StarGAT: Star-Shaped Hierarchical Graph Attentional Network for Heterogeneous Network Representation Learning

Wen-Zhi Li*†‡, Ling Huang§¶, Chang-Dong Wang*†‡‖, Yu-Xin Ye**
*School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
†Guangdong Province Key Laboratory of Computational Science, Guangzhou, China
‡Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, China
§College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China
¶Guangdong Provincial Key Laboratory of Public Finance and Taxation with Big Data Application, Guangzhou, China
**College of Computer Science and Technology, Jilin University, Changchun, China
‖Corresponding author
liwenzhi199@126.com, huanglinghl@hotmail.com, changdongwang@hotmail.com,yeyx@jlu.edu.cn.

***Abstract*—Many real-world graphs can be viewed as Heterogeneous Networks or Heterogeneous Information Networks (HINs) for that they comprise a diversity of node types and relation types. Due to the efficient representation ability of Graph Neural Network and the idea of random walk, many recent studies apply graph representation learning to HINs and achieve satisfactory results. However, these works either treat different node types in a metapath equally, which is inconsistent with the original graph semantic information for that different node types should have different statuses, or only consider the first-order (node-level) and second-order (metapath-level, *a.k.a.* link-level) information aggregation while ignoring the higher-order relations. To tackle these two problems, we propose a novel *Star-Shaped Hierarchical Graph Attentional Network* (StarGAT) model to boost representation learning in HINs. Specifically, we assume nodes in HINs can be categorized into a star-shaped structure including one center node type and a bunch of auxiliary node types in a specific task; and we encode node-level, link-level and motif-level attentions in a hierarchical manner to capture richer semantic information. Extensive experiments on three datasets illustrate the model effectiveness.**

***Index Terms*—Graph Attentional Network, Network Embedding, Heterogeneous Information Network, Motif**

I. Introduction

Many real-world data are organized as graphs intrinsically like social networks or publication networks. However, network analysis is not an easy task due to the following facts. (1) Since graphs imply complex geometric information and structural manifold, graph data is a typical type of non-Euclidean data, which makes it infeasible to apply the traditional grid-structured deep learning methods. (2) Adjacency matrix is computationally inefficient since it needs $|\mathcal{V}| \times |\mathcal{V}|$ memory ($|\mathcal{V}|$ is the number of nodes) when representing the graph. To tackle these problems, graph representation learning, which is also called network embedding, has drawn great attention recently. The general idea is to represent each node as a low-dimensional vector. Besides, the learned embeddings could simplify downstream tasks.

Many neural network models have been developed to analyze complex networks recently [1]–[4]. Among them, one of the most brilliant approaches is *Graph Convolutional Network* (GCN), which can be divided into two categories, namely spectral-based methods [5]–[7] and spatial-based methods [8]–[10].

The objectives of network embedding are mostly homogeneous networks, which is unpractical for that real-world graphs are often composed of more than one kind of nodes and/or more than one kind of links, which is called *Heterogeneous Information Networks* (HINs) [11]. This shows that apart from the structural information, we also need to take into account the rich semantic information to get the overall embeddings.

For HINs, most methods [12]–[17] handle semantic information using the idea of metapath. However, the predefined metapaths require specific domain knowledge, and assigning equal status to every node in a metapath is against real-world scenarios. Besides, with the fast development of attention mechanism [10], [15], [16], recent studies usually consider node-level and link-level attention, but they ignore the higher-order relations, *i.e. motif* in our paper.

The main contributions of our work are summarized as follows.

- We propose to study a hybrid-order (from node level to motif level) attention mechanism in HINs, which provides a new perspective for further research.
- We propose a novel *Star-Shaped Hierarchical Graph Attentional Network* (StarGAT), a new neural network framework using three-layer hierarchical attention mechanism to boost graph representation learning without predefined metapaths. It can be easily applied to supervised, semi-supervised and unsupervised tasks.
- Extensive experiments on three public real-world datasets in two downstream tasks (node classification and node clustering) show that StarGAT outperforms seven state-of-the-art baselines.

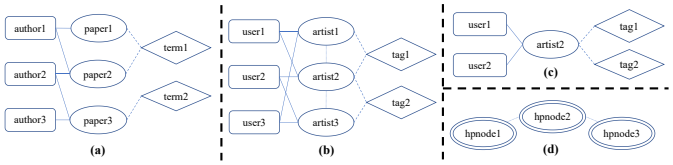


Fig. 1. Some graph schemas used in the paper. (a): An example of k-partite graph. (b): An example of enhanced k-partite graph. (c): An example of star-shaped motif. (d): Graph composed of hypernodes.

II. Related Work

A. Graph Representation Learning

Graph representation learning aims to project nodes in a graph into a low-dimensional feature space while preserving node proximity, so that the low-dimensional vectors can be easily applied to downstream tasks. Existing network embedding algorithms can be categorized into several methodology-based classes, namely matrix eigenvector-based methods, matrix decomposition-based methods, community detection-based methods, neural network-based methods, etc.

When it comes to HINs, the neural network based models can be categorized into two classes, namely shallow neural network and deep neural network. Shallow neural networks usually adopt metapath/metagraph based random walk [12], [13], [18]. Deep neural networks, on other hand, generally use GNN [15], [16], [19], [20], reinforcement learning [21], auto-encoder [22], etc. And many of them apply self-attention to achieve better performance.

B. Higher-Order Structure

In the literature, the most widely studied higher-order structure is motif, which is a subnetwork appearing frequently in the original network [23]. There exist some studies which apply motif to graph analysis. In [24], Lee et al. try to capture higher-order neighborhoods by using weighted multi-hop motif adjacency matrices. In [25], Tsourakakis et al. develop a graph clustering method based on graph motifs for that triangle motif is a better signature than edge. In [10], Sankar et al. propose a motif convolution method which captures semantic higher-order relationships to improve the semi-supervised node classification. Although these works might have achieved state-of-the-art results in specific tasks, there are few methods synthesizing semantic information from node-level to motif-level in a hierarchical manner to boost HIN embedding.

III. Problem Formulation

In this section, several concepts are introduced. We also provide a notation chart (Table I) to summarize the main symbols used in this paper.

Definition 1 (Heterogeneous Information Network (HIN)) *A HIN $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \phi, \varphi\}$ is one kind of graph, where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the link set. It is also associated with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and an edge type mapping function $\varphi : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{A}$ and $\mathcal{R}$ denote the sets of node types and edge types s.t. $|\mathcal{A}| + |\mathcal{R}| > 2$.*

TABLE I
Summary of the main notations.

$\mathbb{R}^n$	n-dim Euclidean space
a,$\mathbf{a}$,$\mathbf{A}$	scalar, vector, matrix
$\mathbf{a}^{\mathrm{T}}$,$\mathbf{A}^{\mathrm{T}}$	vector and matrix transpose
$\lvert\cdot\rvert$	number of elements in the set
$\cdot\|\|\cdot$	concatenation of two arrays
$\mathbf{x}_v$	initial feature vector of node v
$\mathcal{V}$	the set of nodes in the graph
$\mathcal{E}$	the set of edges in the graph
$\mathcal{A}$	the set of node types in the graph
$\mathcal{R}$	the set of link types in the graph
A	one specific node type in $\mathcal{A}$
R	one specific relation type in $\mathcal{R}$
$\mathcal{V}_A$	the set of nodes in type A
A_c^R	the node type linked to the center node c via relation R
$\mathcal{N}_c^R$	the set of neighbor nodes of the center node c via relation R
$\mathcal{V}_c$	the set of center nodes

An example of HIN is illustrated in Fig. 1-(a) with $|\mathcal{A}|$=3 and $|\mathcal{R}| = 2$.

Definition 2 (k-Partite Graph, Enhanced k-Partite Graph) *A k-partite graph is a graph whose nodes can be divided into k different independent sets, and nodes in the same set do not link to each other. An enhanced k-partite graph is a generalized extension of k-partite graph, which allows nodes in one set to link to each other.*

An example of 3-partite graph is Fig. 1-(a) for that author, paper and term nodes are three independent sets.

An example of enhanced 3-partite graph is Fig. 1-(b). The links among artist set make it an enhanced 3-partite graph.

Definition 3 (Motif, Star-shaped Motif, Hypernode) *A motif is a recurrent and statistically significant subgraph of a larger graph. A star-shaped motif is a heterogeneous motif which has one kind of center node linking to a bunch kinds of attribute nodes. All the nodes in a star-shaped motif instance can be abstracted into one hypernode.*

Fig. 1-(c) is an instance of star-shaped motif with a center node artist2 in Fig. 1-(b). The nodes in Fig. 1-(c) can be viewed as a hypernode. Therefore, the enhanced k-partite graph Fig. 1-(b) can be viewed as Fig. 1-(d), which is only made of hypernodes.

IV. The Proposed StarGAT Method

A. Type-Specific Transformation

Due to the fact that different node types share quite different feature spaces [15], [16], [19], we first apply a type-specific node feature transformation operation to transform the raw feature vectors in different feature spaces into the same latent feature space. It is implemented by $|\mathcal{A}|$ transformation matrices. Specifically, for node $v \in \mathcal{V}_A$ of type A, we have

$$\mathbf{e}_v = \mathbf{M}_A \cdot \mathbf{x}_v \tag{1}$$

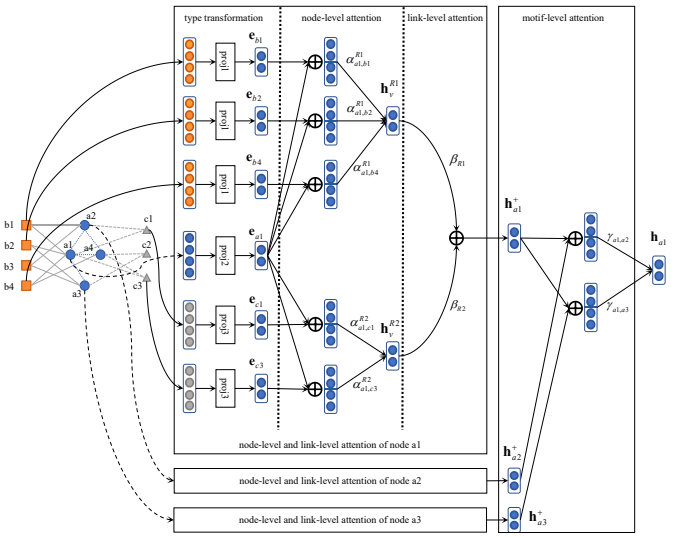


Fig. 2. The overall structure of StarGAT. It includes a type-specific transformation, node-level attention, link-level attention and motif-level attention to form the final embedding of center node $a1$.

where $\mathbf{x}_v \in \mathbb{R}^{d_A}$ is the original feature vector of node v, $\mathbf{e}_v \in \mathbb{R}^d$ is the transformed feature vector in the common latent feature space, $\mathbf{M}_A \in \mathbb{R}^{d \times d_A}$ is the trainable type-specific weight matrix which projects the raw feature into the latent feature space.

B. Node-Level Aggregation

In the node-level aggregation, we aggregate features between a center node and its neighbor nodes linked by one specific link type R. In other words, it is an aggregation via the same link type R within the same star-shaped motif instance, so it can also be construed as intra-link aggregation. Following [8], we adopt a graph attention layer to weight sum the features of all the type A_v^R neighbor nodes of the center node v. The main motivation is that different nodes of the same type might contribute differently to the center node. As in the publication graph, when we want to categorize papers into data mining and computer vision, the terms "social network" and "graph analysis" are more representative than "deep learning" or "neural network". We learn attention coefficients e_{vu}^R for every node u in type A_v^R linked to the center node v, and then weight sum those neighbors according to the coefficients via:

$$\begin{aligned} e_{vu}^R &= LeakyRelu(\mathbf{a}_R^{\mathrm{T}}[\mathbf{e}_v \| \mathbf{e}_u]) \\ \alpha_{vu}^R &= \frac{\exp(e_{vu}^R)}{\sum_{s \in \mathcal{N}_v^R} \exp(e_{vs}^R)} \\ \mathbf{h}_v^R &= \sigma(\sum_{u \in \mathcal{N}_v^R} \alpha_{vu}^R \cdot \mathbf{e}_u) \end{aligned} \tag{2}$$

where $\mathbf{a}^R \in \mathbb{R}^{2d}$ is a trainable attention vector which is the core of graph attention, $\mathcal{N}_v^R$ is the set of neighbor nodes of the center node v via relation R including itself. Following [8], we use *LeakyRelu* as the activation function to get the relation-specific attention coefficients e_{vu}^R for each neighbor node, after which e_{vu}^R is standardized using a *softmax* function thus obtaining α_{vu}^R. After that, we weight summarize the relation-specific neighbor nodes by using α_{vu}^R, and finally an activation function $\sigma(\cdot)$ is applied to get $\mathbf{h}_v^R$ *w.r.t.* relation R.

C. Link-Level Aggregation

Node-level aggregation can only learn one kind of semantic in the original HIN. Therefore, we further conduct a link-level aggregation. In the link-level aggregation, we aggregate $|\mathcal{R}|$ features learned from the above node-level aggregation. In other words, it is an aggregation via different link types R within the same star-shaped motif instance, so it can also be construed as an inter-link aggregation. It is reasonable since different neighbor node types might contribute differently in getting the center node's embedding. For example, the weight of the author nodes should be greater than the term nodes in identifying paper field since many of the terms like "reinforcement learning" and "few-shot learning" are non-discriminative *w.r.t.* paper fields intuitively. Here, we also adopt the attention mechanism to assign different weights to different link types. Following [15], the self-attention is formulized as:

$$\begin{aligned} e^R &= \mathbf{q}_R^{\mathrm{T}}(\frac{1}{|\mathcal{V}_c|} \sum_{v \in \mathcal{V}_c} \tanh[\mathbf{W}_R \mathbf{h}_v^R + \mathbf{b}_R]) \\ \beta_R &= \frac{\exp(e^R)}{\sum_{i \in \mathcal{R}} \exp(e^i)} \\ \mathbf{h}_v^+ &= \sum_{R \in \mathcal{R}} \beta_R \cdot \mathbf{h}_v^R \end{aligned} \tag{3}$$

where $\mathcal{V}_c$ is the set of center nodes of star-shaped motif instances. $\mathbf{W}_R \in \mathbb{R}^{d \times d}$ is a trainable weight matrix which will be applied to every $\mathbf{h}_v^R$. $\mathbf{b}_R \in \mathbb{R}^d$ is a type-specific bias, which is also a trainable parameter. $\mathbf{q}_R \in \mathbb{R}^d$ is the trainable attention vector *w.r.t.* relation type. e^R is standardized using a *softmax* function and we can get β^R. Once we get β^R, we weight summarize type-specific embeddings $\mathbf{h}_v^R$ learned from the node-level aggregation and finally get embedding $\mathbf{h}_v^+$. Note that, $\mathbf{h}_v^+$ has aggregated intra-link and inter-link information, *i.e.*, auxiliary information from all the attribute nodes, which makes $\mathbf{h}_v^+$ a virtual representation of a hypernode.

D. Motif-Level Aggregation

So far, for enhanced k-partite graph, the original HIN has already been converted to a new graph containing only hypernodes *w.r.t.* center nodes. For k-partite graph, however, it is only a bunch of isolated nodes. Here, we only consider θ most frequent indirect neighbor nodes via any attribute nodes for k-partite graph and make it a new graph.

Anyway, we have obtained a new homogeneous graph with hypernodes. Now we apply another attention mechanism, which is the higher-order self-attention, on this new graph. Note that there are two versions of attention mechanism corresponding to the enhanced k-partite graph and the k-partite graph respectively. This is because nodes in the enhanced k-partite graphs have a random number of neighbors, while

nodes in k-partite graphs have a fixed number of neighbors (θ). The two versions of attention mechanism are as follows:

$$\mathbf{h}_i = \sigma(\sum_{j\in\mathcal{N}_i} \frac{\exp(LeakyRelu(\mathbf{p}^{\mathrm{T}}[\mathbf{W}\mathbf{h}_i^+ \| \mathbf{W}\mathbf{h}_j^+]))}{\sum_{k\in\mathcal{N}_i}\exp(LeakyRelu(\mathbf{p}^{\mathrm{T}}[\mathbf{W}\mathbf{h}_i^+ \| \mathbf{W}\mathbf{h}_k^+]))}\mathbf{W}\mathbf{h}_j^+) \tag{4}$$

and

$$\mathbf{h}_i = \sum_{j\in\mathcal{N}_i} \frac{\exp(\mathbf{p}^{\mathrm{T}}\tanh[\mathbf{W}\mathbf{h}_j^+ + \mathbf{b}])}{\sum_{k\in\mathcal{N}_i}\exp(\mathbf{p}^{\mathrm{T}}\tanh[\mathbf{W}\mathbf{h}_k^+ + \mathbf{b}])}\mathbf{h}_j^+ \tag{5}$$

where $\mathcal{N}_i$ is the set of neighbor hypernodes of the target hypernode including itself. The other notations are similar to those in Eq. (2) or Eq. (3).

Algorithm 1 model training for StarGAT

Input: $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \phi, \varphi\}$, raw features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$
Output: learned node embeddings $\{\mathbf{h}_v, \forall v \in \mathcal{V}_c\}$

```
for node type A ∈ 𝒜 do
    node content transformation via Eq. (1)
end for

while not converge do
    for v ∈ 𝒱_c do
        for R ∈ ℛ do
            calculate h_v^R via Eq. (2)
        end for
        calculate h_v^+ using h_v^R via Eq. (3)
    end for
    for node v ∈ 𝒱_c do
        sample neighbor nodes of v
        claculate h_v via Eq. (4) or Eq. (5)
    end for
    backpropagation via Eq. (6) or Eq. (7)
end while
```

E. Training and Optimization

StarGAT is capable of both semi-supervised and unsupervised learning paradigms. For semi-supervised learning, we use cross entropy loss as the loss function:

$$L = -\sum_{v\in\mathcal{V}_L}\sum_{c=1}^{|\mathcal{C}|} y_{vc}\cdot\log\mathbf{h}_{vc} \tag{6}$$

where $\mathcal{V}_L$ is the set of labeled nodes, $\mathcal{C}$ is the set of all label classes, y_{vc} is the ground-truth label of v, and $\mathbf{h}_{vc}$ is the predicted logit of v.

For unsupervised learning, we use negative sampling:

$$L = -\sum \log[\sigma(\mathbf{h}_i^{\mathrm{T}}\cdot\mathbf{h}_j) - \sigma(\mathbf{h}_i'^{\mathrm{T}}\cdot\mathbf{h}_j')] \tag{7}$$

where σ is the *sigmoid* activate function, $\mathbf{h}_i$ and $\mathbf{h}_j$ are embeddings of linked hypernode pair i and j, while $\mathbf{h}_i'$ and $\mathbf{h}_j'$ are embeddings of unlinked hypernode pair as negative samples.

TABLE II
DATASET STATISTICS.

Datasets	Number of Nodes	Number of Edges	Metapaths
ACM	# paper (P): 4,025 # author (A): 7,167 # term (T): 1,902	# P-A: 37,055 # P-T: 972,973	P-A-P P-T-P A-P-T-P-A
DBLP	# paper (P): 14,328 # author (A): 4,057 # term (T): 7,723	# P-A: 19,645 # P-T: 85,810	P-A-P P-T-P A-P-T-P-A
Yelp	# business (B): 2,614 # user (U): 1,286 # service (S): 2 # star level (SL): 9 # reservation (R): 2	# B-U: 30,838 # B-S: 2,614 # B-SL: 2,614 # B-R: 2614	B-U-B B-S-B B-SL-B B-R-B U-B-S-B-U

V. EXPERIMENTS

A. Experimental Settings

1) Datasets: We adopt three HIN datasets. **ACM** is a computer science publisher. It is a 3-partite graph containing paper-author, paper-term links with labels of paper fields, and the center node type is paper. **DBLP** is also a 3-partite graph sharing the same structure with ACM. **Yelp** is an online business transaction platform. It is a 5-partite graph containing business-user, business-service, business-star-level, business-reservation links with labels of business types, and the center node type is business.

2) Baselines: We compare StarGAT against seven algorithms.

- **Deepwalk** [1] is a homogeneous random walk-based model which adopts skip-gram on node sequences generated by random walk.
- **Metapath2vec** [12] is a heterogeneous random walk-based model that uses metapath-based random walk and skip-gram to get HIN embedding.
- **Node2vec** [2] is a homogeneous random walk-based model which extends Deepwalk by taking both DFS and BFS into consideration when sampling neighbor nodes.
- **GCN** [7] is a homogenous GNN that utilizes graph convolutions on the first-order neighbors as layer-wise aggregation in spectral domain.
- **GAT** [8] is a homogenous GNN that performs graph convolutions in the spatial domain with incorporating the first-order neighbor nodes.
- **HAN** [15] is a heterogeneous GNN which learns HIN embedding via node-level and semantic-level attentions in metapath-specific homogeneous graphs.
- **MAGNN** [16] is a heterogeneous GNN which uses intra-metapath aggregation and inter-metapath aggregation to learn HIN embedding.

3) Configurations: For all the three datasets, we split the center nodes into training, validation and test sets of the ratios 10%, 10% and 80% respectively.

For random-walk based methods, we set walk length to 100, walk per node to 10, and window size to 5. For GNN models, we optimize them using Adam with learning rate 0.005 and weight decay 0.001. We set the dropout rate to 0.5. For a fair comparison, we set the number of attention head to 8,

TABLE III
NODE CLASSIFICATION RESULTS (%) ON THE THREE REAL-WORLD DATASETS. **BOLD**: BEST; UNDERLINE: RUNNER-UP.

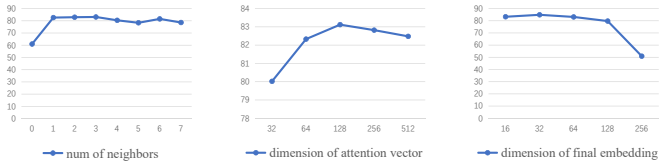
Datasets	Metrics	Deepwalk	Metapath2vec	Node2vec	GCN	GAT	HAN	MAGNN	StarGAT
ACM	Macro-F1	59.44	51.49	73.31	24.85	51.25	65.52	70.40	**82.60**
	Micro-F1	61.91	67.36	74.53	50.17	62.78	69.18	72.03	**83.12**
DBLP	Macro-F1	21.60	58.76	68.20	64.60	66.50	74.49	75.57	**75.65**
	Micro-F1	36.17	66.55	71.73	68.83	70.13	78.53	78.90	**79.15**
Yelp	Macro-F1	48.23	45.48	57.29	54.98	67.52	56.37	71.82	**71.94**
	Micro-F1	65.04	61.38	68.78	72.82	76.23	74.18	77.66	**78.23**

TABLE IV
NODE CLUSTERING RESULTS (%) ON THE THREE REAL-WORLD DATASETS. **BOLD**: BEST; UNDERLINE: RUNNER-UP.

Datasets	Metrics	Deepwalk	Metapath2vec	Node2vec	GCN	GAT	HAN	MAGNN	StarGAT
ACM	NMI	2.70	15.02	25.55	4.67	9.98	29.80	29.19	**42.57**
	ARI	2.40	14.20	24.10	1.84	8.74	19.20	31.42	**40.90**
DBLP	NMI	0.11	22.82	2.51	16.34	25.55	40.56	36.21	**40.66**
	ARI	-0.06	15.10	0.84	17.52	29.71	45.71	39.34	**47.83**
Yelp	NMI	25.17	0.94	10.06	36.23	34.83	**40.97**	35.93	40.46
	ARI	22.43	0.74	11.65	34.64	33.52	**44.15**	31.01	38.79

TABLE V
ABLATION STUDY RESULTS (%) ON THE THREE REAL-WORLD DATASETS.

Datasets	Metrics	$init.$	$vr1$	$vr2$	$vr3$
DBLP	Macro-F1	75.65	74.86	73.69	75.41
	Micro-F1	79.15	78.44	77.48	78.87
ACM	Macro-F1	82.60	82.20	82.15	47.50
	Micro-F1	83.12	83.02	82.96	60.99
Yelp	Macro-F1	71.94	54.65	64.09	71.37
	Micro-F1	78.23	54.60	75.68	77.52



(a) (b) (c)

Fig. 3. Hyper-parameter analysis of StarGAT.

if applicable. The dimensions of attention vectors in StarGAT, HAN and MAGNN are set to 128, the θs are set to 3. We train every model until they converge with the training set, and then get the test set embeddings using the trained model. Early-stopping is used in StarGAT, HAN and MAGNN with the validation set. We set the final output embedding dimensions to 64 for all the algorithms. For metapath-depended baselines, we test all the metapaths in Table II and report the best performance for every dataset.

B. Node Classification and Node Clustering

We conduct the node classification and node clustering tasks on the three datasets. After training the models using the training set, we feed the test set nodes to the models and get their embeddings. Then we partite these embeddings together with their ground-truth labels into a training set and a test set via 80%-20% proportion and train a *SVM* classifier. The results in terms of *Macro-F1* and *Micro-F1* are reported in Table III. For node clustering, similarly, we train a *k-means* model using the test set node embeddings, where k is set to the number of classes. The results in terms of *normalized mutual information* (NMI) and *adjusted rand index* (ARI) are reported in Table IV.

According to both tables, we can see that StarGAT performs the best in almost every metrics, especially the ACM dataset, where StarGAT outperforms the best baseline by achieving around 10% improvement. Generally speaking, GNN-based HIN embedding methods perform the best due to their ability to exploit the rich semantic information besides graph structural information. GNN-based homogeneous network embedding methods perform the second best, which is mainly because indirect link relations are also sufficient to accomplish the center node classification and clustering tasks.

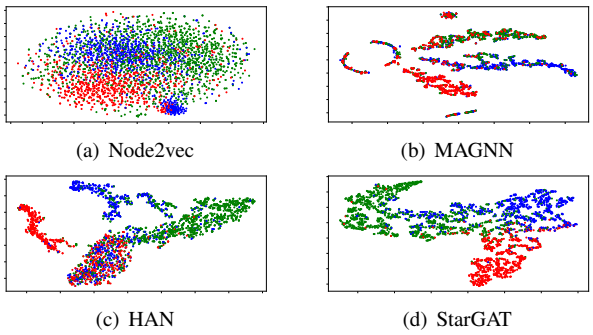
C. Ablation Study

To explore how the three self-attentions affect this model, we conduct ablation study by disabling one of the three self-attentions. For simplicity, we denote $init.$ for the initial StarGAT, $vr1$ for the StarGAT variant with no node-level attention, $vr2$ for the StarGAT variant with no link-level attention and $vr3$ for the StarGAT variant with no motif-level attention. The results are shown in Table V.

As can be seen, on the ACM dataset, the motif-level attention is important in getting better embeddings. For the Yelp dataset, the node-level attention and the link-level attention are more important, perhaps because more attribute node types around the center node make it necessary to distinguish the degree of importance between them. For clean 3-partite graph DBLP, the importance of the three attentions is not trivial.

D. Hyper-Parameter Analysis

We conduct the parameter analysis experiment and report the results of *Micro-F1* on the ACM dataset.



(a) Node2vec (b) MAGNN (c) HAN (d) StarGAT

Fig. 4. Visualization of learned embeddings on ACM. Each point indicates a paper and its color indicates research area.

1) Number of neighbors on the k-partite graph datasets: We test θ most frequent neighbors via all kinds of indirect links for k-partite graphs. As we can get from Fig. 3-(a), setting θ to 0 is a bad idea, and the performance will take a huge leap even only considering one neighbor.

2) Dimension of attention vector: It only influences the model slightly, according to Fig. 3-(b). The performance achieves the best result when it is 128.

3) Dimension of the final embedding: According to Fig. 3-(c), the result is steady when the dimensions are among 16 to 128, but it drops drastically when the dimension is 256, which is also due to the overfitting issue.

E. Visualization

We also visualize the results to achieve an intuitive understanding. We visualize StarGAT, MAGNN, HAN, and Node2vec on the ACM dataset via t-SNE [26] in Fig. 4.

According to the figure, we can see that StarGAT performs the best, for that nodes in different types are relatively far.

VI. Conclusion

In this work, we propose a novel end-to-end star-shaped hierarchical graph attentional network (StarGAT) for heterogeneous information network embedding, which contains node-level, link-level and motif-level attentions. StarGAT assigns different statuses to different node types and forms star-shaped hierarchical HIN. By aggregating neighbor nodes from lower-order to higher-order, StarGAT can exploit richer local structure and neighbor semantic information without predefined metapaths. Experiments on three real-world datasets in the node classification and node clustering tasks show StarGAT can achieve state-of-the-art performance. In future work, we would like to explore more pattern-fixed network schemas like signed network or multi-relational network etc.

Acknowledgment

This work was supported by NSFC (62106079, 61876193, 42050103, 62076108 and U19A2061), Natural Science Foundation of Guangdong Province (2020A1515110337), Guangdong Province Key Laboratory of Computational Science at the Sun Yat-sen University (2020B1212060032), Open Foundation of Guangdong Provincial Key Laboratory of Public Finance and Taxation with Big Data Application, and Guangzhou Science and Technology Plan Project (202102080491).

References

[1] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *KDD*, 2014, pp. 701–710.
[2] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*, 2016, pp. 855–864.
[3] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *KDD*, 2016, pp. 1225–1234.
[4] C.-D. Wang, W. Shi, L. Huang, K.-Y. Lin, D. Huang, and P. S. Yu., "Node-pair information preserving network embedding based on adversarial networks," *IEEE Trans. Cybern.*, 2020.
[5] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *ICLR*, 2014.
[6] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NIPS*, 2016, pp. 3837–3845.
[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[8] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.
[9] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017, pp. 1024–1034.
[10] A. Sankar, X. Zhang, and K. C. Chang, "Meta-GNN: metagraph neural network for semi-supervised learning in attributed heterogeneous information networks," in *ASONAM*, 2019, pp. 137–144.
[11] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 17–37, 2016.
[12] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *KDD*, 2017, pp. 135–144.
[13] T. Fu, W. Lee, and Z. Lei, "HIN2Vec: Explore meta-paths in heterogeneous information networks for representation learning," in *CIKM*, 2017, pp. 1797–1806.
[14] R. Hussein, D. Yang, and P. Cudré-Mauroux, "Are meta-paths necessary? revisiting heterogeneous graph embeddings," in *CIKM*, 2018, pp. 437–446.
[15] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *WWW*, 2019, pp. 2022–2032.
[16] X. Fu, J. Zhang, Z. Meng, and I. King, "MAGNN: metapath aggregated graph neural network for heterogeneous graph embedding," in *WWW*, 2020, pp. 2331–2341.
[17] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, "Heterogeneous information network embedding for recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 357–370, 2018.
[18] Y. He, Y. Song, J. Li, C. Ji, J. Peng, and H. Peng, "HeteSpaceyWalk: A heterogeneous spacey random walk for heterogeneous information network embedding," in *CIKM*, 2019, pp. 639–648.
[19] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *KDD*, 2019, pp. 793–803.
[20] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *WWW*, 2020, pp. 2704–2710.
[21] G. Wan, B. Du, S. Pan, and G. Haffari, "Reinforcement learning based meta-path discovery in large-scale heterogeneous information networks," in *AAAI*, 2020, pp. 6094–6101.
[22] H. Wang, F. Zhang, M. Hou, X. Xie, M. Guo, and Q. Liu, "SHINE: Signed heterogeneous information network embedding for sentiment link prediction," in *WSDM*, 2018, pp. 592–600.
[23] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.
[24] J. B. Lee, R. A. Rossi, X. Kong, S. Kim, E. Koh, and A. Rao, "Graph convolutional networks with motif-based attention," in *CIKM*, 2019, pp. 499–508.
[25] C. E. Tsourakakis, J. Pachocki, and M. Mitzenmacher, "Scalable motif-aware graph clustering," in *WWW*, 2017, pp. 1451–1460.
[26] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, 2008.