Sampling for Heterogeneous Graph Neural Networks

Feiyang Chen, Yongqian Li, Ruoyu He, YuanChing Lin*
University of California, Los Angeles
Los Angeles, CA 90095
fychen@cs.ucla.edu,yongqianli@g.ucla.edu,rhe9527@g.ucla.edu,yclin99@g.ucla.edu

ABSTRACT

Graph sampling is a popular technique in training large-scale graph neural networks (GNNs); recent sampling-based methods have demonstrated impressive success for homogeneous graphs. However, in practice, the interaction between different entities is often different based on their relationship, i.e., the network in reality is mostly heterogeneous. But only a few of the recent works have paid attention to sampling methods on heterogeneous graphs. In this work, we aim to study sampling for heterogeneous GNNs. We propose two general pipelines for heterogeneous sampling. Based on the proposed pipeline, we evaluate 3 representative sampling methods on heterogeneous graphs, including node-wise sampling, layer-wise sampling, and subgraph-wise sampling. To the best of our knowledge, we are the first to provide a thorough implementation, evaluation, and discussion of each sampling method on heterogeneous graphs. Extensive experiments compared sampling methods from multiple aspects and highlight their characteristics for each category. Evaluation of scalability on larger-scale heterogeneous graphs also shows we achieve the trade-off between efficiency and effectiveness. Last, we also analyze the limitations of our proposed pipeline on heterogeneous sub-graph sampling and provide a detailed comparison with HGSampling. Our code is available at: *https://github.com/Eurus-Holmes/Heterogeneous_Sampling*.

1 INTRODUCTION

Objects in the real world are often defined in terms of their relationships, such a set of objects and interactions between them is naturally represented as a graph. Graph neural networks (GNNs) [1, 2] have recently received more and more attention due to their power of modeling graph-structured data. Successful applications of GNNs include drug discovery [3], recommendation systems [4], traffic prediction [5], etc.

However, training large-scale GCNs efficiently is still a challenge. Naive full-batch training of GCNs requires loading all the nodes into the memory, and generates node embeddings by aggregating neighboring node features [2]. Full-batch training therefore becomes impractical when the graph is large and dense[6]. To circumvent this issue, GraphSAGE [6] first proposed a mini-batch training setting for GCNs, namely node-wise sampling, which samples a neighborhood for each node required for computation instead of iterating over all nodes. Different strategies of sampling have been exploited thereafter [7–9]. However, most sampling methods [6–9] focus on homogeneous graphs. But relationships in real world often exhibit in the form of heterogeneous graphs, i.e. entities and interactions are often multi-typed. Unfortunately, only a few of the recent works have paid attention to sampling for heterogeneous graphs [10, 11].

In this work, we aim to study sampling for heterogeneous GNNs. The ideal model should be able to sample on heterogeneous graph efficiently without compromising the training effectiveness. We propose two general pipelines for heterogeneous sampling. One is to preserve the node and relationship types on the heterogeneous graph through the mapping operation, to directly convert the heterogeneous graph into a homogeneous graph, and then use the sampling method on the homogeneous graph to sample. The other is to use relation-specific transformations to target each different relation, then sampling on each relation separately, and the sampled information is pooled to train on heterogeneous GNNs. Based on the proposed pipelines, we evaluate 3 representative types ofsampling methods on heterogeneous graphs, including node-wise sampling, layer-wise sampling, and subgraph-wise sampling. Extensive experiments compared sampling methods from multiple aspects and highlight their characteristics for each category. For node-wise sampling, easily leads to exponential neighborhood expansion. While layer-wise sampling is the improvement of node-wise sampling, through sampling a small set of nodes together to avoid the exponential extension of neighbors. For subgraph-wise sampling, we will discuss detail in Section 5. At last, we also analyze the limitations of our proposed pipeline on heterogeneous sub-graph sampling and provide a detailed comparison with HGSampling [12].

*Equal contribution

SIGCOMM'18, August 21-23, 2018, Budapest, Hungary

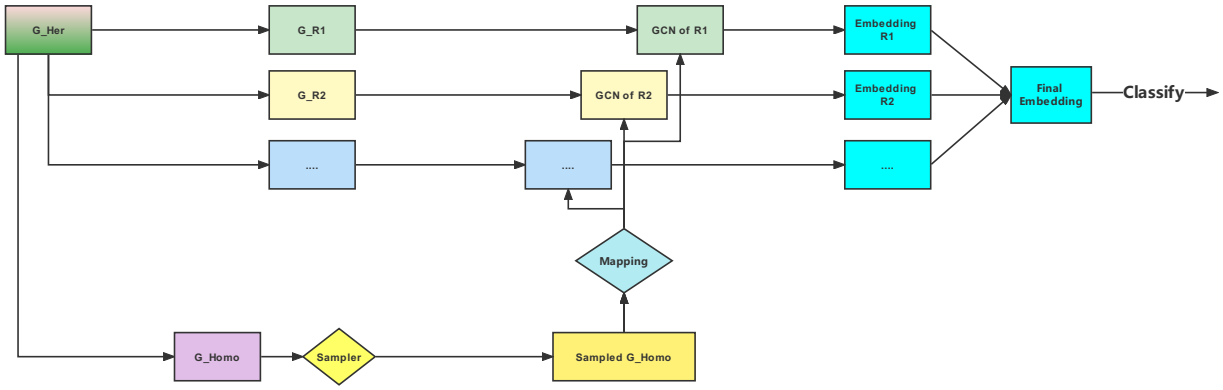

Figure 1: Homo2Hete pipeline for heterogeneous sampling.

We highlight our contributions as follows:

- We propose two general pipelines for heterogeneous sampling and evaluate 3 representative sampling methods on heterogeneous graphs based on the proposed pipelines.
- To the best of our knowledge, we are the first to provide a thorough implementation, evaluation, and discussion of each sampling method on heterogeneous graphs.
- At last, we also analyze the limitations of our proposed pipelines on heterogeneous sub-graph sampling and provide a detailed comparison with HGSampling.

2 SAMPLING METHODS

In this section, we introduce 3 representative sampling methods by category, including node-wise sampling, layer-wise sampling, and subgraph-wise sampling. Following [13]'s work, the taxonomy is based on the granularity of the sampling operation in one sampling batch.

2.1 Node-wise Sampling

Node-wise sampling is the fundamental sampling method. In one sampling batch, sampling a small set of neighbors of a single node for each layer, which means for each node at each layer, only aggregating the information in the sampled neighborhood. Formally, the sampling process of k-th layer can be described by the following formula:

$$SN(v) = \text{Sampling}^{(k)}\left(N(v), P, RN^{(k)}\right) \tag{1}$$

$$\begin{cases} P \sim \text{Uniform}(0, M), & \text{Random} \\ P \propto \text{Metrics}(v), & \text{Non-Random} \end{cases} \tag{2}$$

where $SN(v)$ is the sampled neighbors from $N(v)$ (the neighboring nodes of node v) based on specific probability P, either random sampling or non-random sampling. For random sampling, P obeys uniform distribution between 0 and M (the maximum number of neighbors to be sampled of node v). For non-random sampling, P is proportional to particular metrics of node v, which is pre-computed before performing the sampling process, e.g., PinSage [14] samples neighbors through computing the L1-normalized visit counts. $RN^{(k)}$ is the restricted number of neighbors to be sampled in one sampling batch in the k-th layer. Because sampling all neighbors of each node in the training process is inefficient, the sampling size of neighbors cannot be arbitrarily large. We add this restricted value to be an appropriate value for flexibly controlling.

In this paper, we focus on a classical node-wise sampling method **GraphSAGE** [6], which first introduced the sampling strategy into GCN training. Specifically, the sampling operation randomly selects neighbors for each node at each layer in the graph, followed by aggregation. Aggregation utilizes the features of sampled neighbors in the first layer and the later layers use the output of previous layers, generating an embedding for each node from the top layer to the last layer. The output embeddings are then used for model weight updates and other downstream applications. Overall, GraphSAGE aims to learn an inductive process to generate node embeddings using neighborhood sampling and aggregation for training model efficiency, inspiring the subsequent series of node-wise sampling works. For example, PinSage [14] only sample the most influential neighbors for each node by simulating a random walk process to compute the visit counts. SSE [15] leverages the alternating sampling strategy to sample 1- hop neighbors for embedding computation. VR-GCN [16] uses the historical activation to approximate the embedding to achieve comparable predictive performance with an arbitrarily small sampling size. However, these methods suffer from an obvious flaw in that they recursively sample neighbors for each node, which leads to exponential

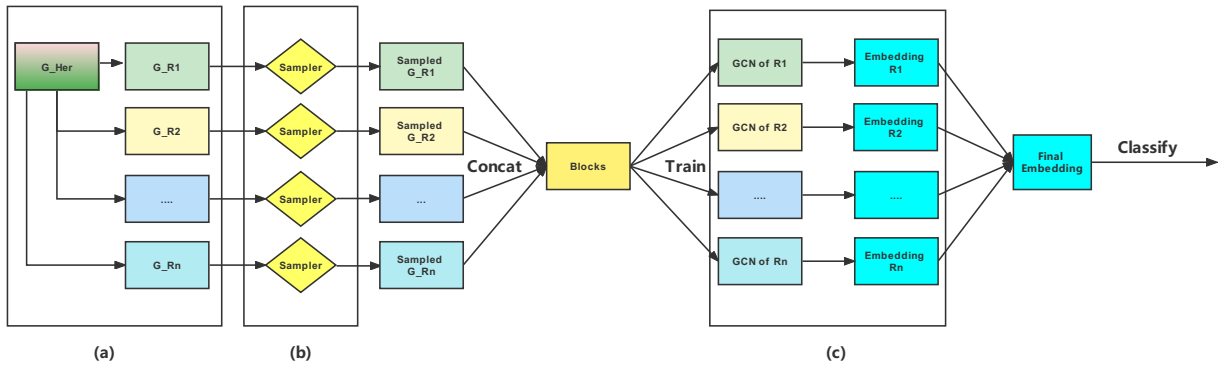

Figure 2: HeteSampling pipeline for heterogeneous sampling. (a). relation-specific transformations inspired by RGCN; (b). Sampling Methods; (c). Heterogeneous GNNs (e.g. RGCN).

neighborhood expansion and incurs significant computational and memory costs, which may degrade the efficiency of the method.

2.2 Layer-wise Sampling

Layer-wise sampling is the improvement of node-wise sampling through sampling a small set of nodes together in one sampling step, which means multiple nodes in each layer are sampled simultaneously, then in the aggregation step, for each node, only aggregate the information among the neighbor nodes that are in the sampled node set for the previous layer. In this way, time cost of the sampling process is significantly reduced by avoiding the exponential extension of neighbors.

In this paper, we begin with **FastGCN** [17] to study layer-wise sampling method, which sample a certain number of nodes in each layer independently based on the pre-set probability distribution. More importantly, FastGCN first introduced an integral transformation of the embedding function to interpret the convolution operation in GCN. Formally, consider the layer-wise propagation rule as an integral format:

$$h^{(l+1)}(v) = \sigma\left(\int \hat{A}(v,u)h^{(l)}(u)W^{(l)}dP(u)\right) \tag{3}$$

where $\sigma(\cdot)$ denotes a specific activation function, $\hat{A}(v,u)$ is a renormalized adjacency matrix, $h^{(l)}(u)$ and $W^{(l)}$ denote the hidden feature vector of u and weight matrix in the l-th layer, respectively. The probability P is the sampling distribution to represent various sampling, such as uniform probability for independent sampling and conditional probability for node/layer-dependent sampling. When one node is sampled under the influence of other nodes in the current layer, the impact of other nodes in the form of importance sampling as:

$$h^{(l+1)}(v) = \sigma\left(\hat{A}(v,u)\mathbb{E}_{q(u)}\left[\frac{p(u \mid v)\cdot h^{(l)}(u)}{q(u)}\right]W^{(l)}\right) \tag{4}$$

where $q(u)$ denotes the probability of sampling node u under the condition that all nodes in the current layer are given, which can be estimated the expectation $\mu_q(v)$ using the Monte Carlo approach:

$$\mu_q(v) = \mathbb{E}_{q(u)}\left[\frac{p(u \mid v)\cdot h^{(l)}(u)}{q(u)}\right] \tag{5}$$

$$\hat{\mu}_q(v_i) = \frac{1}{n}\sum_{j=1}^{n}\frac{p(\hat{u}_j \mid v_i)\cdot h^{(l)}(\hat{u}_j)}{q(\hat{u}_j \mid v_1,\cdots,v_n)} \tag{6}$$

where $\hat{\mu}_q(v_i)$ is the approximate expectation, $q(\hat{u}_j \mid v_1,\cdots,v_n)$ is the specified format of $q(u)$ that denotes the probability of sampling node u_j given nodes $v_1, v_2, ..., v_n$ in the current layer. Then the target for optimizing sampling is to reduce the variance of $\hat{\mu}_q(v_i)$ as far as possible, which means find an optimal sampling probability $q^*(u_j)$ to minimize $\mathrm{Var}_q(\hat{\mu}_q(v_i))$. In this way, the embedding is approximatively evaluated by sampling t_l nodes in each layer. Specifically, FastGCN defines the optimal probability that is proportional to $\|\widehat{A}(:,u)\|^2$:

$$q(u) = \|\widehat{A}(:,u)\|^2 \sum_{u'\in V}\left\|\widehat{A}(:,u')\right\|^2, u \in V \tag{7}$$

Based on the q(u), t_l nodes are sampled in each layer independently, and the inter-layer connections are reconstructed after the sampling process to link the sampled nodes. Thereby, GCN training can be represented as an inductive learning process and achieve a considerable speedup. Such importance sampling technique has been widely used in later works. AS-GCN [18] samples n nodes based on the parent nodes sampled in the upper layer, where the sampling process is probability-based and dependent between layers. LADIES [7] samples n nodes per layer with the restriction that nodes

being sampled are from the union of neighbors of the already sampled nodes, which preserves the inter-layer dependence. Overall, layer-wise sampling methods sample nodes per layer in a mini-batch manner to guarantee the scalability of training for large-scale graphs, achieving considerable efficiency compared with the original GCN in terms of training cost and speed.

2.3 Subgraph-wise Sampling

Subgraph-wise sampling is the type of methods which samples subgraphs for each mini-batch and use subgraphs to do the further training. There are two methods to generate subgraphs, one is directly formed by the graph partition algorithm and the other is induced from the specifically sampled nodes set. For the former case, we introduce **Cluster-GCN** [8], which first partitions the original graph into multiple clusters by using graph clustering algorithms (e.g., Metis [20] and Graclus [21]). Then, Cluster-GCN randomly samples a fixed number of clusters as a batch and forms a subgraph by combining the chosen clusters. Finally, the batch training of GCN is executed based on a subgraph in each iteration, which avoids the neighborhood searching outside the subgraph and ultimately reduces the training cost. More importantly, Cluster-GCN leverages a stochastic multiple clustering approach to address the imbalanced distribution of nodes' labels caused by the clustering algorithm (Metis). Although Cluster-GCN outperforms the previous works, it does not explicitly account for or solve the bias caused by the graph sampling.

For the subgraph which is progressively generated from a specifically sampled nodes set, we focus on **ShaDow-GNN** [19]. In this situation, one or several nodes are chosen as initial nodes. Based on the initial nodes, more nodes or edges are sampled by specific expansion and added to a candidate sampling set. Thereby, a subgraph can be induced from the candidate sampling set. Specially, for ShaDow-GNN k-hop sampler, starting from the target node v, the sampler traverses up to k hops. At a hop-l node u, the sampler will add to its hop-$(l+1)$ node set either all neighbors of u, or randomly selected neighbors of u. Then the subgraph is induced from all the nodes selected by the sampler. The key idea of ShaDow-GNN is applying shallow subgraph-based sampling methods to deep GCNs to guarantee the unique aggregation of any two nodes and preserve node feature information, which helps avoid the damage to the diversity of the converged aggregation features. Overall, in Cluster-GCN, graph clustering consumes a lot of time due to the complexity of the original graph. While in ShaDow-GNN, the sampling process is based on neighbor searching and selecting, so neighbor traversal causes non-trivial computation overhead.

3 PROPOSED METHODS

3.1 Challenges

The above-mentioned sampling methods are all based on homogeneous graph. Only a few recent works have paid attention to sampling for heterogeneous graphs (HetGNN [22] and HGSampling [12]). Multi-type relationships have interdependence among them, and a good sampled neighborhood should retain those information. Therefore, it is critical for a sampling method to carefully consider different types of nodes and determine the effect on the sampled results.

3.2 Homo2Hete Pipeline

In order to overcome the challenge, we initially propose a pipeline in which the heterogeneous graph is converted into a homogeneous one and using the existing samplers for homogeneous graphs. Specifically, we firstly partition the heterogeneous graph into relational subgraphs: each partition has and only has one relation type. At the same time, we transform the original heterogeneous graph into a full homogeneous graph by removing the edge types. Using this full homogeneous graph, we can apply the typical sampling methods for homogeneous graphs which are proved to have high performance on them. Using this sampled homogeneous graph, we map the sampling choices of nodes and edges to the relational subgraphs mentioned before, so in this way we can acquire the sampling result for the relational subgraphs. Then using these sampling results, we can train on each subgraph, and finally combine the results. This process is illustrated by Figure 1.

The methods for homogeneous sampling in Section 2 can be incorporated into this pipeline at the step where we perform sampling on the full homogeneous graph (from *G_Homo* to *sampled G_Homo* in the Figure 1). The sampler in this pipeline is for homogeneous sampling. However, information of the heterogeneity is lost during the conversion step, so the sampler cannot incorporate this part of information, leading to more information lose in the sampling step. Then, even if the step of combining the embeddings from the subgraphs is cleverly designed, it cannot retrieve the lost information, so this process may lead to unsatisfactory performance.

3.3 HeteSampling Pipeline

Inspired by Relational Graph Convolutional Networks (R-GCNs)[23], which is an exemplary work of GCN on heterogeneous graphs, we extend our idea. R-GCN aggregates the neighborhood information differently for different relation types, leading to different neural network weights for different relation types. Therefore, we propose to perform sampling on each relational subgraph before the relation-specific transformations, instead of applying homogeneous

Dataset	AIFB	MUTAG	BGS	AM
Entities	8,285	23,644	333,845	1,666,764
Relation Types	45	23	103	133
Edges	29,043	74,227	916,199	5,988,321
Labeled	176	340	146	1,000
Entity Types	7	5	27	7
Classes	4	2	2	11

Table 1: Number of entities, relations, edges and classes along with the number of labeled entities for each of the datasets. Labeled denotes the subset of entities that have labels and that are to be classified.

Datasets	Sample Methods	Acc (%)	Total Time (s)	Batch Time (s)
AIFB	Original R-GCN	95.83	39.1	0.52
	Layer-wise sampling	**97.22**	8.97	0.22
	Node-wise sampling	91.67	18.57	0.21
	Subgraph sampling (ShaDowK)	47.22	5.06	0.25
	Subgraph sampling (Cluster-GCN)	58.33	19.23	0.32
MUTAG	Original R-GCN	73.23	28.98	0.43
	Layer-wise sampling	**75.00**	6.99	0.11
	Node-wise sampling	69.12	65.26	0.11
	Subgraph sampling (ShaDowK)	66.18	2.23	0.22
	Subgraph sampling (Cluster-GCN)	67.65	6.76	0.11
BGS	Original R-GCN	83.10	36.58	0.35
	Layer-wise sampling	**96.55**	10.18	0.25
	Node-wise sampling	89.66	312.08	0.23
	Subgraph sampling (ShaDowK)	65.52	2.32	0.23
	Subgraph sampling (Cluster-GCN)	65.52	8.10	0.24
AM	Original R-GCN	**89.29**	149.4	2.31
	Layer-wise sampling	86.36	80.78	0.32
	Node-wise sampling	52.02	2423.15	0.25
	Subgraph sampling (ShaDowK)	23.23	17.31	0.13
	Subgraph sampling (Cluster-GCN)	53.03	56.22	0.21

Table 2: Comparison of different sampling methods with original R-GCN (Full-Batch) on heterogeneous datasets, including node-wise sampling [6], layer-wise sampling [17], and subgraph-wise sampling [8, 19], in terms of accuracy, total training time, and mean time for each epoch.

sampling on the full homogeneous graph and mapping the sampling results.

Specifically, in this pipeline, applying one sampler on each relational subgraph, these sampled subgraphs are concatenated into blocks. The blocks will then be used for training by applying one GCN on each relation and generate the final embeddings (in our implementation we are using the framework of R-GCN, but it can also be extended to other heterogeneous GNNs). Finally the embeddings acquired from the training of all relational subgraphs are combined into the final embeddings. This process is illustrated by Figure 2. In this process, since each relational subgraph only contain one type of relation, we can use samplers for homogeneous sampling on them. Therefore, the sampling methods in Section 2 can be incorporated into this pipeline at step (b) in Figure 2.

4 EXPERIMENTS

4.1 Datasets and Settings

In our experiments, we focus on **node classification** task for heterogeneous graphs. We evaluate our sampling methods on four heterogeneous datasets in Resource Description Framework (RDF) format [24]: AIFB, MUTAG, BGS, and AM. The statistics of the datasets as shown in Table 1. We follow [23]'s experiments setting to remove relations that were used to create entity labels: *employs and affiliatio*n for AIFB,

Datasets	Sample Methods	Sample Nums	Acc (%)	Total Time (s)	Batch Time (s)
AIFB	Layer-wise sampling	4	91.67	8.67	0.21
		8	94.44	8.34	0.20
		16	**97.22**	8.97	0.22
	Node-wise sampling	4	94.44	18.89	0.21
		8	94.44	19.39	0.22
		16	**97.22**	19.06	0.21
	ShadowK sampling	4	22.22	5.07	0.25
		8	38.89	5.07	0.25
		16	47.22	5.06	0.25
	Cluster-GCN pipeline	4	58.33	18.13	0.32
		8	58.33	19.23	0.32
		16	50.00	27.04	0.31
MUTAG	Layer-wise sampling	4	67.65	6.76	0.11
		8	72.06	6.89	0.11
		16	**75.00**	6.99	0.11
	Node-wise sampling	4	45.59	67.40	0.12
		8	67.65	67.08	0.11
		16	64.71	66.20	0.11
	ShadowK sampling	4	54.41	2.18	0.22
		8	33.82	2.20	0.22
		16	66.18	2.23	0.22
	Cluster-GCN pipeline	4	67.65	4.48	0.12
		8	66.18	3.89	0.12
		16	67.65	4.35	0.12

Table 3: Comparison of different sampling numbers on heterogeneous datasets, including node-wise sampling [6], layer-wise sampling [17], and subgraph-wise sampling [8, 19], in terms of accuracy, total training time, and mean time for each epoch.

isMutagenic for MUTAG, *hasLithogenesis* for BGS, and *objectCategory and material* for AM. Our experiments are all base on Deep Graph Library (DGL) [25], which is an easy-to-use, high performance and scalable Python package for deep learning on graphs.

4.2 Benchmark Model

Relational Graph Convolutional Networks (R-GCNs)[23] is the most classic work about heterogeneous graph neural networks, through introducing relation-specific transformations to address highly multi-relational data characteristic of realistic knowledge bases. Our sampling methods are based on the R-GCN model but also can extend to other Heterogeneous GNNs model easily.

4.3 Evaluation of different sampling methods on heterogeneous graphs

For the first experiment, we test four different sampling methods on four different datasets. The table 2 shows that Layer-wise sampling gets the highest accuracy on three different datasets. And for the BGS dataset, the Layer-wise sampling method get 13% higher accuracy. Also, the layer-wise sampling minimizes the training time by 3.31 times. Another thing that is worth noting is that node-wise sampling time will take longer time when the dataset size gets larger. At large dataset AM, the Node-wise sampling even spends $10x$ times more than the Original R-GCN methods. All other sampling methods speedup the training process. The two sub-graph sampling methods all suffer from low accuracy due to the information being missing during sub-sampling as a homogeneous graph.

4.4 Evaluation of different sampling numbers on heterogeneous graphs

The Table 3 and 4 shows the relationship between sample numbers and the accuracy. For the Layer-wise sampling method and Node-wise sampling method, increasing the sample number gives higher accuracy. Although increasing the sample number slightly increases the total training time. For all methods, sampling number only increase few training time. The most factor that decide the training time is the sampling methods.

Datasets	Sample Methods	Sample Nums	Acc (%)	Total Time (s)	Batch Time (s)
BGS	Layer-wise sampling	4	89.66	9.75	0.24
		8	93.10	9.70	0.24
		16	**96.55**	10.18	0.25
	Node-wise sampling	4	**96.55**	321.63	0.25
		8	93.10	313.62	0.24
		16	89.66	322.96	0.24
	ShadowK sampling	4	62.07	2.28	0.23
		8	44.83	2.32	0.23
		16	65.52	2.32	0.23
	Cluster-GCN pipeline	4	68.97	6.54	0.24
		8	62.07	6.95	0.25
		16	68.97	8.10	0.25
AM	Layer-wise sampling	4	74.24	59.18	0.32
		8	80.81	79.57	0.32
		16	**86.36**	80.78	0.32
	Node-wise sampling	4	52.02	2423.15	0.26
		8	43.43	2440.09	0.25
		16	53.54	2402.25	0.26
	ShadowK sampling	4	23.23	17.31	0.13
		8	3.03	19.26	0.14
		16	0.51	19.6	0.14
	Cluster-GCN pipeline	4	48.48	83.09	0.22
		8	47.98	99.96	0.22
		16	49.49	58.95	0.23

Table 4: Comparison of different sampling numbers on heterogeneous datasets, including node-wise sampling [6], layer-wise sampling [17], and subgraph-wise sampling [8, 19], in terms of accuracy, total training time, and mean time for each epoch.

Datasets	Sample Methods	Acc (%)	Total Time (s)	Batch Time (s)
OGBN-MAG	Original R-GCN	39.77	-	-
	Cluster R-GCN	37.32	-	-
	HGSampling	49.27	-	-
	NARS	52.09	26241	26.2
	Layer-sampling R-GCN	**46.86**	**800**	**8.61**

Table 5: Evaluation of different sampling methods on larger-scale heterogeneous graphs, in terms of accuracy, total training time, and mean time for each epoch. The first three lines of results are from the official leaderboard for ogbn-mag.

4.5 Evaluation of scalability on larger-scale heterogeneous graphs

Furthermore, we also evaluated the scalability of our sampling methods on larger-scale heterogeneous graph. In the table 5, we train the RGCN model with layer sampling method and Yu et. al methods [26] to compare the accuracy result and time result. The first two lines of results are from the official leaderboard of OGBN-MAG, indicating that the sampling effect of Cluster R-GCN is indeed not very good. And our layer sampling on R-GCN achieves better results than original R-GCN and Cluster R-GCN for about 7% and 9%, and at the same time achieves comparable performance to the current state-of-the-art method NARS, but takes less time. It shows that layer-sampling method have save about $30x$ times than the NARS, indicating that our method achieves good efficiency without compromising on the effectiveness on heterogeneous graphs.

5 DISCUSSION

5.1 Limitations

In this project, we find that for node-wise sampling and layer-wise sampling methods, it is easier to transfer from homogeneous graphs to heterogeneous graphs through our proposed pipeline. However, for subgraph-wise sampling methods, there are some limitations. Previous works (HGSampling [12] and GPT-GNN [27]) has shown impressive performance for heterogeneous sub-graph sampling, but our experiments come to a different conclusion. For Cluster-GCN in Homo2Hete pipeline, We speculate that poor performance is due to the forced conversion from a homogeneous graph to a heterogeneous graph, resulting in losing part of information. Even though the relationship can be preserved through mapping operation, to some extent, the subgraphs generated by graph partition and clustering algorithms on such graphs have broken the relationship on the original heterogeneous graph, and may further lose more information compared with node-wise and layer-wise sampling methods. The results of ClusterGCN (R-GCN aggr) and Full-batch R-GCN from the leaderboard [1] for ogbn-mag also verify our conjecture, consistent with our experimental results. Therefore, we reasonably deduce that for subgraph-wise sampling on heterogeneous graphs, the subgraphs generated by graph partition algorithms cannot achieve the desired results, unless design a subtle method of concating subgraphs, which will also be one of our future work.

5.2 Comparison to HGSampling

For another type of subgraph-wise sampling method, i.e., the subgraph is induced from the specifically sampled nodes set, which is also the method adopted by HGSampling. Although our experiments based on ShaDow-GNN are following this method, it does not achieve very good results. We speculate that it may be because the focus of ShaDow-GNN is to apply shallow subgraph-based sampling methods to deep GCNs to guarantee the unique aggregation of any two nodes and preserve node feature information, but does not consider the characteristics of heterogeneous graphs, i.e., multi-type nodes and edges, as well as problems with imbalanced neighbors' numbers in different types. HGSampling neatly solved these problems by targeting the properties of heterogeneous graphs. Specifically, in HGSampling, a subgraph is generated by repeating their Heterogeneous Mini-Batch Graph Sampling for L times. The induced subgraph with L depth is adequately dense due to the importance sampling and normalization technique, which is beneficial to variance reduction. For the imbalanced problem, HGSampling samples a similar number of nodes for each type to maintain a balanced sampling result. Furthermore, HGSampling adds the normalized degree of a node to its neighbors stored in corresponding budgets, which alleviates the impact of some high degree nodes. In this way, nodes with higher values in the corresponding budgets are given a higher probability of being sampled. In the future, we can study how further improve the performance of subgraph-wise sampling on heterogeneous graphs based on the design idea of HGSampling.

[1] https://ogb.stanford.edu/docs/leader_nodeprop/#ogbn-mag

6 CONCLUSION

In this paper, we conduct a comprehensive survey on sampling methods for Heterogeneous GNNs and propose two general pipelines for heterogeneous sampling to extend the sampling methods designed for homogeneous graphs. Specifically, based on the proposed pipeline, we evaluate 3 representative sampling methods by category on heterogeneous graphs, including node-wise sampling, layer-wise sampling, and subgraph-wise sampling. To the best of our knowledge, we are the first to provide a thorough implementation, evaluation, and discussion of each sampling method on heterogeneous graphs. Extensive experiments compared sampling methods from multiple aspects and highlight their characteristics for each category. Evaluation of scalability on larger-scale heterogeneous graphs also shows we achieve the trade-off between efficiency and effectiveness. Furthermore, we also analyze the limitations of our proposed pipeline on heterogeneous sub-graph sampling and provide a detailed comparison with HGSampling. In the future, we are expected to study how further improve the performance of subgraph-wise sampling on heterogeneous graphs based on the design idea of HGSampling, as well as improve the original sampling method by merging the characteristics of several sampling methods in different categories.

7 ACKNOWLEDGEMENTS

We would like to thank Yewen (Emily) Wang, Prof. Yizhou Sun and DGL community for helpful discussions and comments.

8 WORKLOAD DISTRIBUTION

- **Node-wise Sampling**: Yongqian Li;
- **Layer-wise Sampling**: Feiyang Chen;
- **Subgraph-wise Sampling (ShaDowKHopSampler)**: Feiyang Chen and YuanChing Lin;
- **Subgraph-wise Sampling (ClusterGCNSampler)**: Yongqian Li, YuanChing Lin, and Ruoyu He;
- **Write the report**: All members.

REFERENCES

[1] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[2] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2016.

[3] Dejun Jiang, Zhenxing Wu, Chang-Yu Hsieh, Guangyong Chen, Ben Liao, Zhe Wang, Chao Shen, Dongsheng Cao, Jian Wu, and Tingjun Hou. Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. *Journal of cheminformatics*, 13(1):1–23, 2021.

[4] Chen Gao, Xiang Wang, Xiangnan He, and Yong Li. Graph neural networks for recommender system. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1623–1625, 2022.

[5] Frederik Diehl, Thomas Brunner, Michael Truong Le, and Alois Knoll. Graph neural networks for modelling traffic participant interaction. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 695–701. IEEE, 2019.

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[7] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. Layer-dependent importance sampling for training deep and large graph convolutional networks. *Advances in neural information processing systems*, 32, 2019.

[8] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 257–266, 2019.

[9] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. GraphSAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020.

[10] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 793–803, New York, NY, USA, 2019. Association for Computing Machinery.

[11] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer, 2020.

[12] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, pages 2704–2710, 2020.

[13] Xin Liu, Mingyu Yan, Lei Deng, Guoqi Li, Xiaochun Ye, and Dongrui Fan. Sampling methods for efficient training of graph convolutional networks: A survey. *IEEE/CAA Journal of Automatica Sinica*, 9(2):205–234, 2021.

[14] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

[15] Hanjun Dai, Zornitsa Kozareva, Bo Dai, Alex Smola, and Le Song. Learning steady-states of iterative algorithms over graphs. In *International conference on machine learning*, pages 1106–1114. PMLR, 2018.

[16] Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction. *arXiv preprint arXiv:1710.10568*, 2017.

[17] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

[18] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph representation learning. *Advances in neural information processing systems*, 31, 2018.

[19] Hanqing Zeng, Muhan Zhang, Yinglong Xia, Ajitesh Srivastava, Andrey Malevich, Rajgopal Kannan, Viktor Prasanna, Long Jin, and Ren Chen. Deep graph neural networks with shallow subgraph samplers. *arXiv preprint arXiv:2012.01380*, 2020.

[20] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[21] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11):1944–1957, 2007.

[22] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 793–803, 2019.

[23] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[24] Petar Ristoski and Heiko Paulheim. Rdf2vec: Rdf graph embeddings for data mining. In *International Semantic Web Conference*, pages 498–514. Springer, 2016.

[25] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

[26] Lingfan Yu, Jiajun Shen, Jinyang Li, and Adam Lerer. Scalable graph neural networks for heterogeneous graphs, 2020.

[27] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1857–1867, 2020.