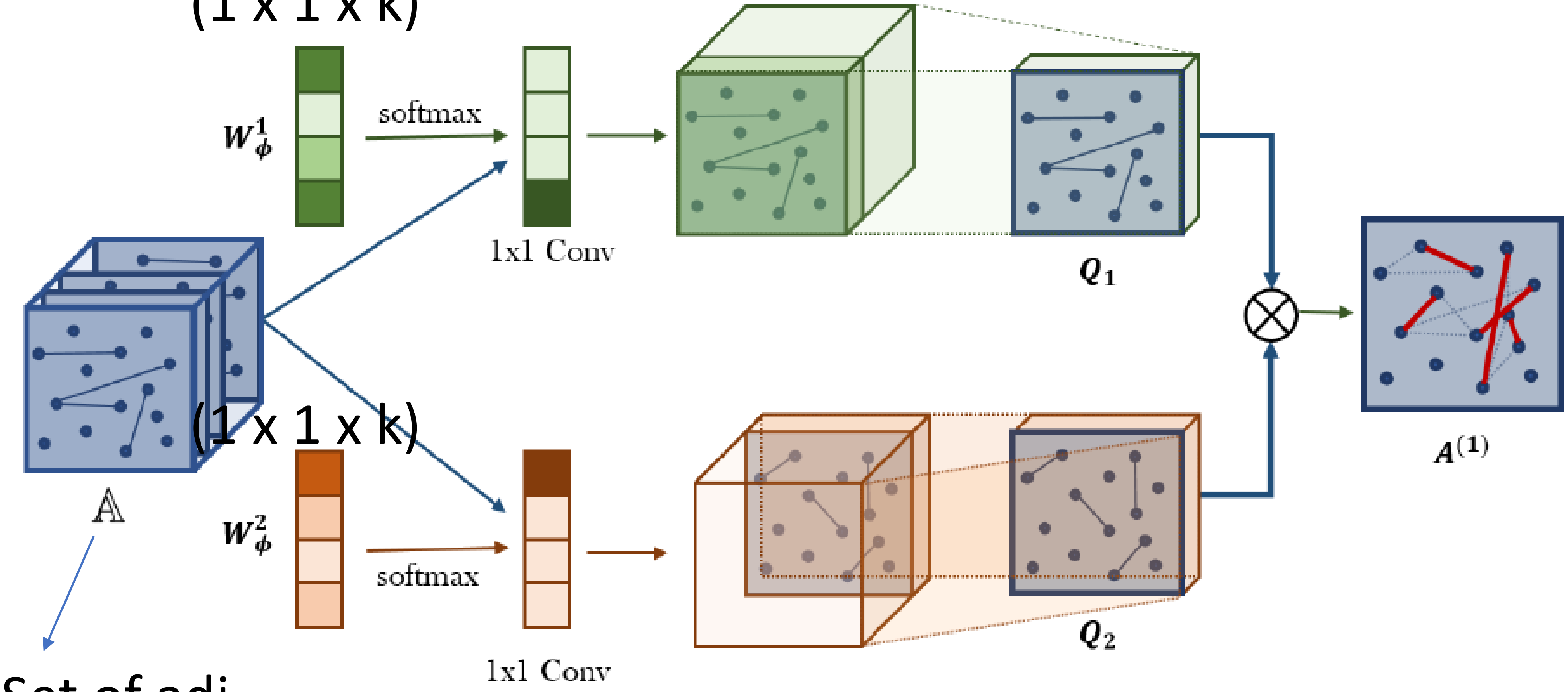
Graph Transformer Layer



1. Set of adj.
for every (k) edge types
(n x n x k)

Graph Transformer Layer

(1 x 1 x k)



(1 x 1 x k)

1. Set of adj.
for every (k) edge types
(n x n x k)

2. *Softly* select two adj. $(\boldsymbol{Q}_1, \boldsymbol{Q}_2)$.

$$\boldsymbol{A}_{t_1} = \boldsymbol{Q}_1 \boldsymbol{Q}_2$$

Graph Transformer Networks

If we stack multiple GT layers..

From convolution weights

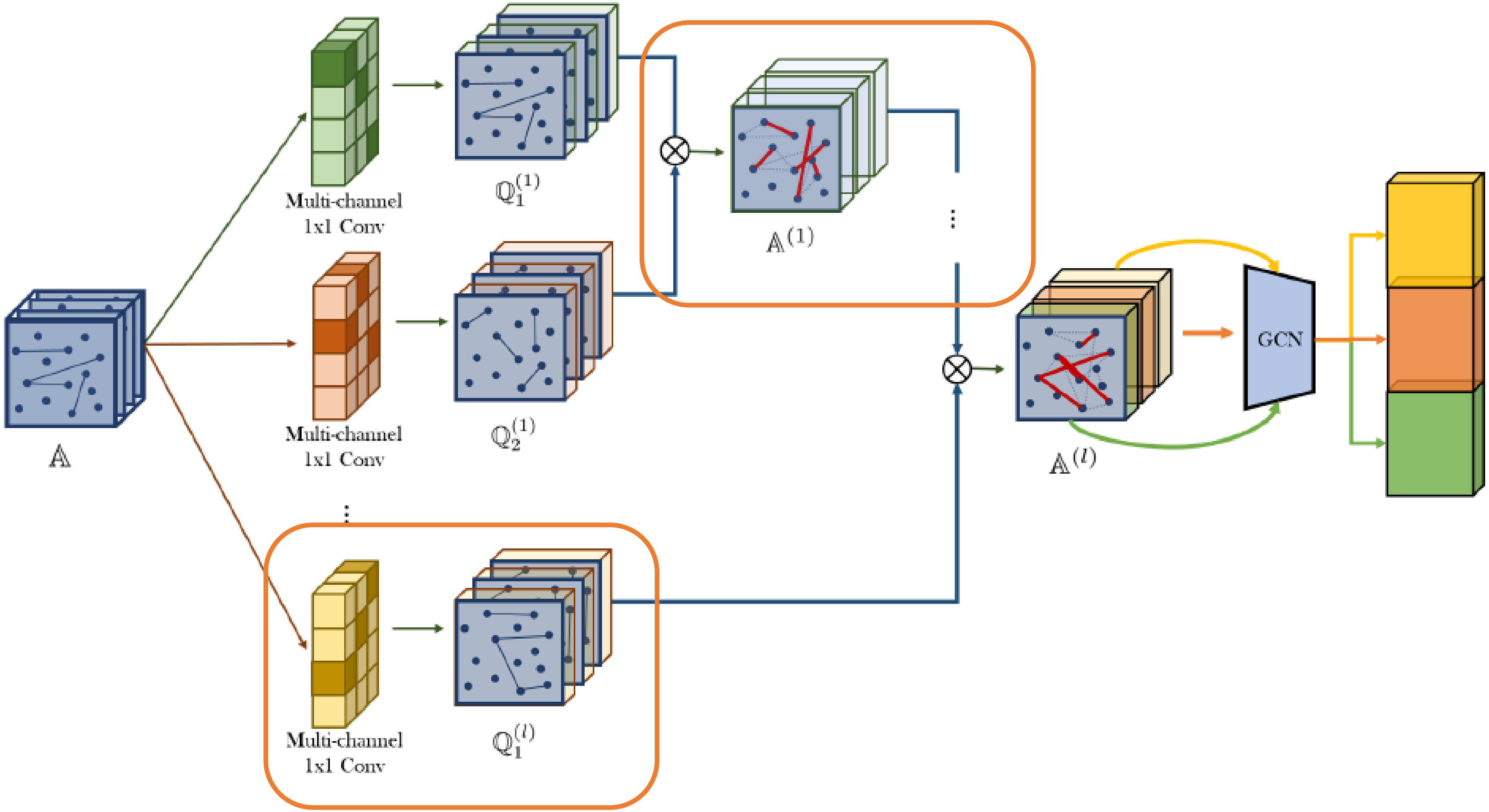
$$A_P = \left(\sum_{t_1 \in \mathcal{T}^e} \alpha_{t_1}^{(1)} A_{t_1}\right)\left(\sum_{t_2 \in \mathcal{T}^e} \alpha_{t_2}^{(2)} A_{t_2}\right) \dots \left(\sum_{t_l \in \mathcal{T}^e} \alpha_{t_l}^{(l)} A_{t_l}\right)$$

Arbitrary l-length meta path

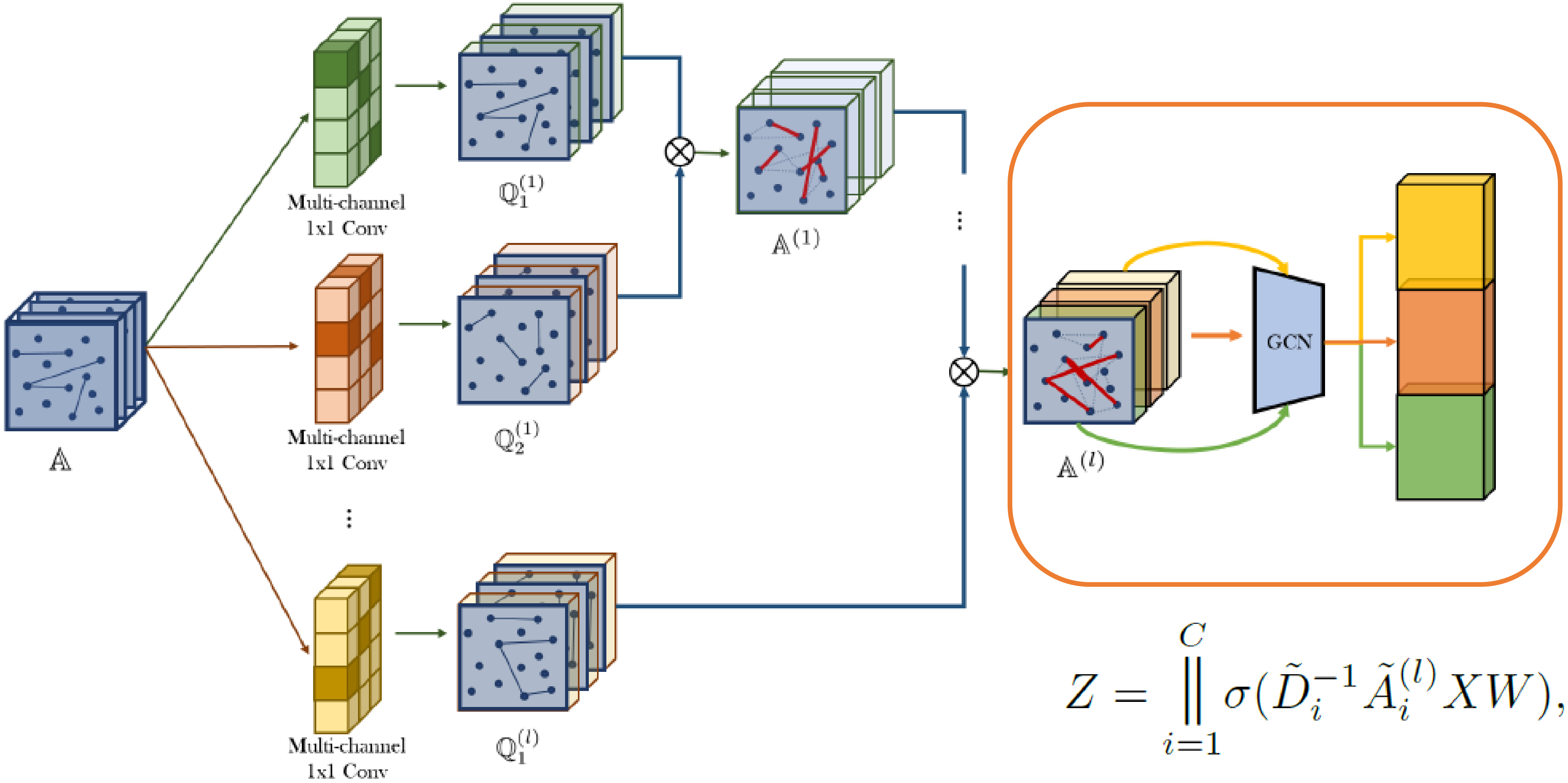
l: hyperparameter

Meta path can be learned via optimization.

Graph Transformer Networks



Graph Transformer Networks



$$Z = \Big\|_{i=1}^{C} \sigma(\tilde{D}_i^{-1} \tilde{A}_i^{(l)} XW),$$

Experimental Results

Settings

* all have 3 types of nodes

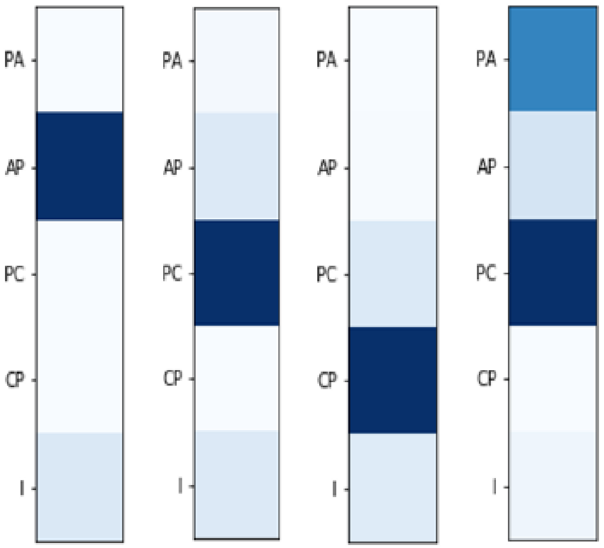
Dataset	# Nodes	# Edges	# Edge type	# Features	# Training	# Validation	# Test
DBLP	18405	67946	4	334	800	400	2857
ACM	8994	25922	4	1902	600	300	2125
IMDB	12772	37288	4	1256	300	300	2339

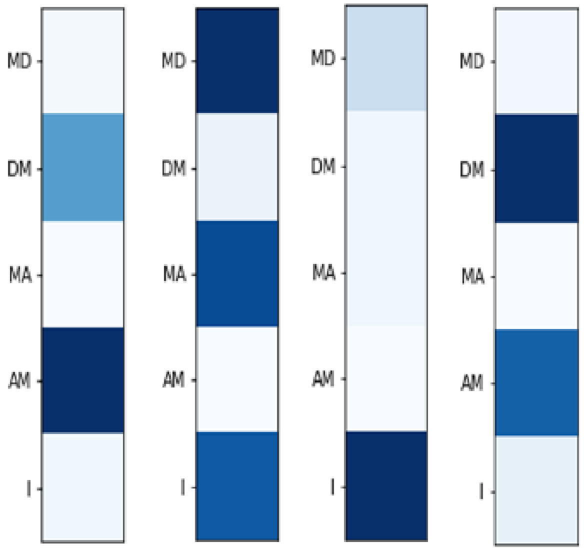
Main Results

	DeepWalk	metapath2vec	GCN	GAT	HAN	GTN_{-I}	**GTN (proposed)**
DBLP	63.18	85.53	87.30	93.71	92.83	93.91	**94.18**
ACM	67.42	87.61	91.60	92.33	90.96	91.13	**92.68**
IMDB	32.08	35.21	56.89	58.14	56.77	52.33	**60.92**

Experimental Results

Dataset	Predefined Meta-path	Meta-path learnt by GTNs Top 3 (between target nodes)	Top 3 (all)
DBLP	APCPA, APA	APCPA, APAPA, APA	CPCPA, APCPA, CP
ACM	PAP, PSP	PAP, PSP	APAP, APA, SPAP
IMDB	MAM, MDM	MDM, MAM, MDMDM	DM, AM, MDM





Discussion

	DeepWalk	metapath2vec	GCN	GAT	HAN	GTN_{-I}	**GTN (proposed)**
DBLP	63.18	85.53	87.30	93.71	92.83	93.91	**94.18**
ACM	67.42	87.61	91.60	92.33	90.96	91.13	**92.68**
IMDB	32.08	35.21	56.89	58.14	56.77	52.33	**60.92**

[ref]

	HAN [36]		GTN [43]			RSHN [45]			HetGNN [44]				MAGNN [12]	
Dataset	ACM		DBLP	ACM	IMDB	AIFB	MUTAG	BGS	MC (10%)		MC (30%)		DBLP	
Metric	Macro-F1	Micro-F1	Macro-F1	Macro-F1	Macro-F1	Accuracy	Accuracy	Accuracy	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
model*	91.89	91.85	94.18	92.68	60.92	97.22	82.35	93.10	97.8	97.9	98.1	98.2	93.13	93.61
GCN*	89.31	89.45	87.30	91.60	56.89	-	-	-	-	-	-	-	88.00	88.51
GAT*	90.55	90.55	93.71	92.33	58.14	91.67	72.06	66.32	96.2	96.3	96.5	96.5	91.05	91.61
model	90.94	90.96	92.95↓	92.28	57.53±2.22↓	97.22	**82.35**	93.10	97.06	97.11	97.34	97.37	92.81	93.36
GCN	**92.25**↑	**92.29**↑	91.48↑	92.28	**59.11±1.73**↑	97.22	79.41	96.55	91.88	92.04	95.37	95.57	88.31	89.37
GAT	92.08↑	92.15↑	**94.18**	**92.49**	58.86±1.73	**100**↑	80.88↑	**100**↑	**98.25**↑	**98.30**↑	**98.42**↑	**98.50**↑	**94.40**↑	**94.78**↑

Mostly, the GNNs on HG is unstable

(GAT > HAN)

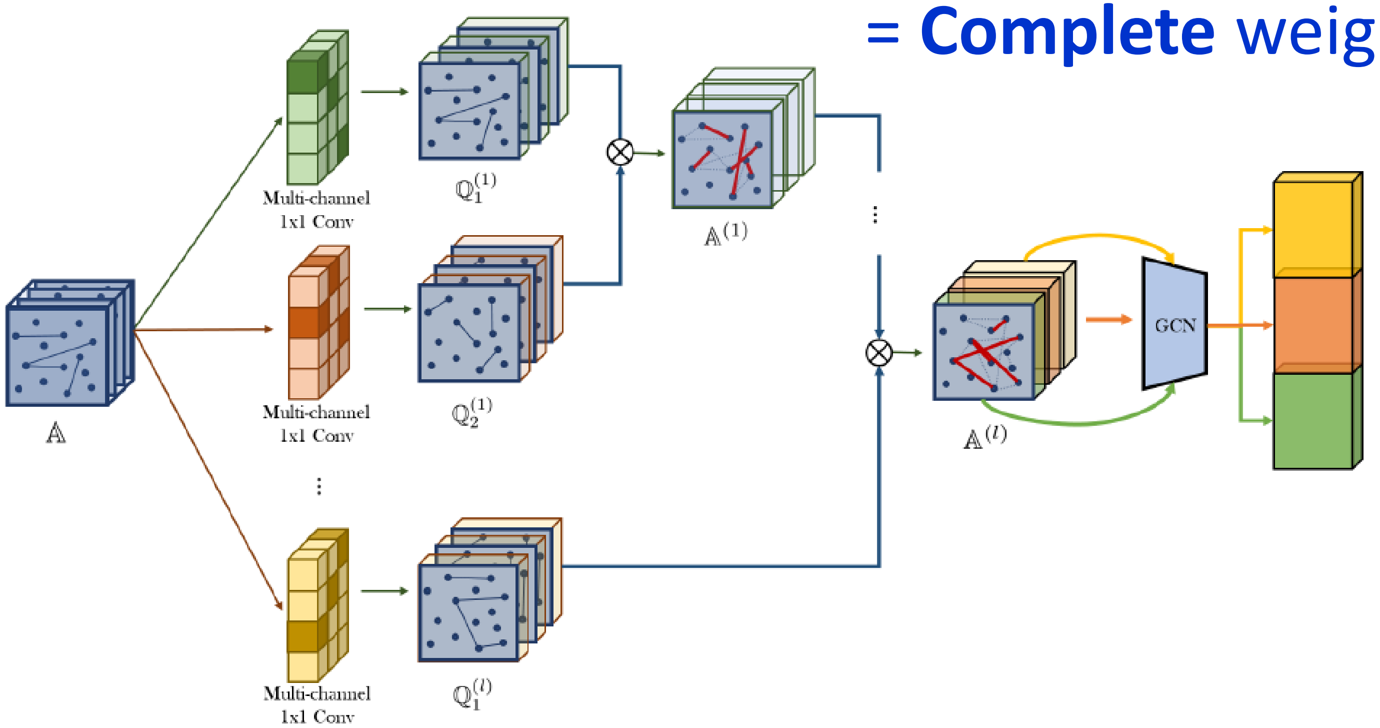
(GAT > GTN)

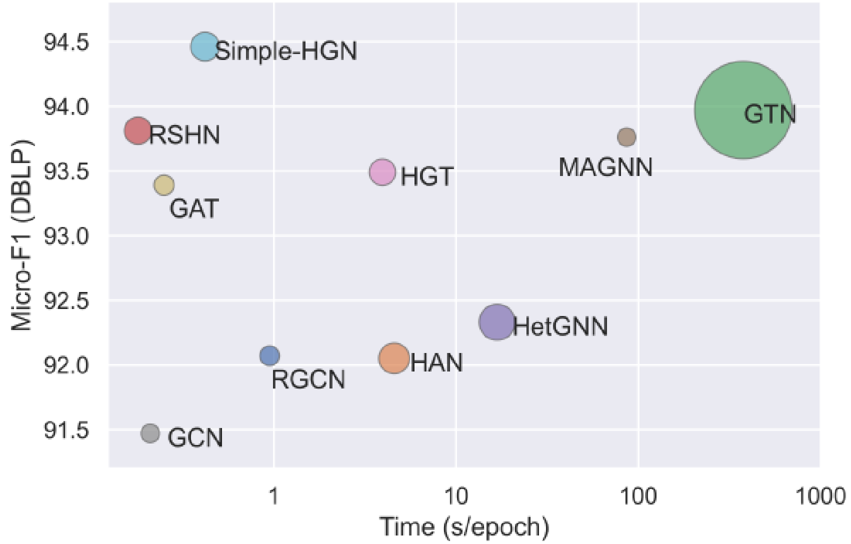
[ref] Lv, Qingsong, et al. "Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks." *KDD 2021*

Discussion

Soft adjacency

= **Complete** weighted graph





Mostly, HG-GNN takes large computational cost

(e.g, GTN takes more than x1000 computational cost)

Lv, Qingsong, et al. "Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks." *KDD 2021*

"Success is not final, failure is not fatal:
it is the courage to continue that counts.“
- Winston Churchill

Thank you!

jindeok6@yonsei.ac.kr