Detecting Communities from Heterogeneous Graphs: A Context Path-based Graph Neural Network Model

Linhao Luo
Harbin Institute of Technology, Shenzhen
China
luolinhao@stu.hit.edu.cn

Yixiang Fang*
The Chinese University of Hong Kong, Shenzhen
China
fangyixiang@cuhk.edu.cn

Xin Cao
The University of New South Wales
Australia
xin.cao@unsw.edu.au

Xiaofeng Zhang*
Harbin Institute of Technology, Shenzhen
China
zhangxiaofeng@hit.edu.cn

Wenjie Zhang
The University of New South Wales
Australia
wenjie.zhang@unsw.edu.au

ABSTRACT

Community detection, aiming to group the graph nodes into clusters with dense inner-connection, is a fundamental graph mining task. Recently, it has been studied on the heterogeneous graph, which contains multiple types of nodes and edges, posing great challenges for modeling the high-order relationship between nodes. With the surge of graph embedding mechanism, it has also been adopted to community detection. A remarkable group of works use the meta-path to capture the high-order relationship between nodes and embed them into nodes' embedding to facilitate community detection. However, defining meaningful meta-paths requires much domain knowledge, which largely limits their applications, especially on schema-rich heterogeneous graphs like knowledge graphs. To alleviate this issue, in this paper, we propose to exploit the context path to capture the high-order relationship between nodes, and build a Context Path-based Graph Neural Network (CP-GNN) model. It recursively embeds the high-order relationship between nodes into the node embedding with attention mechanisms to discriminate the importance of different relationships. By maximizing the expectation of the co-occurrence of nodes connected by context paths, the model can learn the nodes' embeddings that both well preserve the high-order relationship between nodes and are helpful for community detection. Extensive experimental results on four real-world datasets show that CP-GNN outperforms the state-of-the-art community detection methods [1].

CCS CONCEPTS

• **Information systems → Data mining**.

* Corresponding authors
[1]Code and data are available at: https://github.com/RManLuo/CP-GNN

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8446-9/21/11...$15.00
https://doi.org/10.1145/3459637.3482250

KEYWORDS

Community Detection, Heterogeneous Graphs, Context Path, Graph Neural Network, Unsupervised Learning

ACM Reference Format:
Linhao Luo, Yixiang Fang, Xin Cao, Xiaofeng Zhang, and Wenjie Zhang. 2021. Detecting Communities from Heterogeneous Graphs: A Context Path-based Graph Neural Network Model. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3459637.3482250

1 INTRODUCTION

As a fundamental topic in network science, community detection, aiming to group the graph nodes into clusters with dense inner-connection, has been studied for decades and found various real-world applications, such as recommendation (e.g., [25, 36]), anomaly detection (e.g., [46]), and scientific discipline discovery (e.g., [60]). Most existing works of community detection (e.g., [7, 11, 45]) mainly focus on detecting communities from homogeneous network that contains the same type of nodes. These solutions, however, may not work well on many real-world graphs that are with multiple node types and edge types, which are also called *heterogeneous graphs*, and they are prevalent in many real-world applications, including bibliographic networks, social media, and knowledge graphs. For example, Figure 1 depicts a bibliographic network with four types of nodes, i.e., *paper*, *author*, *venue*, and *topic*, and four relations (edge types) among them. The existing works of community detection on heterogeneous graphs can generally be classified into two groups; the first group [42] focuses on detecting clusters, each of which contains objects with multiple types, and the second group [2] aims to generate clusters of nodes with a specific type. In this paper, we follow the second group and aim to cluster nodes such that nodes in the same cluster have strong relationships.

Detecting communities from heterogeneous graphs is more challenging than that on homogeneous graphs, since the multiple types of nodes and edges carry more abundant semantic information. In Figure 1, for example, the authors a_1 and a_5 should belong to the same community (Community 1), but they are not directly connected in the graph, making them hard to be grouped into the same cluster

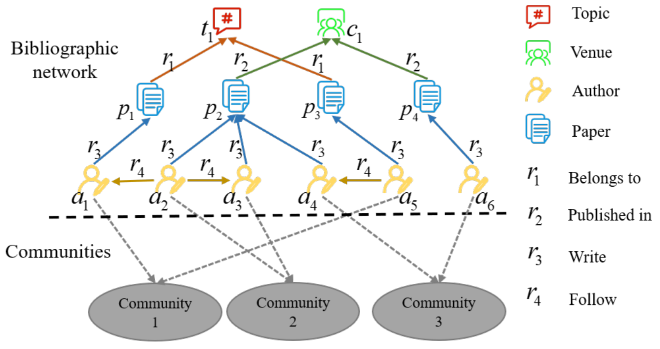


Figure 1: Detecting communities from a bibliographic graph.

if only direct relations (e.g., "follow") are considered). However, a_1 and a_5 have written papers p_1 and p_3 respectively, which share the same topic t_1. As a result, they can be grouped into the same cluster if we consider their high-order relationship, or the relationship that cannot be captured by directed links.

To capture the high-order relationship above, several efforts have been made (e.g., [12, 41]), but most of them rely on some pre-defined meta-paths [40], which reveals the latent high-order relationships. For example, the path *author-paper-topic-paper-author* can model the relationship we showed above. To further capture and represent the high-order relations, a few works [9, 59] integrate the meta-path oriented graph embedding mechanism with community detection. However, the problem of these methods is that their performance highly depends on the quality of the pre-defined meta-paths, which need to be selected by domain experts. Moreover, the number of meta-paths increases exponentially with the path length, meaning that it is almost infeasible to find all meaningful meta-paths to capture the high-order relationships. Furthermore, different meta-paths contribute differently to the community detection, which imposes great challenges for distinguishing their importance.

In this paper, we propose a novel model, called the Context Path-based Graph Neural Network (CP-GNN), for detecting communities with nodes of the same target type in the heterogeneous graph. Here, the target type is also called the *primary type*, while the other types of nodes are called auxiliary types. In this model, we adopt the concept of "context path" [1], which links two primary type nodes via a sequence of edges with some auxiliary type nodes. It can not only well capture the high-order relationship, but also avoid requiring customized meta-paths selected by domain experts. We first introduce the *context path probability* that is the probability that two nodes are connected by a context path. Then, we propose a novel objective function for learning node embeddings unsupervisedly, by maximizing the expectation of the co-occurrence of context neighbors (nodes connected by context paths), which exploits both the structure and the high-order context relationships among nodes.

To learn the embeddings, instead of exhaustedly enumerating all the context paths, we employ the graph neural network model to recursively embed the context path information between nodes into the node embeddings. We further propose the length-wise and relation-wise attention mechanisms to discriminate the importance of different context paths that preserve different high-order relationships. Thus, our model not only avoids customizing the meta-paths, but also well captures the high-order relationships between nodes that are preserved by the context paths with different importance. Finally, the learned embeddings from the neural network are directly used for community detection.

In summary, our principal contributions are as follows:

- We adopt the context path to capture the high-order relationship information and introduce the context path probability to model the learning objective function.
- We propose a novel neural network model CP-GNN, which can capture the rich high-order relationship for learning the node embeddings unsupervisedly.
- We conduct extensive experiments on four real datasets, which demonstrate the superior performance of our CP-GNN model over the state-of-the-art methods. And the visualization experiments show the CP-GNN can capture high-order relationships with different importance.

2 RELATED WORK

In this section, we review three representative groups of existing works on the topic of network community detection.

• **Conventional community detection.** Community detection has attracted a lot of research attention [6, 13, 14, 28]. The early works often exploit the local link structure to group the the vertices into different clusters [35, 45]. More related works can be found in these survey papers [11, 26].

However, most of these methods focus on homogeneous graphs. Recently, some works have studied the community detection task on heterogeneous graph [27, 37]. Reference [3] proposes a method that can learn an optimal linear combination of the relations in heterogeneous graph. Then by adopting MinCut-based and Regression-based algorithms, it can achieve a better performance on community detection. Reference [30] models the structure and content of heterogeneous graph with outlier links. HeProjI [38] projects a heterogeneous graph into a sequence of sub-networks and conducts community detection. TCSC [2] considers both the graph connection and vertex attributes to detect clusters. AGGMMR [58] proposes a framework to perform community detection utilizing both the attributes and topological information through a greedy modularity maximization model. Reference [12, 41] adopt the meta-path to capture the high-order relationships between nodes for detecting communities in heterogeneous graphs. Nevertheless, as aforementioned, the meta-paths need to be carefully selected by domain experts, which imposes a great limitation for their applications.

• **Graph embedding for community detection.** Recently, with the surge of graph embedding methods [9, 17], many researchers focus on addressing the community detection problem with the help of graph embedding [43, 50, 53]. Cavallari et al [4] intergates the node embedding and clustering together to conduct community detection by optimizing the first-order and second-order neighbors' loss, high-order loss, and clustering loss. CDE [23] proposes a novel embedding based method. It embeds the inherent community structures into structure embeddings via known community memberships. Then based on the node attributes and community structures embeddings, it formulates the community detection as a matrix factorization optimization problem. NEC [39] proposes an algorithm to learn graph embedding for community detection in heterogeneous graphs, which learns graph structure-based representations and clustering-oriented representations together. Then it adopts the K-means to perform the community detection.

• **GNN-based community detection.** Many deep learning-based community detection methods are also developed [19]. As one of the most widely used deep learning techniques, the Graph Neural Network (GNN) [22] has also shown great power in community detection [7, 56]. LGNN [7] is a graph neural network model which exploits edges' adjacency information of the graph for community detection. MRFasGCN [56] proposes a Markov random field enhanced GNN to group nodes into different communities. However, most of them do not consider complex relationships, which leads them inadequate to fuse enough relationships for heterogeneous graph community detection. Recently, HTGCN [59] shows a temporal graph neural network to perform the community detection on temporal heterogeneous graph. It considers both the temporal and heterogeneous information of the graph to increase the performance. Despite the existing success, most GNN-based approaches [7, 48] regard the community detection as a supervised node classification task, which predicts the target community for each node. However, the ground truths of the community are not always available, making them inapplicable in this case. Thus, it is desirable to develop fully unsupervised GNN-based community detection methods.

3 PRELIMINARIES

Definition 3.1 (**Heterogeneous graph [40]**). The heterogeneous graph is defined as a graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$ with a node mapping function $\phi(v) : \mathcal{V} \rightarrow \mathcal{A}$ and an edge mapping function $\psi(e) : \mathcal{E} \rightarrow \mathcal{R}$, where $|\mathcal{A}| + |\mathcal{R}| > 2$, each node $v \in \mathcal{V}$ belongs to a node type $\phi(v) \in \mathcal{A}$, and each edge $e \in \mathcal{E}$ belongs to an edge type (also called relation) $\psi(e) \in \mathcal{R}$.

Definition 3.2 (**Primary type and auxiliary type**). As aforementioned, for the purpose of community detection, usually only one node type is targeted by the task, and we call it the *primary type*, denoted by P. The nodes with type P are called primary nodes. The other node types are called auxiliary types, which constitute a set of types $\mathcal{A}'$. Note that technically, any node type can be regarded as the primary type.

Definition 3.3 (**Primary graph and auxiliary graph**). Given a heterogeneous graph G, the primary graph is a subgraph of G, denoted by G_P=($\mathcal{V}_P, \mathcal{E}_P$) where each node $v \in \mathcal{V}_P$ is of the primary node type P and each edge $e \in \mathcal{E}_P \subseteq \{\mathcal{V}_P \times \mathcal{V}_P\}$. Similarly, the auxiliary graph is also a subgraph of G with nodes of a specific type A, denoted by $G_A = (\mathcal{V}_A, \mathcal{E}_A)$, where for each $v \in \mathcal{V}_A$, $\phi(v) = A \in \mathcal{A}'$ and each edge $e \in \mathcal{E}_A \subseteq \{\mathcal{V}_A \times \mathcal{V}_A\}$.

Example 1. As the heterogeneous graph shown in Figure 1, the "Author" can be defined as the primary type, and the remaining node types are treated as auxiliary types. Similarly, the "Paper" can also be chosen as the primary type if necessary. Meanwhile, the "Author", "Topic", and "Venue" nodes and their relations will form the auxiliary graphs.

Definition 3.4 (**Context path and context neighbors [1]**). Given a heterogeneous graph G, a *context path* is a path connecting two nodes v_i and v_j in the primary graph G_P, denoted by $\rho^K = \langle v_i, R^K, v_j \rangle$, where R^K is any path connecting v_i and v_j that contains only K ($K \geq 0$) nodes in auxiliary graphs, and it is also called the *context edge sequence*. The length of a context path is K (when K=0, R^K=$\emptyset$). We say two nodes are *context neighbors* if they are connected by a context-path.

Example 2. Figure 2 depicts four possible context paths ρ^* with different lengths that can connect authors a_1 and a_2, where R^* denotes the auxiliary nodes that constitute the path.

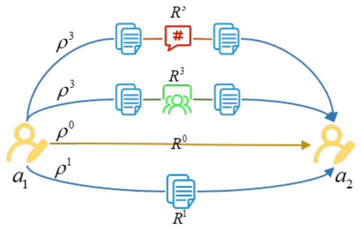


Figure 2: Context paths between a_1 and a_2.

Difference between context path and meta-path. Intuitively, the different high-order semantic relationships revealed from the context path come from the different auxiliary nodes. The purpose of the meta-path is to manually define the combination of the auxiliary nodes in the context path. However, the number of the combinations explodes exponentially when the node types and order size increase. Thus, the context path relaxes the restriction of the auxiliary nodes. Given a length K, the context path contains all the possible K-order relationships. For example, in Figure 2, two meta-paths *Author-Paper-Topic-Paper-Author* and *Author-Paper-Venue-Paper-Author* can both be represented by a 3-length context path, and we propose the CP-GNN to futher differentiate them.

Besides, specifying an integer K is much easier than defining the meta-path, because the number of meta-paths of different node/edge types grows exponentially as the meta-path length increases, while the choices of K are rather limited since the average length of the shortest path between two nodes in real-world networks is between 4 to 6, according to [54], thus the K can be determinated empirically or with the help of the proposed context path length attention mechanism.

Problem definition. Given a heterogeneous graph G, our goal is to learn a good primary node embedding $Z_P \in \mathbb{R}^{N \times d}$ where N denotes the number of primary type nodes and d is the embedding dimension, such that they can be used to group the nodes into a set of communities $C = \{1, \cdots, C\}$ with strong inner-connection.

4 OUR CP-GNN APPROACH

In this section, we present the Context Path-based Graph Neural Network (CP-GNN) model for learning node representations that well preserve the high-order relationship between nodes, and the overall framework is depicted in Figure 3. Given a heterogeneous graph (composed of a primary graph and auxiliary graphs), for each node in the primary graph G_P, we first extract all its context neighbors by using the context paths whose lengths range from 0 to K. Then, we learn the representations of nodes in G_P by training the CP-GNN model, in which the objective function is to maximize the probability of having the context neighbors for each node in G_P. In the following, we first introduce the objective function and then discuss the details of the CP-GNN model.

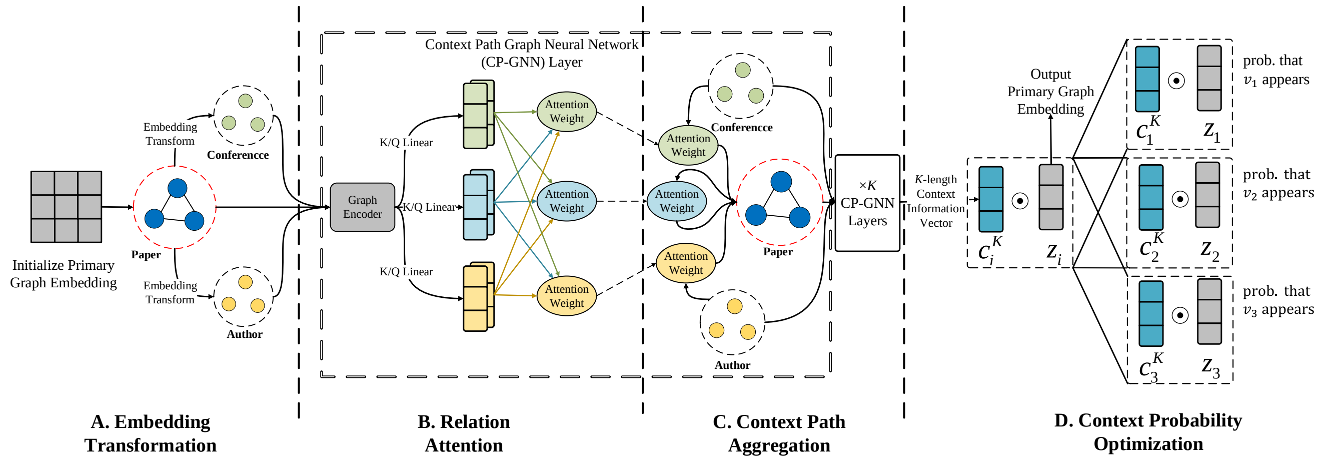


Figure 3: The overall framework of the CP-GNN (A: The embedding of each auxiliary graph is transformed from the primary graph. B: The relation attentions are calculated from the corresponding graph representations. C: The context information vector is generated from the CP-GNN. D: The embedding of the primary graph is optimized with the context probability).

4.1 Objective Function

In this section, we first introduce the context path probability, which is defined as the probability that two nodes v_i and v_j in G_P are context neighbors. Specifically, given a context path $\rho^K = \langle v_i, R^K, v_j \rangle$, the context path probability is

$$p(v_j | v_i, R^K; \theta), \tag{1}$$

where θ is the parameters for computing the probability.

In our model, to learn effective node representations in G_P, a maximum length K is firstly given. Then, for each length $k \in [0, K]$, we aim to maximize the co-occurrence probability of all nodes in G_P and their context neighbors w.r.t. all the k-length context paths. The objective function can be written as

$$arg\,\max_{\theta} O(\theta) = \sum_{v_i \in G_P} \sum_{k=0}^{K} \sum_{v_j \in N_P^k(v_i)} \sum_{\rho^k \in \mathbb{P}^k} log p(v_j | v_i, R^k; \theta), \tag{2}$$

where $N_P^k(v_i)$ is the set of k-length context neighbors of v_i in G_P and $\mathbb{P}^k$ is a set of all k-length context paths connecting v_i and v_j.

Intrinsically, Breadth-first search (BFS) is the easiest way to get all the context paths between nodes. However, the number of context paths increases exponentially with the path length, which makes it impossible to traverse all the context paths, and the walk-based methods (e.g., Node2vec [17], Metapath2vec [57]) are also computational heavily and cannot fully excavate the relationships.

To address this issue, in our CP-GNN model, we adopt graph neural network to recursively embed the high-order relationship of each node into a context information vector c^k to represent the relation information of all context paths with length k, which takes linear time complexity cost. Many previous researches have already adopted the GNN to capture the structure and path information in graph [47, 51]. The GNN message passing is essentially a simulation of BFS, which exhibits the ability to capture paths between nodes [55].

After k times message passing, CP-GNN can embed all the k-length context paths into the context vector c^k. Then, the probability of two context neighbor nodes v_i and v_j connected by all the possible k-length context paths can be approximated by their respective context information vectors c_i^k and c_j^k, written as $p(v_j | v_i, c_i^k, c_j^k; \theta)$, which could be calculated using a softmax function:

$$p(v_j | v_i, c_i^k, c_j^k; \theta) = \frac{exp\left(\varphi(z_i, c_i^k, c_j^k, z_j)\right)}{\sum_{v_x \in G_P} exp\left(\varphi(z_i, c_i^k, c_x^k, z_x)\right)} \tag{3}$$

$$\varphi(z_i, c_i^k, c_j^k, z_j) = \sigma\left((z_i \odot c_i^k)^\top (z_j \odot c_j^k)\right),$$

where z_i and z_j are the node embeddings of v_i and v_j we want to learn, c_i^k and c_j^k denote the context information vectors, $\odot$ denotes the element-wise vector product operation, and $\sigma(\cdot)$ denotes the sigmoid function.

Thus, the objective function could be futher simplified as

$$arg\,\max_{\theta} O(\theta) = \sum_{v_i \in G_P} \sum_{k=0}^{K} \alpha_k \sum_{v_j \in N_P^k(v_i)} log p(v_j | v_i, c_i^k, c_j^k; \theta), \tag{4}$$

where $N_P^k(v_i)$ is the set of k-length context neighbors of v_i in G_P.

To differentiate the importance of context paths with different lengths from 0 to K, we propose a **Context Path Length Attention** mechanism to assign attention weights for different path lengths. For the k-length context path, we use α_k to denote its attention weight, which indicates the importance of the k-length context paths and $\alpha_k \in (0, 1]$. Inspired by a multi-task leaning method [20], we adopt the similar way to optimize α_k during the training. By considering the negative sampling technique, the final model objective function is converted to the following loss function:

$$\mathcal{L} = \sum_{v_i \in G_P} \left(\sum_{k=0}^{K} -\alpha_k \Big(\sum_{v_j \in N_P^k(v_i)} log\varphi(z_i, c_i^k, c_j^k, z_j) + \sum_{v_x \in N_P^{k-}(v_i)} log - \varphi(z_i, c_i^k, c_x^k, z_x) \Big) - log\alpha_k \right), \tag{5}$$

where N_P^{k-} is the set of negative context neighbors of v_i and $\log \alpha_k$ is a penalty to prevent α_k from over small. Currently, the computation complexity of CP-GNN is $O(K \times |V_P| \times (n^+ + n^-))$ that grows linearly with K, where K is the defined maximal context path length, $|V_P|$ is the number of primary nodes, and n^+, n^- are the numbers of the positive and negative sampling neighbors.

Thus we can optimize the parameters with gradient descent written as $\theta \leftarrow \theta - \gamma \nabla_\theta \mathcal{L}(\theta)$, where γ is the learning rate.

4.2 Details of CP-GNN Model

Our CP-GNN model aims to capture the context information and generate the context information vectors for optimizting the final node embedding. CP-GNN has two major components: *Embedding Transformation* for transforming the node embedding from the primary graph to auxiliary graphs, and *CP-GNN Layer* for embedding the high-order relationship of each node into a context information vector.

4.2.1 Embedding Transformation. This component is used to transform the node embedding from the primary graph to the auxiliary graphs along the edges. In this way, we only learn the embedding of nodes in the primary graph thus reducing the parameters to be learned, as we only need to learn the transformation weight matrix.

We also observe that this can achieve even better performance than directly learning the node embeddings for auxiliary graphs in the experiments as shown in Section 5.5. Because we can learn the representations of the primary graph by exploiting the information from the auxiliary graphs by embedding transformation. It can generate the representations of primary graph under different contexts (auxiliary graphs), which is essential in heterogeneous graph representation and community detection [10, 24, 29].

The embedding transformation function $\mathcal{T}_{ST}(\cdot)$ from one graph G_S with node type S to another graph G_T with node type T is

$$Z_T = \sigma(A_{ST} Z_S W + B), \tag{6}$$

where Z_S and Z_T respectively denote the embeddings of nodes in G_S and G_T, A_{ST} is a bipartite adjacency matrix between G_S and G_T, W_{ST} is the transformation weight matrix, and $\sigma(\cdot)$ is the non-linear activation function such as ReLU. The ReLU can be seen as a "Mask" for filtering unnecessary embedding features (values less than 0) during the transformation [52].

4.2.2 Context Path Graph Neural Network Layer. This component recursively captures the high-order relationship from the graph by repeating the Relation Attention and Context Path Aggregation operations.

Relation Attention aims to calculate the attention score of each relation, so that the contributions of different relations are well differentiated. We first use a graph encoder to encode each graph to a summary vector h. After that, the attention score of each relation is calculated based on the graph summary vectors.

To enhance the model robustness, the graph encoder contains a node dropout mechanism which randomly drops nodes from the original graph. Then an averaging operation is adopted to calculate the global graph representation h. Although there exist several techniques to generate the graph summary vector h, the simple averaging operation demonstrates superior performance [32], and thus h is calculated as

$$\begin{aligned} G' &= NodeDropout(G) \\ h &= Mean(C'), \end{aligned} \tag{7}$$

where C' contains the context information vectors of all the nodes in the graph G'.

After calculating the h, for the l-th CP-GNN layer, we calculate the h-head attention score $\alpha_{ST}^{h,l}$ for each relation $r_{ST} \in \mathcal{R}$ by

$$\begin{aligned} h_S^l &= GraphEncoder(C_S^{l-1}) \\ h_T^l &= GraphEncoder(C_T^{l-1}) \\ \alpha_{ST}^{h,l} &= \underset{S \in \mathcal{A}}{Softmax} \frac{Q^h(h_T^l)^\top K^h(h_S^l)}{\sqrt{d}} \\ Q^h(h_T^l) &= QLinear_T^h(h_T^l) \\ K^h(h_S^l) &= KLinear_S^h(h_S^l), \end{aligned} \tag{8}$$

where S and T respectively denote the source and target node types in the relation r_{ST}, h_S^l and h_T^l denote the graph summary vector of G_S and G_T at layer l respectively, C_S^{l-1} and C_T^{l-1} are the context information vectors at the $(l-1)$-th layer, $QLinear$ and $KLinear$ are the linear projection functions that project the graph summary vectors to a Query vector and a Key vector. We want to learn more diverse importance of the relations, thus we adopt total H different heads of Relation Attention with their own parameters to be learned during the training. $\alpha_{ST}^{h,l}$ is the attention weight in head h at layer l for the relation r_{ST}.

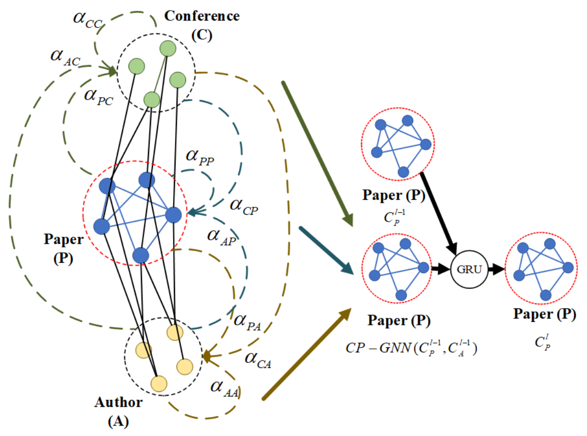


Figure 4: The Context Path Aggregation component.

Context Path Aggregation aims to aggregate the information along relations to generate the context information vectors for all nodes. As discussed in Section 4.1, it is infeasible to enumerate all the context paths with lengths at most K, so we propose to use the context information vectors to approximately compute the probability that two context neighbors are connected by a context path. After calculating the scores of different relationships, we aggregate the information for a node v_i of type T from its one-hop neighbors by adopting the widely used GNN aggregation method. The procedure of this component is shown in Figure 4.

Assume that at layer l, we are going to obtain the context information vector c_i^l for v_i by aggregating the information from its neighbors along different relations. We utilize its neighbors' context information vectors obtained at layer $l-1$, and the computation is as below

$$c_i^l = W_2^l\Big(||_{h\in[1,H]}\sigma(W_1^l \sum_{r_{ST}\in\mathcal{R}} \alpha_{ST}^{h,l} \sum_{v_j\in N_S(i)} c_j^{l-1} + B_1^l) + B_2^l\Big), \tag{9}$$

where $N_S(i)$ denotes the adjacent neighbors of v_i in graph G_S for each relation $r_{ST} \in \mathcal{R}$ relevant to node type T, W_1^l, W_2^l, B_1^l, and B_2^l are the trainable parameters in the l-th layer, and H is the number of different heads. Note that finally we only use the embedding of the primary graph nodes at the k-th layer to obtain k-length context information vectors C_P^k.

In order to get the information of k-length context paths for nodes, we run CP-GNN layer k times to obtain the C_P^k at layer k. The GRU mechanism [8] is also utilized to alleviate the over smoothing problem unusually occured in GNN model [5]. The computation is as below

$$C_P^l = GRU\Big(C_P^{l-1}, CP-GNNLayer(C_P^{l-1}, C_A^{l-1})\Big), \tag{10}$$

where C_A^{l-1} is the embedding of the auxiliary graphs in layer $l-1$, and CP-GNNLayer is the computing process as shown in Eq. 8 and 9. Therefore, the final context information vector c_i^k of each node in G_P can be taken from C_P^k. The overall process of CP-GNN in shown in Algorithm 1.

Algorithm 1: The overall process of CP-GNN.

```
Input: $G = \{\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}\}, G_P = \{\mathcal{V}_P, \mathcal{E}_P\}, K$.
Output: The final embedding $Z_P$.
1  Randomly initialize the $Z_P$, and set the loss $L \leftarrow 0$;
2  Initialize relation attention weight $\alpha_1 \leftarrow 1, \cdots, \alpha_K \leftarrow 1$;
3  for $k = 1, \cdots, K$ do
4      for $v_i \in \mathcal{V}_P$ do
5          for $v_j \in N_P^k(v_i)$ do
6              $C_P^0 \leftarrow Z_P$;
7              $C_A^0 = EmbTransform(Z_P^0)$;
8              for $l = 1, \cdots, k$ do
9                  $C_P^l = GRU\Big(C_P^{l-1}, CP-GNNLayer(C_P^{l-1}, C_A^{l-1})\Big)$;
10             end
11             $L \leftarrow L - \alpha_k \cdot log\varphi(z_i, z_j, c_i^k, c_j^k)$, where $c_i^k, c_j^k \in C_P^k$;
12         end
13     end
14 end
15 Back propagation and update CP-GNN parameters, $\alpha_1, \cdots, \alpha_K, Z_P$;
16 return $Z_P$
```

5 EXPERIMENTS

5.1 Dataset

Three widely used real-world heterogeneous graph datasets, i.e., ACM [48], DBLP [16], IMDB [49], together with a schema-rich knowledge graph dataset AIFB [33] are chosen in the experiments to evaluate the performance of CP-GNN and other baseline models. We report their statistics in Table 1, and discuss their details as follows.

Table 1: Statistics of datasets (the primary node types are marked with "*").

Dataset	Node type	# Nodes	Edge type	# Edges	Meta-path
ACM	*Paper (**P**)	12,499	Paper - Paper	30,789	
	Author (**A**)	17,431	Paper - Author	37,055	PAP
	Subject (**S**)	73	Paper - Subject	12,499	PSP
	Facility (**F**)	1,804	Author - Facility	30,424	
DBLP	*Author (**A**)	14,475	Author - Paper	41,794	APA
	*Paper (**P**)	14,736	Paper - Conference	14,736	APCPA
	Conference (**C**)	20	Paper - Term	114,624	APTPA
	Term (**T**)	8,920			
IMDB	*Movie (**M**)	4,275	Movie - Actor	12,831	MAM
	Actor (**A**)	5,432	Movie - Director	4,181	MDM
	Director (**D**)	2,083	Movie - Keyword	20,428	MKM
	Keyword (**K**)	7,313			
AIFB	7 different types	Total 7,262	104 different types	Total 48,810	-

Table 2: Statistics of the training and testing set.

Dataset	Primary Type	Training	Testing
ACM	Paper	805	3,220
DBLP-A	Author	811	3,246
DBLP-P	Paper	20	80
IMDB	Movie	855	3,420
AIFB	Personen	36	141

- ACM dataset [48] is a bibliographic information network with four types of nodes. We use paper nodes to generate the primary graph and the rest three types of nodes are respectively used to construct auxiliary graphs. In the original dataset, the paper nodes are categorized into 3 classes, i.e., *database*, *wireless communication* and *data mining*. To evaluate the compared models, we pre-define two meta-paths according to [48], i.e., Paper-Author-Paper (PAP), and Paper-Subject-Paper (PSP).
- DBLP dataset [16] is a monthly updated citation network consisting of four node types. As we mentioned in Definition 3.2, any node type can be chosen as the primary type. Thus we choose the author node and paper as the primary node, respectively, with four classes, i.e., *database*, *data mining*, *information retrieval* and *machine learning*. They are denoted as DBLP-A and DBLP-P in the following section. We use three meta-paths for this dataset, which are Author-Paper-Author (APA), Author-Paper-Conference-Paper-Author (APCPA), and Author-Paper-Term-Paper-Author (APTPA).
- IMDB dataset [49] consists of four types of nodes, i.e., "Director", "Actors", "Movie" and "Key word". We choose the movie node as the primary node with three classes, i.e., *Action*, *Comedy* and *Drama*. We also use three meta-paths on this network, i.e., Movie-Actor-Movie (MAM), Movie-Director-Movie (MDM), and Movie-Keyword-Movie (MKM).
- AIFB dataset [33] is a knowledge graph dataset consists of 7 types of nodes and 104 types of edges. We choose the "Personen" node as the primary node with four classes. Due to the complexity of the graph, we do not provide detail illustration in Table 1, and pre-define the meta-paths by ourselves.

Notabally, we adopt the meta-paths in ACM, DBLP, and IMDP defined by previous works to evaluate the meta-path-based methods.

Due to the rich-schema property of AIFB, we do not define the meta-path by ourselves. Besides, since the unsupervised methods do not need the training data, to make a fair comparison between the unsupervised and supervised methods, the splits of the labled primary nodes for training and testing are shown in Table 2.

5.2 Baseline Methods

To evaluate the effectiveness of our approach, we compare it with a list of the state-of-the-art unsupervised methods (i.e., Node2vec [17], Metapath2vec [9], and HIN2vec [15]) and supervised baseline methods (i.e., GCN [22], GAT [44], LGNN [7], HAN [48], and HGT [18]). Especially, HGT is a semi-supervised neural network model which adopts the transformer mechanism to capture the importance of relations. It is considered as the SOTA approach.

Except for graph embedding-based and GNN-based baselines, we also choose two traditional community detection methods (i.e., InfoMap [34] and LP (Label Propagation) [31]), for comparison.

5.3 Settings of Model Parameters

We now briefly discuss the settings of model parameters. For unsupervised approaches such as Node2vec, Metapath2vec, and HIN2vec, we respectively set the length of a random walk to 20, the sampling window size to 3, the number of walks per node to 5, and the number of negative samplings to 3. For supervised-based methods such as GCN, GAT, LGNN, HAN and HGT, the number of graph convolution layer is set to 2, and their node features are first randomly initialized then updated during the model learning process. The dimension of node feature embedding for all compared methods is set to 128.

For our CP-GNN, the number of attention heads is set to 8, the dimension of the output vectors of K/Q-Linear components is set to 128, and the node dropout rate is set to 0.3. The number of positive context neighbors for each node in a context path is set to 20, and the corresponding negative sampling size is set to 3. The Adam [21] is adopted to optimize all models, and the learning rate is set to 0.05.

5.4 Performance Comparison

In this experiment, we first learn the node embeddings and then employ the k-Means algorithm on these embeddings to detect communities, where k is set to the number of node classes. We evaluate the community detection results using the ground-truth labels with four commonly adopted community detection evaluation metrics, i.e., F1, NMI, ARI, and Purity, and report the performance results in Table 3. Note that due to the lack of pre-defined meta-paths, the meta-path-based methods (i.e., Metapath2vec and HAN) cannot be evaluated on AIFB.

From Table 3, we can clearly see that CP-GNN outperforms all the baselines on most metrics. This demonstrates that via the context paths, it can learn node representations that are more suitable for community detection by capturing more meaningful and high-order relationships.

For unsupervised methods, HIN2vec achieves better performance than Node2vec and Metapath2vec on ACM but worse on other datasets. It shows that even though HIN2vec can automatically discover meta-paths by random walk, the discovered meta-paths may not be suitable for community detection. Besides, HIN2vec does not differentiate the importance of the found meta-paths, which incorporates some unrelated relations to the community detection result. The performances of the Metapath2vec are better on DBLP than ACM, when using the longer meta-path. This demonstrates the importance of high-order relationships. Last, all the graph embedding-based methods are better than InfoMap, which shows the great potential of them in community detection task.

For supervised methods, LP performs well in schema-simple heterogeneous graphs (i.e., ACM and IMDB), but its performance drops quickly when it meets the schema-rich heterogeneous graph (i.e., AIFB). This indicates the simple label propagation mechanism does not consider the complex relations in heterogeneous graphs, which leads to its poor performance. GCN and LGNN achieve the worst results in the GNN-based baselines. The possible reason is that they are originally proposed for homogeneous graph, thus they do not consider the complex context information in heterogeneous graph. GAT performs better than GCN and LGNN, which strongly supports the importance of the attention mechanism. The attention mechanism used in GAT can be regarded as a simple way to differentiate the node type and edge type in heterogeneous graph. Thanks to the meta-path, HAN can explicitly excavates the complex semantic information and reaches a better result. In addition, HGT achieves the second-best performance since it can capture more diverse relations without the limitation of the meta-paths, and be easily fit into different datasets. However, the HGT still requires the labeled data to optimize the model, which strongly limits its application.

Last, as Definition 3.2 said, each node type can be treated as primary type. Therefore, the result in DBLP-A and DBLP-P shows that no matter what node type is chosen as the primary type, the proposed GP-GNN can still achieve better results with the help of other auxiliary nodes.

5.5 Parameters Analysis and Ablation Study

In this section, we experimentally investigate the sensitivity of the parameters and report the results on the ACM dataset with various parameters shown in Figure 5, and various CP-GNN structures shown in Tables 4 and 5.

As shown in Figure 5, with the increase of parameter values, CP-GNN's performances raise first and then drop slightly; the best performances are reached when the Embedding size, Attention head, and Node droupout rate reach 128, 8, and 0.3, respectively. The reason is that an over large embedding dimension may introduce unnecessary redundancies to the CP-GNN model. Although more attention heads can capture more diverse relation importance and increase the representation ability, they also introduce more parameters to the model, making it hard to train. Besides, too many nodes are dropped, which causes that the graph summary vectors cannot be well generated from the remaining nodes.

In Table 4, we evaluate the effect of the embedding transformation mechanism where *CP-GNN w/o Trans.* means that the initial embeddings of all the auxiliary nodes are randomly initialized and optimized during the training; *CP-GNN w/o ReLU* denotes that the initial embeddings are transformed from the primary node embeddings but without non-linear function in Eq. 6. *CP-GNN* denotes the final embedding transformation function used in our CP-GNN. The experiment results show that our non-linear transformation function

Table 3: Comparison of the performance of different community detection methods.

Dataset	Metrics	InfoMap	Node2vec	Metapath2vec	HIN2vec	LP	GCN	GAT	LGNN	HAN	HGT	CP-GNN
ACM	F1	0.5733	0.6954	0.7142	0.7732	0.6691	0.5366	0.6876	0.6987	0.7922	0.7599	**0.8596**
	NMI	0.1933	0.2666	0.3596	0.4066	0.3933	0.0966	0.2577	0.2746	0.394	0.4509	**0.4832**
	ARI	0.1286	0.2469	0.2956	0.3313	0.2992	0.1022	0.1422	0.2368	0.319	0.3813	**0.3924**
	Purity	0.5876	0.4355	0.4969	0.6969	0.6691	0.5808	0.6186	0.6594	0.6942	0.7032	**0.715**
DBLP-A	F1	0.3601	0.7572	0.7144	0.313	0.2451	0.32	0.9023	0.321	0.9023	**0.9386**	0.9125
	NMI	0.0819	0.0638	0.2554	0.0044	0.0984	0.0186	0.618	0.0069	0.624	0.7032	**0.7089**
	ARI	0.0131	0.0409	0.2722	0.0022	0.0033	0.0166	0.5264	-0.0012	0.665	0.7322	**0.766**
	Purity	0.3601	0.3884	0.6169	0.2971	0.2935	0.3564	0.7476	0.2988	0.8496	**0.9325**	0.9004
DBLP-P	F1	0.3111	0.3	0.3125	0.3375	0.4	0.31	0.3	0.225	0.3375	0.4	**0.4875**
	NMI	0.0463	0.0655	0.0034	0.0514	0.0429	0.0171	0.0495	0.0431	0.0732	0.1086	**0.1846**
	ARI	0.0032	-0.0016	0.0013	-0.0021	0.0042	-0.0048	-0.0029	0.0016	-0.0103	0.0724	**0.0564**
	Purity	0.4111	0.432	0.4212	0.431	0.4	0.41	0.422	0.44	0.426	0.426	**0.488**
IMDB	F1	0.3038	0.5494	0.488	0.4184	0.3826	0.3628	0.3587	0.3646	0.4888	0.3634	**0.614**
	NMI	0.0098	0.0745	0.027	0.0031	0.0081	0.0018	0.0012	0.0158	0.1172	0.0101	**0.1225**
	ARI	-0.005	0.0471	0.0146	-0.0022	-0.0004	0.0013	-0.0009	-0.0079	**0.131**	0.0083	0.1231
	Purity	0.3898	0.4442	0.438	0.3744	0.3825	0.3885	0.3746	0.3738	0.3734	0.4023	**0.4949**
AIFB	F1	0.434	0.7517	-	0.6524	0.4151	0.6524	0.7375	0.6809	-	0.7163	**0.7659**
	NMI	0.0645	0.2401	-	0.1912	0.2216	0.1567	0.2117	0.2435	-	0.3812	**0.4147**
	ARI	0.0286	0.1518	-	0.1202	0.0985	0.1248	0.1142	0.079	-	0.3011	**0.3898**
	Purity	0.4403	0.6091	-	0.5494	0.5157	0.5835	0.5568	0.5875	-	**0.7102**	0.6966

Table 4: Effectiveness of embedding transform functions.

Trans. Function	NMI	ARI
CP-GNN *w/o* Trans.	0.3397	0.3665
CP-GNN *w/o* ReLU	0.3908	0.2908
CP-GNN	**0.4832**	**0.3924**

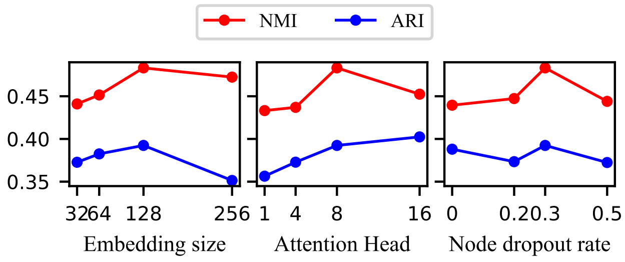


Figure 5: Parameter sensitivity w.r.t. different parameters.

performs the best. The reason is that using the proposed transformation function can establish stronger connections between the primary graph and auxiliary graphs. Meanwhile, it provides different contextual representation of the primary graph. Besides, the non-linear function, such as ReLU, can "mask" some unimportant features during the transformation to provide better result.

In Table 5, to examine the effectiveness of different attention methods, we gradually remove the relation and length attention mechanism. *CP-GNN w/o Attention* means CP-GNN without any attention mechanism. *CP-GNN w/o Relation Att.* and *CP-GNN w/o Att.* respectively denote the CP-GNN without relation attention and context path length attentnion. The experimental results show that

Table 5: Effectiveness of attention mechanisms.

Attention	NMI	ARI
CP-GNN *w/o* Attention	0.2214	0.1132
CP-GNN *w/o* Relation Att.	0.3239	0.1008
CP-GNN *w/o* Length Att.	0.4452	0.3739
CP-GNN	**0.4832**	**0.3924**

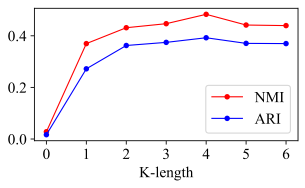
both the context path length attention and the relation attention are helpful for improving the model performance.

5.6 Case Study

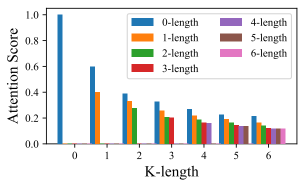
5.6.1 Context Path Length Attention. To analyze the effect of context path length attention, we conduct experiments on the ACM dataset with different maximum context path length K. The experiment results and attention weight of each context path length are depicted in Figure 6(a) and 6(b). Clearly, with the increase of K, the performances of CP-GNN increases and reaches the best when K=4. This demonstrates our assumption that the high-order relationship information is crucial for community detection. When $K > 4$, the performance of the model slightly drops, which may due to the fact that the nodes will be fully connected when K is overly increasing.

Besides, by reporting the attention score of each length, we can see that with the context path length increases, the corresponding attention weight decreases, which is consistent to the common sense that the short paths often reflect stronger connections than the long ones. What is more, the attention scores of lengths longer than 4 (i.e., $k = 5, 6$) are barely the same. It indicates the less relevance of these paths, and explains why the performance of CP-GNN drops slightly when $K > 4$. By using the context path length attention, we not only capture the high-order relationships among the graph nodes, but also discard some less important high-order relationships during learning.

Since the context path length attention can discriminate the importance of context paths with different lengths and assign lower importance to unnecessary long meta-paths, we can set a slightly larger value for K, which can yield a result closed to the optimal one. As demonstrated by Figure 6(a) where the optimal result is reached when K=4 but the results are close when K=5 or 6. This will alleviate the model's dependence on the hyper-parameter K.

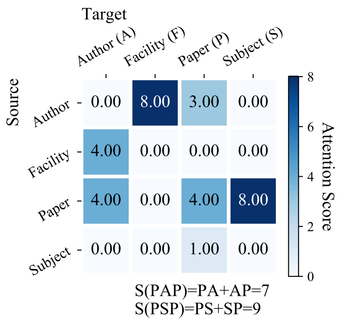


(a) CP-GNN's performances under different K-length context paths.

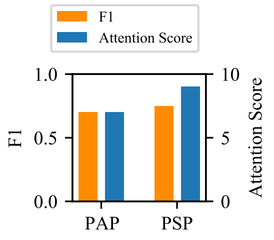


(b) Attention weight of each context path length.

Figure 6: Visualization of the context path length attention mechanism.



(a) Attention matrix of 1-length context path on ACM.

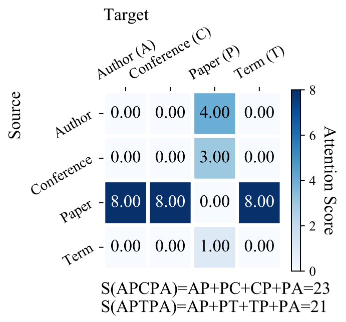


(b) Metapath2vec F1 values on ACM.

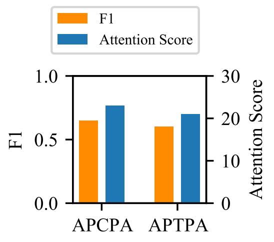
Figure 7: Visualization of the context relation attention matrix on ACM.

5.6.2 Relation Attention. To further analyze whether CP-GNN can differentiate the context paths, we first present the corresponding relations attention matrix acquired from CP-GNN in Figure 7(a) and 8(a), where each entry is the attention score of the relation with a source node type and a target node type. The attention score of each context path can be computed by summarizing the scores of its relations. Then, to justify whether the paths with higher attention score are more meaningful for community detection, we adopt the Metapath2vec to evaluate the effect of each path. The results are shown in Figure 7(b) and 8(b)

For example, the paths PAP and PSP in ACM are both 1-length context path. Therefore, there attention scores can be computed from the relation attention matrix of 1-length context path shown in Figure 7(a) where $S(PAP) = PA+AP = 7$ and $S(PSP) = PS+SP = 9$.



(a) Attention matrix of 3-length context path on DBLP.



(b) Metapath2vec F1 values on DBLP.

Figure 8: Visualization of the context relation attention matrix on DBLP.

Clearly, we can find that the attention score of PSP is higher than PAP, which means the path PSP is slightly more important than PAP for community detection. This can be justified by the result shown in Figure 7(b) where the F1 score of PSP is higher than PAP. Similarly, from Figure 8(a), the attention scores of paths APCPA and APTPA are $S(APCPA) = AP+PC+CP+PA = 23$ and $S(APTPA) = AP+PT+TP+PA = 21$. This indicates that the relationship reflected by APCPA is a little bit more important than that of APTPA. This finding can be proved by the result shown in Figure 8(b) where the F1 of APCPA is higher than APTAP.

In summary, the above analysis demonstrates that the Relation Attention can discover context path of different importance and capture the context path of high importance that are more meaningful and useful for community detection.

6 CONCLUSION

In this paper, we propose the Context Path-based Graph Neural Network (CP-GNN) model for detecting communities from heterogeneous graphs, which not only avoids using pre-defined meta-paths, but also well captures the high-order relationship among nodes. In particular, we adopt the context path and propose the context path probability to model the objective function. Besides, CP-GNN distinguishes the importance of different context paths. Extensive experiments on real-world datasets show that CP-GNN outperforms the baselines, and its attention mechanisms can well differentiate the importance of different context paths.

7 ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61872108, and the Shenzhen Science and Technology Program under Grant No. JCYJ20200109113201726, JCYJ20170811153507788. Xin Cao is supported by ARC DE190100663. Yixiang Fang is supported by CUHK-SZ grant UDF01002139. Wenjie Zhang is supported by ARC DP200101116.

REFERENCES

[1] Debaditya Barman, Subhayan Bhattacharya, Ritam Sarkar, and Nirmalya Chowdhury. 2019. *k*-Context Technique: A Method for Identifying Dense Subgraphs in a Heterogeneous Information Network. *IEEE Transactions on Computational Social Systems* 6, 6 (2019), 1190–1205.

[2] Brigitte Boden, Martin Ester, and Thomas Seidl. 2014. Density-based subspace clustering in heterogeneous networks. In *ECML PKDD*. Springer, 149–164.

[3] Deng Cai, Zheng Shao, Xiaofei He, Xifeng Yan, and Jiawei Han. 2005. Mining hidden community in heterogeneous social networks. In *Proceedings of the 3rd international workshop on Link discovery*. 58–65.

[4] Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. 2017. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 377–386.

[5] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3438–3445.

[6] Yankai Chen, Jie Zhang, Yixiang Fang, Xin Cao, and Irwin King. 2020. Efficient Community Search over Large Directed Graph: An Augmented Index-based Approach.. In *IJCAI*. 3544–3550.

[7] Zhengdao Chen, Lisha Li, and Joan Bruna. 2019. Supervised Community Detection with Line Graph Neural Networks. In *ICLR*.

[8] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[9] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *SIGKDD*. 135–144.

[10] Alessandro Epasto and Bryan Perozzi. 2019. Is a single embedding enough? learning node representations that capture multiple social contexts. In *The World Wide Web Conference*. 394–404.

[11] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. 2020. A survey of community search over big graphs. *The VLDB Journal* 29, 1 (2020), 353–392.

[12] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. 2020. Effective and efficient community search over large heterogeneous information networks. *PVLDB* 13, 6 (2020), 854–867.

[13] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. 2020. Effective and Efficient Community Search over Large Heterogeneous Information Networks. *Proc. VLDB Endow.* 13, 6 (Feb. 2020), 854–867. https://doi.org/10.14778/3380750.3380756

[14] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[15] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *CIKM*. 1797–1806.

[16] Jing Gao, Feng Liang, Wei Fan, Yizhou Sun, and Jiawei Han. 2009. Graph-based consensus maximization among multiple supervised and unsupervised models. In *NIPS*. 585–593.

[17] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*. 855–864.

[18] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *WWW*. 2704–2710.

[19] Di Jin, Zhizhi Yu, Pengfei Jiao, Shirui Pan, Philip S Yu, and Weixiong Zhang. 2021. A Survey of Community Detection Approaches: From Statistical Modeling to Deep Learning. *arXiv preprint arXiv:2101.01669* (2021).

[20] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*. 7482–7491.

[21] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[22] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[23] Ye Li, Chaofeng Sha, Xin Huang, and Yanchun Zhang. 2018. Community detection in attributed graphs: An embedding approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[24] Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu. 2019. Is a single vector enough? exploring node polysemy for network embedding. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 932–940.

[25] Linhao Luo, Kai Liu, Dan Peng, Yaolin Ying, and Xiaofeng Zhang. 2020. A Motif-Based Graph Neural Network to Reciprocal Recommendation for Online Dating. In *International Conference on Neural Information Processing*. Springer, 102–114.

[26] Fragkiskos D Malliaros and Michalis Vazirgiannis. 2013. Clustering and community detection in directed networks: A survey. *Physics reports* 533, 4 (2013), 95–142.

[27] Vincenzo Moscato and Giancarlo Sperlì. 2021. A survey about community detection over On-line Social and Heterogeneous Information Networks. *Knowledge-Based Systems* (2021), 107112.

[28] Mark EJ Newman. 2004. Finding and Evaluating Community Structurein Networks. *Physical Review E* 69, 26113 (2004), 1–16.

[29] Chanyoung Park, Carl Yang, Qi Zhu, Donghyun Kim, Hwanjo Yu, and Jiawei Han. 2020. Unsupervised Differentiable Multi-aspect Network Embedding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1435–1445.

[30] Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. 2012. On clustering heterogeneous social media objects with outlier links. In *WSDM*. 553–562.

[31] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E* 76, 3 (2007), 036106.

[32] Yuxiang Ren, Bo Liu, Chao Huang, Peng Dai, Liefeng Bo, and Jiawei Zhang. 2019. Heterogeneous deep graph infomax. *arXiv preprint arXiv:1911.08538* (2019).

[33] Petar Ristoski, Gerben Klaas Dirk De Vries, and Heiko Paulheim. 2016. A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web. In *International Semantic Web Conference*. Springer, 186–194.

[34] Martin Rosvall and Carl T Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences* 105, 4 (2008), 1118–1123.

[35] Marta Sales-Pardo, Roger Guimera, André A Moreira, and Luís A Nunes Amaral. 2007. Extracting the hierarchical organization of complex systems. *PNAS* 104, 39 (2007), 15224–15229.

[36] Venu Satuluri, Yao Wu, Xun Zheng, Yilei Qian, Brian Wichers, Qieyun Dai, Gui Ming Tang, Jerry Jiang, and Jimmy Lin. 2020. SimClusters: Community-Based Representations for Heterogeneous Recommendations at Twitter. In *SIGKDD*. 3183–3193.

[37] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. 2016. A survey of heterogeneous information network analysis. *TKDE* 29, 1 (2016), 17–37.

[38] Chuan Shi, Ran Wang, Yitong Li, Philip S Yu, and Bin Wu. 2014. Ranking-based clustering on general heterogeneous information networks by network projection. In *CIKM*. 699–708.

[39] Heli Sun, Fang He, Jianbin Huang, Yizhou Sun, Yang Li, Chenyu Wang, Liang He, Zhongbin Sun, and Xiaolin Jia. 2020. Network embedding for community detection in attributed networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14, 3 (2020), 1–25.

[40] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDP* 4, 11 (2011), 992–1003.

[41] Yizhou Sun, Brandon Norick, Jiawei Han, Xifeng Yan, Philip S Yu, and Xiao Yu. 2013. Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. *TKDD* 7, 3 (2013), 1–23.

[42] Yizhou Sun, Yintao Yu, and Jiawei Han. 2009. Ranking-based clustering of heterogeneous information networks with star network schema. In *SIGKDD*. 797–806.

[43] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. 2014. Learning deep representations for graph clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 28.

[44] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[45] Ken Wakita and Toshiyuki Tsurumi. 2007. Finding community structure in mega-scale social networks. In *WWW*. 1275–1276.

[46] Jing Wang and Ioannis Ch Paschalidis. 2016. Botnet detection based on anomaly and community detection. *IEEE Transactions on Control of Network Systems* 4, 2 (2016), 392–404.

[47] Ping Wang, Khushbu Agarwal, Colby Ham, Sutanay Choudhury, and Chandan K Reddy. 2021. Self-Supervised Learning of Contextual Embeddings for Link Prediction in Heterogeneous Networks. *Proceedings of The Web Conference 2021* (2021).

[48] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*. 2022–2032.

[49] Chao-Yuan Wu, Alex Beutel, Amr Ahmed, and Alexander J Smola. 2016. Explaining reviews and ratings with paco: Poisson additive co-clustering. In *WWW*. 127–128.

[50] Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*. PMLR, 478–487.

[51] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[52] Carl Yang et al. 2020. MultiSage: Empowering GCN with Contextualized Multi-Embeddings on Web-Scale Multipartite Networks. In *SIGKDD*. 2434–2443.

[53] Liang Yang, Xiaochun Cao, Dongxiao He, Chuan Wang, Xiao Wang, and Weixiong Zhang. 2016. Modularity Based Community Detection with Deep Learning.. In *IJCAI*, Vol. 16. 2252–2258.

[54] Qi Ye et al. 2010. Distance distribution and average shortest path length estimation in real-world networks. In *ADMA*. 322–333.

[55] Jiaxuan You et al. 2021. Identity-aware Graph Neural Networks. In *AAAI*.

[56] Binbin Zhang, Zhizhi Yu, and Weixiong Zhang. 2020. Community-Centric Graph Convolutional Network for Unsupervised Community Detection. IJCAI.

[57] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2018. Metagraph2vec: Complex semantic path augmented heterogeneous network embedding. In *PAKDD*. Springer, 196–208.

[58] Chen Zhe, Aixin Sun, and Xiaokui Xiao. 2019. Community detection on large complex attribute network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2041–2049.

[59] Yaping Zheng, Shiyi Chen, Xinni Zhang, Xiaofeng Zhang, Xiaofei Yang, and Di Wang. 2019. Heterogeneous-Temporal Graph Convolutional Networks: Make the Community Detection Much Better. *arXiv preprint arXiv:1909.10248* (2019).

[60] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment* 2, 1 (2009), 718–729.