Accelerating Large-Scale Heterogeneous Interaction Graph Embedding Learning via Importance Sampling

YUGANG JI, Beijing University of Posts and Telecommunications
MINGYANG YIN, HONGXIA YANG, and JINGREN ZHOU, Alibaba Group
VINCENT W. ZHENG, Advanced Digital Sciences Center
CHUAN SHI, Beijing University of Posts and Telecommunications
YUAN FANG, Singapore Management University

In real-world problems, heterogeneous entities are often related to each other through multiple interactions, forming a Heterogeneous Interaction Graph (HIG). While modeling HIGs to deal with fundamental tasks, graph neural networks present an attractive opportunity that can make full use of the heterogeneity and rich semantic information by aggregating and propagating information from different types of neighborhoods. However, learning on such complex graphs, often with millions or billions of nodes, edges, and various attributes, could suffer from expensive time cost and high memory consumption. In this article, we attempt to accelerate representation learning on large-scale HIGs by adopting the importance sampling of heterogeneous neighborhoods in a batch-wise manner, which naturally fits with most batch-based optimizations. Distinct from traditional homogeneous strategies neglecting semantic types of nodes and edges, to handle the rich heterogeneous semantics within HIGs, we devise both type-dependent and type-fusion samplers where the former respectively samples neighborhoods of each type and the latter jointly samples from candidates of all types. Furthermore, to overcome the imbalance between the down-sampled and the original information, we respectively propose heterogeneous estimators including the self-normalized and the adaptive estimators to improve the robustness of our sampling strategies.

Finally, we evaluate the performance of our models for node classification and link prediction on five real-world datasets, respectively. The empirical results demonstrate that our approach performs significantly better than other state-of-the-art alternatives, and is able to reduce the number of edges in computation by up to 93%, the memory cost by up to 92% and the time cost by up to 86%.

CCS Concepts: • **Information systems → World Wide Web**; • **Computing methodologies → Machine learning**; **Artificial intelligence**;

Additional Key Words and Phrases: Heterogeneous interaction graphs, large-scale graphs, type-dependent sampler, type-fusion sampler, importance sampling

Both Chuan Shi and Yuan Fang are the co-corresponding authors.

This work is supported in part by the National Natural Science Foundation of China (No. 61772082, 61806020, and 61702296), the National Key Research and Development Program of China (2018YFB1402600), and Alibaba Group under its Alibaba Innovative Research (AIR) program. This research is also supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) program.

Authors' addresses: Y. Ji and C. Shi, Beijing University of Posts and Telecommunications, Xitucheng Road, Beijing, China; emails: {jiyugang, shichuan}@bupt.edu.cn; M. Yin, H. Yang, and J. Zhou, Alibaba Group, Wengyi Road, Hangzhou, China; emails: {hengyang.ymy, yang.yhx, jingren.zhou}@alibaba-inc.com; V. W. Zheng, Advanced Digital Sciences Center, Create Tower, Singapore, Singapore; email: vincent.zheng@adsc-create.edu.sg; Y. Fang, Singapore Management University, Victoria Street, Singapore, Singapore; email: yfang@smu.edu.sg.

1556-4681/2020/12-ART10 $15.00
https://doi.org/10.1145/3418684

ACM Reference format:
Yugang Ji, Mingyang Yin, Hongxia Yang, Jingren Zhou, Vincent W. Zheng, Chuan Shi, and Yuan Fang. 2020. Accelerating Large-Scale Heterogeneous Interaction Graph Embedding Learning via Importance Sampling. *ACM Trans. Knowl. Discov. Data* 15, 1, Article 10 (December 2020), 23 pages.
https://doi.org/10.1145/3418684

1 INTRODUCTION

Graphs are universal representations of pair-wise interactions. In real-world scenarios, heterogeneous entities are often related to each other through multiple interactions, forming a heterogeneous interaction graph (HIG) [4, 40, 42]. In Figure 1, taking the typical e-commerce interaction graph as an example, there are two types of entities, i.e., users and items, and four kinds of interactions among these entities including "click," "cart," "favorite," and "purchase," as well as several attributes or features on various interactions. Compared with traditional homogeneous graphs where the types of nodes and edges are neglected, HIGs are able to describe more abundant semantics by higher-order structures such as meta-paths [32] and meta graphs [11, 22]. Typically, HIGs are large scale, consisting of millions or billions of entities and their interactions.

Towards effectively modeling such complex graphs to address real-world problems like node classification and recommendation, graph embedding (or called graph representation learning), which projects a high-dimension sparse graph into a low-dimension space that preserves its structural information, has attracted more and more attention [2, 3, 20, 21, 37, 41]. In particular, Graph neural networks (GNNs) are a family of powerful graph representation learning approaches which reconstruct node representations by aggregating information from neighboring nodes. Due to their superior performance, GNNs have been widely studied on a wide range of fundamental problems in the real world. While previous works focus on homogeneous graphs with single-typed nodes and interactions [13, 23, 24, 34], recent research becomes aware of the abundant semantics on heterogeneous graphs and propose to reconstruct node representations from multiple types of neighborhoods [4, 35, 42]. However, when dealing with large-scale HIGs, these models generally suffer from expensive time complexities and heavy memory costs because of the huge number of neighbors and interactions.

To reduce the computational and memory costs on HIGs, one promising direction is to apply sampling on HIGs: intuitively, we could sample smaller but representative neighborhoods from a distribution that over-weighs the important regions, known as *Importance Sampling* (IS) [1, 7, 10, 26]. There are two crucial ingredients of IS [26], one of which is to design an effective sampling distribution, and the other is to adjust the estimator according to the sampled neighbors.

Challenges and Insights. More recently, to make large-scale graph representation learning possible, researchers have proposed several sampling strategies on GNNs, including node-wise neighborhood sampling [16, 39] and layer-wise neighborhood sampling [5, 19, 43] to accelerate the training process of GNNs by reducing the number of edges in computation. The former is to sample neighbors for each node while the latter is to sample neighbors from the whole graph. Unfortunately, both the node- and layer-wise sampling only deal with homogeneous graphs, which are inadequate in many real-world scenarios such as E-commerce graphs, as they (1) deal with homogeneous graphs only; (2) take the take all the nodes of a graph as initial candidates; and (3) still incur rapidly increasing cost with more layers.

One immediate question is, how to efficiently apply sampling strategies to large-scale HIGs? A naïve solution is to respectively sample smaller size of typed neighborhoods for each target node. However, this typically only works for smaller graphs. On very large graphs, training with such *node-wise* heterogeneous sampling becomes prohibitive due to the overhead associated with

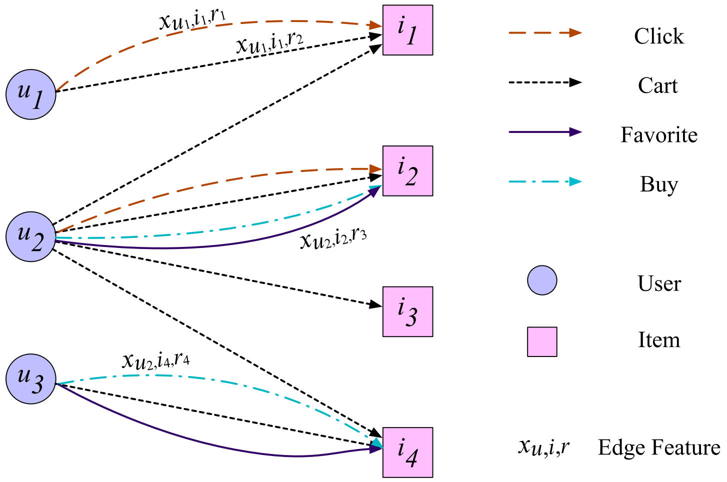


Fig. 1. A toy example of the HIG in e-commerce. There are four types of interactions including "click," "cart," "favorite," and "buy," and the corresponding edge feature $x_{i,j,r}$ between users and items. Notice that all edges contain the features while we just showcase some in this figure for the sake of brevity.

every node. Thus, we propose *batch-wise* heterogeneous sampling, which naturally fits with most optimization algorithms that utilize batch-based gradient updates. That is, we sample from the union of neighborhoods of all target nodes in a batch at the same time, reducing the overhead of node-wise sampling significantly. However, adopting batch-wise heterogeneous sampling faces two major challenges.

First, how to design an effective sampler that works with the heterogeneous neighborhoods? While a sampler can be easily defined on a per-node, per-type basis, it is not clear how we can sample neighborhoods in a batch, given that the candidates consist of different types of nodes. Two alternatives could work, namely, *type-dependent* and *type-fusion* sampling. In the former, a sampler is deployed for each neighbor type, and these type-based samplers sample from their respective sub-neighborhood on their own. In the latter, a single sampler is deployed, which treats the entire common neighborhood of the batch as its candidates. Intuitively, the type-fusion sampler would work better, as it takes into account the influences of total types of interactions and models the sampling distribution over all types jointly.

Second, how to design the corresponding effective estimators with the sampled heterogeneous neighborhoods? With the neighbors being sampled based on the global importance in a batch, traditional IS could introduce unwanted variance because of the imbalance between the local importance to the given target node and the global importance to the given batch. To address this challenge, we respectively propose *self-normalized estimators* and *adaptive estimators* where the former is to adjust the estimators by self-normalizing importance of sampled neighborhoods while the latter is to automatically learn the global importance (i.e., the importance distribution of candidate neighborhoods) by taking into account both structural and attributed information to ensure variance reduction.

Contributions. In this article, we identify the need to work with large-scale HIGs in real-world applications such as e-commerce platforms. Experimental results demonstrate that the proposed framework can achieve statistically significant improvements. Compared with the full model without any sampling, we achieve a memory cost reduction by up to 92.48%, a time cost reduction by up to 85.95%, and a reduction of edges in computation by up to 93.36% during training, while maintaining the same level of accuracy on the four real-world datasets. To summarize, we make the following contributions.

- To the best of our knowledge, this is the first work to accelerate large-scale HIG embedding learning via IS. While previous works focus on speeding up training on homogeneous networks, it is challenging to effectively sample neighborhoods when dealing with large-scale heterogeneous graphs because of the heterogeneity of edges and nodes.
- We design multiple IS strategies, including type-dependent and type-fusion sampling, for general heterogeneous GNNs that utilize batch-wise gradient updates. Moreover, to reduce the variance resulted from sampling, we respectively design self-normalized estimators and adaptive estimators to more effectively aggregate sampled information from the neighborhoods.
- We conduct extensive experiments on five real-world datasets, and evaluate the performance of node classification and link prediction. In particular, we have worked with an public large-scale Alibaba E-commerce graph on an important task of purchase prediction, and have achieved promising results.

2 RELATED WORK

In this section, we summarize the related work of three main aspects, including graph embedding, heterogeneous graph embedding and node sampling.

Graph Embedding. Graph embedding is to project graphs into low-dimensional vector spaces keeping the structure similarity. Previous works [9, 12, 15, 27, 33] attempt to preserve the neighborhood structure of nodes by using randomly sampled pair-wises. For instance, DeepWalk [27] uses random walks to keep the similarity of nodes in same sequences, whereas LINE [33] makes use of first-order and second-order proximity. These models are usually unsupervised without being tailored to any specific task. As a family of powerful graph representation learning approaches, GNNs [36], which aim to extend the deep neural networks to deal with graph structured data, have been widely used for graph embedding [14, 16, 23, 28, 34]. In particular, there is a surge of generalizing convolutional operations to graphs. Kipf et al. [23] have proposed the Graph Convolutional Network (GCN), which designs a GCN via a localized first-order approximation of spectral graph convolutions. Moreover, GAT [34] puts forward several attention mechanisms when learning node embedding. However, these models are limited for the requirement of the whole structural information. By considering the convolution process as information aggregation, Hamilton et al. [16] further extend the convolutional model to an inductive setting. In this model known as GraphSAGE, information is aggregated from neighbors repeatedly through one or more layers. The powerful technique of information aggregation from neighboring nodes through deep layers has been commonly used in GNNs. Unfortunately, when dealing with large-scale graphs, such strategies suffer from expensive memory and time costs because of a large number of neighbors and multiple layers. To keep low computational complexity when learning representation of nodes, Duran et al. [13] propose to propagate information from neighborhoods and keep the complexity linear with the scale of graphs.

Heterogeneous Graph Embedding. While current graph embedding methods mainly deal with homogeneous graphs that neglect the types of nodes and edges, they fail to preserve the abundant semantics when modeling real-world graphs consisting of different-typed nodes and edges [3, 31]. In order to keep the semantics as much as possible, heterogeneous graphs [30, 32] are proposed to model such complex data. Recently, embedding learning on heterogeneous graphs is more and more popular [4, 9, 12, 29]. The earlier Esmi [29] adopts multiple meta-paths [32] to extract various semantics between nodes and adopts factorization machine to learn node embedding of the pointed type. Metapath2vec [9] proposes the meta path-based random walks to generate type-wise sequences and designs the heterogeneous skip-gram to learn latent representation of nodes. HIN2vec [12] learns representation of both nodes and relation types by utilizing feed-forward neural networks. However, these methods mainly learn general representation of nodes while failing to directly address the supervised tasks. Recently, Wang et al. [35] proposes the heterogeneous attention networks (HAN) to construct node embedding by learning the weight of different meta-path-based information in a semi-supervised manner. To deal with link prediction on attributed multiplex heterogeneous graphs, Cen et al. [4] propose GATNE to aggregate information from multiple interactions rather than customized meta-paths. When dealing with larger graphs, GATNE has to fix the number of neighborhoods so as to make training possible. Inspired by embedding propagation [13], Zheng et al. [42] and Yang et al. [38] design embedding propagation mechanisms on heterogeneous graphs to ensure the computational complexity linear with the size of nodes and edges.

Node Sampling. Node sampling has been widely used to work with very large graphs to approximate solutions efficiently. There are three broad categories of sampling: (1) extract a smaller dataset offline, often through performing random walks [9, 12, 15, 27]; (2) sample representative nodes, including positive nodes and negative neighbors [6, 33]; and (3) sample nodes to better represent the global information, including node-wise sampling [16, 39] and layer-wise [5, 19, 43] sampling. Recent studies mostly belong to the third category. GraphSAGE [16] is an inductive model which gathers neighborhood information to construct embeddings of the target nodes. To avoid the high memory and time cost of GCNs [8], GraphSAGE [16] attempts to sample neighbors with a node-wise sampling strategy rather than using the full neighborhood of each node. PinSage [39] also adopts node-wise sampling to sample the neighborhood around each node and dynamically constructs a computation graph from the sampled neighborhood. However, such node-wise sampling often comes with a significant computational overhead associated with each node. Instead, layer-wise sampling is advantageous in deep models with multiple layers, where the combined neighborhood of all target nodes in a layer is sampled in one go. FastGCN [5] pays attention to the global structural information and proposed a layer-wise node sampling strategy to sample important neighbors based on the edges between the candidates and their linked target nodes. To avoid the sparse of sampled nodes, LADIES [43] samples nodes with layer-dependent samplers where lower-layer candidates should be connected with higher-layer target nodes. Focusing on the variance reduction, AS-GCN [19] further designs an adaptive layer-wise sampling which optimizes node classification and reduces variance at the same time. All FastGCN, LADIES and AS-GCN build upon IS, which is a widely used technique to reduce the variance of a Monte Carlo estimator by an appropriate change of measure [17, 26]. However, the complexity of these methods still increases obviously when larger layers are used. Furthermore, all these sampling strategies are proposed for homogeneous networks and cannot be directly adopted on heterogeneous graphs with different types of nodes and edges.

3 PRELIMINARIES

In this section, we first introduce related concepts including HIG and IS. The main notations are summarized in Table 1.

Definition 1 (Heterogeneous Interaction Graph (HIG)). A heterogeneous interaction graph is $G = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}, \mathcal{X}, \phi, \psi)$, where $\mathcal{V}$ is a set of nodes with types, $\mathcal{E}$ is the set of edges among these nodes, $\mathcal{T} = \{t_1, t_2, \ldots, t_{|\mathcal{T}|}\}$ are the distinct node types, $\mathcal{R} = \{r_1, r_2, \ldots, r_{|\mathcal{R}|}\}$ represents the distinct edge types, $\mathcal{X}$ is the set of edge features $\{\mathcal{X}_r | r \in \mathcal{R}\}$ such that $\mathcal{X}_r$ is the d_r-dimension features of type-r

Table 1. Notations

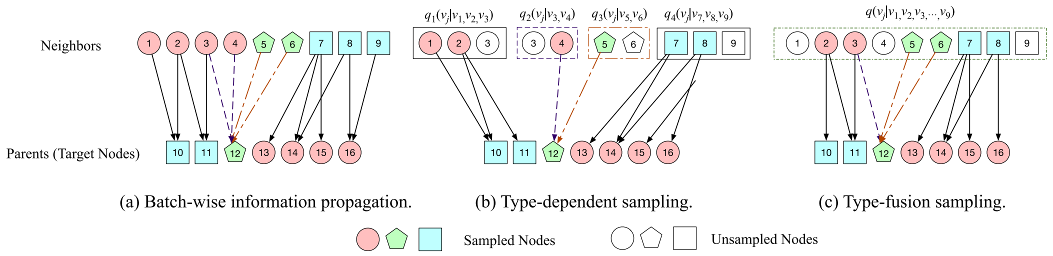
Symbols	Descriptions
$G = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{X})$	The given HIG.
d	The dimension of embedding vector.
$\boldsymbol{B}_k$	The set of nodes in k^{th} mini-batch.
v_i	The i-th node in G.
$\hat{v}_i$	The i-th sampled node.
$\phi(v_i)$	The type of v_i.
$\psi(v_i, v_j)$	The edge type between v_i and v_j.
$p(v_j \mid v_i)$	The weight of v_j to v_i.
$q(v_j \mid \cdot)$	The sampling probability of v_j.
$\boldsymbol{x}_{v_i,v_j,r} \in \mathbb{R}^{d_r}$	The type-r edge feature vector of d dimensions between v_i and v_j.
$\boldsymbol{\lambda}_r \in \mathbb{R}^{d_r \times 1}$	The weight of type-r edges.
b_r	The bias of type-r edges.
$\boldsymbol{g}'_{v_i,r} \in \mathbb{R}^d$	The propagated information of type-r neighborhoods.
$\tilde{\boldsymbol{g}}_{v_i,r} \in \mathbb{R}^d$	The estimated $\boldsymbol{g}'_{v_i,r}$.
$\boldsymbol{h}_i \in \mathbb{R}^d$	The embedding vector of v_i.
$\tilde{\boldsymbol{h}}_i \in \mathbb{R}^d$	The reconstructed $\boldsymbol{h}_i$.
$\boldsymbol{H} \in \mathbb{R}^{\lvert\mathcal{V}\rvert \times d}$	The node embedding matrix.
$\boldsymbol{W}_t \in \mathbb{R}^{\lvert\mathcal{R}\rvert d \times d}$	The type-t embedding weight matrix.
$\mathcal{N}_{v_i,r}, \hat{\mathcal{N}}_{v_i,r}$	The type-r neighborhoods and the sampled type-r neighborhoods of v_i.
$\mathcal{N}_{v_i}, \hat{\mathcal{N}}_{v_i}$	The neighborhoods and the sampled neighborhoods of v_i.

edges, $\phi : \mathcal{V} \rightarrow \mathcal{T}$ is the node type mapping function to return the types of given nodes, while $\psi : \mathcal{E} \rightarrow \mathcal{R}$ is the edge type mapping function to return the types of given edges.

For instance, as shown in Figure 1, there are two types of nodes, namely, users (U) and items (I), and four types of interactions among these nodes, namely, "click," "cart," "favorite," and "buy," as well as several attributes within these interactions like the stay time and the frequency of actions. Notice that different from traditional heterogeneous graphs (or called heterogeneous information networks) [32, 35] mainly pay attention to the heterogeneity of nodes, we can consider HIGs as the special heterogeneous graphs which not only take into account types of nodes but also capture the multiple interactions and edge features.

Definition 2 (Heterogeneous Interaction Graph Embedding (HIGE)). Given a heterogeneous interaction graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}, \mathcal{X}, \phi, \psi)$ defined in Definition 1, the goal of HIGE is to learn a project function $\mathcal{H} : \mathcal{H}(v_i, \phi(v_i)) \rightarrow \boldsymbol{h}_i, \boldsymbol{h}_i \in \mathbb{R}^d$ and $d \ll |\mathcal{V}|$ with the assumption that the embeddings of nodes and their neighborhoods should be as similar as possible [4, 29, 35].

To address the problem of HIGE, previous works [9, 12] adopt meta-path-based random walks to sample similar nodes and input them into deep neural networks. Recently, with the rapid development of GNNs, it becomes popular to reconstruct node embedding by aggregating information of heterogeneous neighborhoods, and then keep the similarity of node embedding and its reconstructed embedding based on heterogeneous attention mechanisms [4, 35] or the direct Euclidean distance [42]. Taking Figure 1 as an example, given user u_2 and the interacted items (i.e., i_1, i_2, i_3, and i_4), we respectively aggregate information from items for each type of interactions and then reconstruct the embedding of u_2 based on the aggregated information. The details of heterogeneous



(a) Batch-wise information propagation.

(b) Type-dependent sampling.

(c) Type-fusion sampling.

Fig. 2. The process of sampling neighbors including type-dependent sampling and type-based sampling. (a) is the batch-wise information propagation/aggregation to reconstruct node embedding based on multiple neighbors, (b) is the type-dependent sampling to sample neighbors for each type of interactions, and (c) is the type-fusion sampling to sample neighbors from the union candidates taking all interactions into consideration.

information aggregation are described in Section 4.1. Actually, such aggregation could suffer from the expansive computational complexity and memory cost when dealing with large-scale graphs.

Definition 3 (Importance Sampling (IS) [26]). Importance sampling is to sample important nodes from a distribution that over-weighs the important regions. Suppose that our problem is to aggregate information $\mu = \sum_{j=1}^{N} p(v_j) f(v_j)$ where N is the number of nodes, $p(v_j) \in \mathbb{R}_+$ is the weight of v_j and $f(v_j)$ denotes the information of node v_j, importance sampling is to sample n important nodes subject to distribution q, and ensures the gathered information $\mathbb{E}(\hat{\mu}_q) = \mu$. In this article, we can consider the edge weight of neighbor v_j as $p(v_j)$ and set embedding of v_j as the information $f(v_j)$.

Taking Figure 2(b) as an example, given the neighbors v_3, v_4, v_5, and v_6 to propagate information with edge weights to their target node v_{12}, IS firstly measures the importance of these neighbors based on their edge weights $p(v)$ and the propagated information $f(v)$, and then samples the more effective neighbors rather than the whole neighbors.

In particular, μ can be re-formulated as the expectation over the sampling distribution q:

$$\mu = \sum_{j=1}^{N} p(v_j) f(v_j) = \sum_{j=1}^{N} \frac{p(v_j) f(v_j)}{q(v_j)} q(v_j) = \mathbb{E}_q \left[\frac{p(v_j)}{q(v_j)} f(v_j) \right]. \tag{1}$$

Therefore, the IS estimator of μ is

$$\hat{\mu}_q = \frac{1}{n} \sum_{j} \frac{p(\hat{v}_j)}{q(\hat{v}_j)} f(\hat{v}_j), \quad \hat{v}_j \sim q, \tag{2}$$

where n denotes the number of sampled neighbors, and $\hat{v}_j$ is a sampled neighbor from q. By setting $n \ll N$, the computational sources are saved actually. While node-wise sampling utilizes $q(v_j|v_i)$ to sample neighborhoods of v_i with $\sum_{v_j} q(v_j|v_i) = 1$, layer-wise sampling is to sample information from the union candidates with $q(v_j|\mathcal{V})$, and $\sum_{v_j} q(v_j|\mathcal{V}) = 1$.

4 THE PROPOSED METHOD

In this section, we firstly introduce the general HIGE method which aggregates information from different types of neighborhoods. Focusing on accelerating HIGE on large-scale graphs, we then propose heterogeneous sampling strategies including type-dependent and type-fusion samplers to overcome the expensive time cost and high computational complexity.

4.1 The General HIGE Model

To tackle with HIGs, a general idea is to reconstruct node embedding by propagating information from its heterogeneous neighbors, and backwardly propagate gradients [4, 35, 42], as shown in Figure 2(a). Specifically, given a node v_i of type $\phi(v_i)$ and its edges $\{e_{v_i,v_j,r} | v_j \in \mathcal{N}_{v_i,r}, r \in \mathcal{R}\}$, the aggregated information of node v_i from type-r neighborhoods is:

$$\boldsymbol{g}'_{v_i,r} = \sum_{v_j \in \mathcal{N}_{v_i,r}} \frac{1}{|\mathcal{N}_{v_i,r}|} w(v_i, v_j, r) \boldsymbol{h}_{v_j}, \tag{3}$$

where $\boldsymbol{g}'_{v_i,r}$ is the aggregated information, $\mathcal{N}_{v_i,r}$ is the type-r neighborhoods of node v_i, $v_j \in \mathcal{N}_{v_i,r}$ is a specific type-r neighbor of node v_i, $w(v_i, v_j, r)$ denotes the weight of edge $e_{v_i,v_j,r}$, $\boldsymbol{h}_{v_j} \in \mathbb{R}^d$ denotes the embedding of node v_j, d is the dimension of $\boldsymbol{h}_{v_j}$. Notice that, $w(v_i, v_j, r)$ between node v_i and v_j is calculated by:

$$w(v_i, v_j, r) = \sigma(\boldsymbol{x}_{v_i,v_j,r} \boldsymbol{\lambda}_r + b_r), \tag{4}$$

where σ is an activation function, $\boldsymbol{x}_{v_i,v_j,r} \in \mathbb{R}^{d_r}$ is the edge features between node v_i and v_j, $r = \psi(i, j)$ is the edge type, d_r is the length of type-r edge features, $\psi(\cdot)$ is the edge type mapping function, $\boldsymbol{\lambda}_r \in \mathbb{R}^{d_r \times 1}$ and $b_r \in \mathbb{R}$ are the weight and bias parameters shared by type-r edges.

And then, the reconstructed embedding of node v_i is calculated by:

$$\boldsymbol{h}'_{v_i} = \sigma\left(concat(\boldsymbol{g}'_{v_i,r_0}, \boldsymbol{g}'_{v_i,r_1}, \cdots, \boldsymbol{g}'_{v_i,r_{|\mathcal{R}|}}) \boldsymbol{W}_{\phi(v_i)} + b_{\phi(v_i)} \right), \tag{5}$$

where $\boldsymbol{h}'_{v_i} \in \mathbb{R}^d$ is the reconstructed embedding, σ is an activation function, $concat(\cdot)$ is the concatenation option, $\boldsymbol{W}_{\phi(v_i)} \in \mathbb{R}^{|\mathcal{R}|d \times d}$ denotes the projection matrix of type-$\phi(v_i)$ nodes, $b_{\phi(v_i)}$ denotes the bias and $|\mathcal{R}|$ denotes the total number of edge types $\mathcal{R}$ in the heterogeneous graph. Obviously, the computational complexity of the general model is linear with the scale of edges and nodes on heterogeneous graphs.

4.2 Batch-Wise Heterogeneous Sampling

To reduce computational overhead and memory cost of the training process, a naïve idea is to sample several neighbors rather than aggregating information from all neighborhoods. Previous works [16, 39] on homogeneous works usually adopt node-wise sampling to sample several neighbors per node for learning. Paying attention to the global and local structural information, current works [5, 19, 43] propose layer-wise sampling strategies which consider the whole neighborhood as the candidates. However, these strategies are to deal with homogeneous graphs which ignore the abundant semantics of heterogeneous nodes and edges. Moreover, these strategies can only deal with smaller graphs because the overhead for node-wise sampling and layer-wise sampling during optimization could be quite expensive, or even unaffordable. In this section, by re-writing Equation (3) as:

$$\begin{aligned} \boldsymbol{g}'_{v_i,r} &= \sum_{v_j \in \mathcal{N}_{v_i,r}} p(v_j | v_i, r) w(v_i, v_j, r) \boldsymbol{h}_{v_j} \\ &= \sum_{v_j \in \mathcal{N}_{v_i,r}} q(v_j | \boldsymbol{B}_k) \frac{p(v_j | v_i, r)}{q(v_j | \boldsymbol{B}_k)} w(v_i, v_j, r) \boldsymbol{h}_{v_j} \\ &= \mathbb{E}_q \left[\frac{p(v_j | v_i, r)}{q(v_j | \boldsymbol{B}_k)} w(v_i, v_j, r) \boldsymbol{h}_{v_j} \right], \end{aligned} \tag{6}$$

where $\boldsymbol{B}_k$ denotes the target nodes in the k^{th} mini-batch, $q(v_j | \boldsymbol{B}_k)$ denotes the corresponding sampling probability in the k^{th} mini-batch, or in other words, the importance in this batch, and

$p(v_j|v_i, r)$ equals to $\frac{1}{|\mathcal{N}_{v_i,r}|}$, we attempt to make heterogeneous sampling for each batch, or in other words, batch-wise heterogeneous sampling with probability distribution

$$\hat{v_j} \sim q(\hat{v_j}|v_1, v_2, \ldots, v_{|\boldsymbol{B}_k|}) \quad v_j \in \{\mathcal{N}_{v_i,r}|r \in \mathcal{R}, v_i \in \boldsymbol{B}_k\}, \tag{7}$$

where $\hat{v_j}$ is the sampled neighbor. As shown in Figure 2(a), we sample instances from the union neighborhood of all the target nodes in a batch, where the union neighborhood denotes the union of the individual neighbors of each target node. Since sampling is now done for each batch instead of each target node, batch-wise sampling is often a good choice to reduce computational overhead and memory cost of the training process. Compared with layer-wise samplings [5, 19, 43] which require the total nodes as candidates and have to load the whole graph structure before training, our batch-wise heterogeneous sampling focus on union neighborhoods in the current batch. Compared with node-wise sampling which sample neighborhoods for each target node, our batch-wise heterogeneous sampling contains the advantages of reducing the overhead of sampling.

4.3 Type-Dependent Sampling Strategy

By now, we have defined the general batch-wise heterogeneous sampling. Different from traditional IS, the neighborhoods of each batch connect with target nodes based on different-typed interactions. A straightforward idea is to respectively design a sampler for each type. For example, in Figure 2(b), there are four types of candidates, and we respectively sample neighbors from each type of candidates by using one sampler. By adopting type-dependent sampling, we consider $q(v_j|v_1, v_2, \ldots, v_{|\boldsymbol{B}_k|})$ as a set of $\{q_r(v_j|\cdot)|r \in \mathbb{R}\}$, and sample type-$r$ neighborhoods with the corresponding sampler $q_r(v_j|\cdot)$. The remaining question for type-dependent sampling is how to design the exact form of this sampler so as to keep low variance for efficient training. Here we define the average information of $\boldsymbol{g}'_{i,r}$ as:

$$\begin{aligned} \boldsymbol{\mu}_{q_r} = \frac{1}{|\boldsymbol{B}_k|} \sum_{v_i \in \boldsymbol{B}_k} \boldsymbol{g}'_{v_i,r} = \frac{1}{|\boldsymbol{B}_k|} \sum_{v_j} q_r(v_j|\cdot) \\ \times \sum_{v_i \in \boldsymbol{B}_k} \frac{p(v_j|v_i, r) w(v_i, v_j, r) \boldsymbol{h}_{v_j}}{q_r(v_j|\cdot)} \quad v_j \in \{\mathcal{N}_{v_i,r}|r \in \mathcal{R}, v_i \in \boldsymbol{B}_k\}, \end{aligned} \tag{8}$$

where $\boldsymbol{\mu}_{q_r}$ is the average information by utilizing the type-dependent sampler q_r, $|\boldsymbol{B}_k|$ is the number of type-r samples. Then, the variance Var_{q_r} of $\boldsymbol{\mu}_{q_r}$ is calculated by:

$$\begin{aligned} Var_{q_r}(\boldsymbol{\mu}_{q_r}) &= \frac{1}{|\boldsymbol{B}_k|} \sum_{v_j} q(v_j|\cdot) \left[\boldsymbol{\mu}_{q_r} - \sum_{v_i \in \boldsymbol{B}_k} \frac{p(v_j|v_i, r) w(v_i, v_j, r) \boldsymbol{h}_{v_i}}{q_r(v_j|\cdot)} \right]^2 \\ &= \frac{1}{|\boldsymbol{B}_k|} \mathbb{E}_{q_r} \frac{\left[\boldsymbol{\mu}_{q_r} q_r(v_j|\cdot) - \sum_{v_i \in \boldsymbol{B}_k} p(v_j|v_i, r) w(v_i, v_j, r) \boldsymbol{h}_{v_i} \right]^2}{q_r(v_j|\cdot)^2}. \end{aligned} \tag{9}$$

Obviously, to ensure minimizing the variance, a better sampler is shown as follows:

$$q_r(v_j) = \frac{\sum_{v_i \in \boldsymbol{B}_k} p(v_j|v_i, r) |w(v_i, v_j, r) \boldsymbol{h}_{v_i}|}{\sum_{v_j} \sum_{v_i \in \boldsymbol{B}_k} p(v_j|v_i, r) |w(v_i, v_j, r) \boldsymbol{h}_{v_i}|}, \tag{10}$$

where $q_r(v_j)$ is the abbreviation of $q_r(v_j|v_1, v_2, \ldots, v_{|\boldsymbol{B}_k|})$ and $q_r(v_j|\cdot)$. We can intuitively understand the component $p(v_j|v_i, r)|w(v_i, v_j, r)\boldsymbol{h}_{v_i}|$ as the importance of v_j, where $|w(v_i, v_j, r)\boldsymbol{h}_{v_i}|$ denotes L1-norm value of $w_{v_i,v_j,r}\boldsymbol{h}_j$. A drawback of defining the sampling distribution $q_r(v_j)$ in this manner is that it involves $|w_{v_i,v_j,r}\boldsymbol{h}_j|$, which is constantly updated during training. As a compromise, similar to that in [5], we approximate $|w_{v_i,v_j,r}\boldsymbol{h}_j|$ with $p(v_j|v_i, r)$ as well in the sampler

$q_r(v_j)$, utilizing only the structural information of v_j as its importance:

$$q_r(v_j) = \frac{\sum_{v_i \in \boldsymbol{B}_k} p(v_j|v_i, r)^2}{\sum_{v_j} \sum_{v_i \in \boldsymbol{B}_k} p(v_j|v_i, r)^2}. \tag{11}$$

4.4 Type-Fusion Sampling Strategy

Essentially, the type-dependent strategy pays attention to the influence of same-type neighborhoods, without jointly considering the effect of heterogeneous types. Thus, they only reduce the individual variance of each type, lacking a global picture on the overall variance of the batch. Moreover, neighbors which contain multiple interactions may be not sampled. To address this weakness, we propose a type-fusion sampling strategy which considers the entire neighborhoods as the candidates, and sample neighborhoods with the same sampler in each batch. Compared with type-dependent sampling in Figure 2(b), the type-fusion sampling in Figure 2(c) can sample node v_3 which contains two types of interactions.

By dividing the parameter $\boldsymbol{W}_{\phi(v_i)}$ based on relations, we can rewrite the concatenation of $\{\boldsymbol{g}'_{v_i,r}|r \in \mathcal{R}\}$ in Equation (5) as:

$$\boldsymbol{g}'_{v_i} = \sum_{r \in \mathcal{R}} \sum_{v_j \in \mathcal{N}_{v_i,r}} p(v_j|v_i, r) w(v_i, v_j, r) \boldsymbol{h}_{v_j} \boldsymbol{W}_r, \tag{12}$$

where $\boldsymbol{W}_r \in \mathbb{R}^{d \times d}$ denotes the relation-wise projection matrix. Under the type-fusion strategy, the average information in k-th batch over all types is given by:

$$\begin{aligned} \boldsymbol{\mu}_q = \frac{1}{|\boldsymbol{B}_k|} \sum_{v_i \in \boldsymbol{B}_k} \boldsymbol{g}'_{v_i} &= \frac{1}{|\boldsymbol{B}_k|} \sum_{v_j} q(v_j|\cdot) \\ &\times \sum_{r \in \mathcal{R}} \sum_{v_i \in \boldsymbol{B}_k} \frac{p(v_j|v_i, r) w(v_i, v_j, r) \boldsymbol{h}_{v_j} W_r}{q(v_j|\cdot)} \quad v_j \in \{\mathcal{N}_{v_i,r}|r \in \mathcal{R}, v_i \in \boldsymbol{B}_k\}, \end{aligned} \tag{13}$$

This formulation assumes that all candidates are sampled by the same sampler, i.e., $q(v_j|v_1, v_2, \ldots, v_{|\boldsymbol{B}_k|})$. As shown in Figure 2(c), the different-typed neighborhoods are sampled according to the type-fusion sampling distribution. Similar to the type-dependent sampler in Section 4.3, the optimal sampler to minimize the variance with only structural information can be defined as follows:

$$q_s(v_j) = \frac{\sum_{r \in \mathcal{R}} \sum_{v_i} p(v_j|v_i, r)^2}{\sum_{r \in \mathcal{R}} \sum_{v_j} \sum_{v_i \in \boldsymbol{B}_k} p(v_j|v_i, r)^2}, \tag{14}$$

where $q_s(v_j)$ is the structure-based type-fusion sampler.

While the above sampling strategies only consider link-based weight as the importance of nodes but ignoring the attributed-based importance on edges, we further pay attention to the edge features within HIGs and propose to measure the importance of nodes with both structural and attributed information like ratings and number of fans. Considering the edges of different types contain different features, here we put forward a learnable type-fusion sampler which measures the importance of heterogeneous neighborhoods with latent parameters and sample representative candidates during optimization, namely,

$$q_a(v_j) = \frac{\sum_{r \in \mathcal{R}} \sum_{v_i} p(v_j|v_i, r) f(\boldsymbol{x}(v_i, v_j, r))}{\sum_{r \in \mathcal{R}} \sum_{v_j} \sum_{v_i \in \boldsymbol{B}_k} p(v_j|v_i, r) f(\boldsymbol{x}(v_i, v_j, r))}, \tag{15}$$

where $q_a(v_j)$ denotes the adaptive sampler, $f(\boldsymbol{x}(v_i, v_j, r))$ denotes the importance of edge features which is calculated by $\sigma(\boldsymbol{x}(v_i, v_j, r)\boldsymbol{W}_{x,r})$ and $\boldsymbol{W}_{x,r} \in \mathbb{R}^{d_r \times 1}$ is the type-r latent parameter need learn.

4.5 Heterogeneous Self-Normalized and Adaptive Estimators

In type-dependent or type-fusion strategies, the estimator $\hat{\boldsymbol{g}}_{v_i,r}$ is weighted by $\frac{p(\hat{v}_j|\cdot)}{q(\hat{v}_j|\cdot)}$, and the batch-wise IS estimator can be calculated as:

$$\hat{\boldsymbol{g}}_{v_i} = \frac{1}{n} \sum_{r \in \mathcal{R}} \sum_{v_j \in \hat{\mathcal{N}}_{B_k,r}} \frac{p(\hat{v}_j|\cdot)}{q(\hat{v}_j|\cdot)} w(v_i, \hat{v}_j, r) \boldsymbol{h}_{\hat{v}_j} \boldsymbol{W}_r, \tag{16}$$

where $\hat{\boldsymbol{g}}_{v_i}$ is the approximated information, n is the number of samples, $\hat{\mathcal{N}}_{B_k,r}$ is the sampled type-r neighborhoods in this batch, $p(\hat{v}_j|\cdot)$ equals to $\frac{1}{|\mathcal{N}_{v_i,r}|}$ and $q(\hat{v}_j|\cdot)$ denotes the heterogeneous samplers, such as q_r, q_s, and q_a. However, this could increase the variance resulted from the imbalance of $p(\hat{v}_j|\cdot)$ and $q(\hat{v}_j|\cdot)$. The former is a "local" weight based on each target node's own neighborhoods, whereas the latter is the batch-wise "global" sampling probability. In general, the union neighborhoods of a batch is much larger than the individual neighborhoods of a single target node.

To address such problem, for structure-based $q_s(v_j)$, a promising way is to balance the weights based on self-normalized importance, similar to that in [26]. The corresponding estimator can be computed as:

$$\hat{\boldsymbol{g}}_{sn,v_i} = \sum_{r \in \mathcal{R}} \sum_{\hat{v}_j \in \hat{\mathcal{N}}_{v_i,r}} \frac{\pi(\hat{v}_j)}{\sum_{\hat{v}'_j \in \hat{\mathcal{N}}_{v_i,r}} \pi(\hat{v}'_j)} w_{v_i,v_j,r} \boldsymbol{h}_j \boldsymbol{W}_r, \tag{17}$$

where $\hat{\boldsymbol{g}}_{sn,v_i}$ denotes the self-normalized information, $\pi(v_j) = \frac{p(v_j|\cdot)}{q(v_j|\cdot)}$, $\hat{\mathcal{N}}_r$ is the sampled neighborhoods of type r.

Besides, for the adaptive sampler $q_a(v_j)$, we estimate the reconstructed $\hat{\boldsymbol{g}}_{v_i}$ the same as Equation (16). Considering the adaptive sampler $q_a(v_j)$ not necessarily results in a minimal variance, we add the variance to the loss function and explicitly minimize the variance by model training, to fulfill variance reduction. The variance from adaptive samplers is $Var_{q_a}(\hat{\boldsymbol{\mu}}_q)$ where $\hat{\boldsymbol{\mu}}_q = \frac{1}{|\boldsymbol{B}_k|} \sum_{v_i \in \boldsymbol{B}_k} \hat{\boldsymbol{g}}_{v_i}$.

4.6 Computational Complexity Analysis

Considering the main computational cost of our models is from embedding propagation/aggregation, here we analyze the time complexity and memory cost of such a process of both the general HIGE and the variants utilizing heterogeneous sampling strategies. Given the HIG $\mathcal{G}$, the computational complexity of the general HIGE is $O(N_{neg}|\mathcal{V}|d + |\mathcal{E}|d^2)$ and the corresponding memory cost is $O(|\mathcal{V}|d + |\mathcal{R}|d^2 + |\mathcal{E}|d)$, where N_{neg} denotes the number of negative samples of each node, and it will increase with the scale of $\mathcal{G}$. For VarR-TD based variants, the computational complexity is $O(N_{neg}|\mathcal{V}|d + \sum_{r \in \mathcal{R}} |\hat{e}_r|d^2)$ and the memory cost is $O(|\mathcal{V}|d + |\mathcal{R}|d^2 + \sum_{r \in \mathcal{R}} |\hat{e}_r|d)$. For VarR-TF, VarR-TF-SN and VarR-TF-AS variants, the complexity is associated with $O(N_{neg}|\mathcal{V}|d + |\hat{e}|d^2)$ and the memory cost is $O(|\mathcal{V}|d + |\mathcal{R}|d^2 + |\hat{e}|d)$. Compared with the GCN-based models including Fast-GCN and AS-GCN where the computational complexity is related to the number of layers [43], both the general HIGE and its variants with sampling just propagate information between nodes where the complexity is linear with the scale of $\mathcal{G}$. Compared with the general HIGE, the main complexity and memory reduction of sampling-based variants is associated with the number of sampled edges, i.e., $|\hat{e}_r|$ and $|\hat{e}|$. With a small sample size n (or n_r), the coefficient $|\hat{e}|$ (or $|\hat{e}_r|$) can be much smaller than the total number of edges $|\mathcal{E}|$, namely, $|\hat{e}| \ll |\mathcal{E}|$. Therefore, our heterogeneous sampling strategies can bring in a significant drop of the computational and memory cost.

Notice that it may be useless if the number of samples is too small since the propagation will be quite ineffective. The problem of effective sample size is meaningful enough, where the common solution is the ratio of the variances of the estimators $N\frac{Var_p(\boldsymbol{\mu})}{Var_q(\boldsymbol{\mu})}$ [25]. In this article, since the effective sample size of each batch may be not stable, we set the sample size n (or n_r) as the reasonable fixed number and analyze the extreme cases, i.e., $n = 0$ (or $n_r = 0$) and $n = N_B$ (or $n_r = N_{B,r}$) where N_B (or $N_{B,r}$) is the total number of batch-wise neighbors (or type-r neighbors) in experiments.

4.7 Optimization Framework

To integrate the proposed batch-wise sampling strategies to large-scale HIGE learning, the overall loss function consists of four parts, namely, the loss of a specific task, the reconstruction loss of the embedding propagation with sampling, the variance of sampled information and the regularization of latent parameters, which are shown as follows:

$$L_k = L_{task,k} + \alpha L_{ep,k} + \beta\Omega(\boldsymbol{\Theta}) + \xi Var_{q_a,k}(\hat{\mu}_q), \tag{18}$$

where L_k is the loss value in k-th batch, $L_{task,k}$ is the loss from supervised learning in the same batch, $L_{ep,k}$ is the embedding propagation loss with sampling in the k-th batch, $Var_{q_a,k}(\hat{\mu}_q)$ is the variance of sampled information, $\Omega(\boldsymbol{\Theta})$ is the regularization of all latent parameters, and α, β and ξ are three hyper-parameters. Notice that, for type-dependent sampling and structure-based type-fusion sampling, ξ is set as 0.

Specifically, as an example, to address the problem of purchase prediction in E-commerce, we design a semi-supervised framework to optimize the embedding propagation with sampling and purchase prediction simultaneously:

$$L_{task,k} = -\frac{1}{m} \sum_{\langle v_i, v_j, y\rangle \in \mathcal{D}_k} log(\sigma(y(\boldsymbol{h}_i \boldsymbol{W} \boldsymbol{h}_j + b_{task}))), \tag{19}$$

where $\mathcal{D}_k$ is the set of training triplets $\langle i, j, y\rangle$ in the k-th batch such that y is the ground truth of user-item pair v_i and v_j, and $\boldsymbol{W}$ and b_{task} are the weight and bias parameters. This is essentially a link prediction task. More generally, for node classification, $L_{task,k}$ can also employ an cross-entropy loss function. On the other hand, the reconstruction loss of embedding propagation with sampling can be defined as follows:

$$L_{ep,k} = \frac{1}{N|\boldsymbol{Neg}(v_i)|} \sum_{i \in \boldsymbol{B}_k, v_j \in \boldsymbol{Neg}(v_i)} \left[\gamma + ||\tilde{\boldsymbol{h}}_i - \boldsymbol{h}_i||_2^2 - ||\tilde{\boldsymbol{h}}_i - \boldsymbol{h}_j||_2^2\right]_+, \tag{20}$$

where negative sampling from non-neighborhoods is employed such that $\boldsymbol{Neg}(v_i)$ is the set of negative neighborhoods of v_i, $\tilde{\boldsymbol{h}}_i$ is the reconstructed embedding of v_i, γ is a threshold parameter, and $[\cdot]_+$ is *Relu* activation function. Notice that $\tilde{\boldsymbol{h}}_i$ can be calculated by Equation (5) where $\boldsymbol{g}_{i,r}$ is replaced by our proposed estimators in Equation (16) or (17). For adaptive type-fusion samplers, we reduce the variance during training to optimize the parameters of such samplers, namely, $Var_{q_a}(\hat{\boldsymbol{\mu}}_q)$. By designing such a unsupervised loss function, embedding information of neighborhoods can be propagated to nodes iteration by iteration where each iteration can be considered as a convolutional layer.

The process of training with the type-dependent IS strategy is outlined in Algorithm 1. Likewise, the adaptive type-fusion strategy is summarized in Algorithm 2.

ALGORITHM 1: Type-dependent strategy (one batch)

```
Input: target nodes $\{v_i\}$; neighborhood $\{v_j\}$;
       sampling size of each type $\{n_r | r \in \mathcal{R}\}$; embeddings $\boldsymbol{H}^{(k)}$; parameters $\Theta^{(k)}$.
Output: the optimized embedding $\boldsymbol{H}^{(k+1)}$ and parameters $\Theta^{(k+1)}$.
for each $r \in \mathcal{R}$ do
    compute $p$ and the sampler $q_r$ by Equation (11);
    sample $n_r$ neighborhoods with the sampler $q_r$;
end for
for each $v_i$ do
    for each $r \in \mathcal{R}$ do
        compute the estimator $\hat{\boldsymbol{g}}_{i,r}$ by Equation (16);
    end for
    compute the reconstructed $\tilde{\boldsymbol{h}}_i$ by Equation (5);
    compute $L_{ep,k}(v_i)$ based on $\tilde{\boldsymbol{h}}_i$ and $\boldsymbol{h}_i$;
end for
minimize $L_k(\boldsymbol{H}^{(k)}, \Theta^{(k)})$ and perform gradient updates;
output the optimized $\Theta^{(k+1)}$ and $\boldsymbol{H}^{(k+1)}$.
```

ALGORITHM 2: Adaptive type-fusion strategy (one batch)

```
Input: target nodes $\{v_i\}$; neighborhood $\{v_j\}$;
       the size of samples $n$; the balance parameter $\xi$; embeddings $\boldsymbol{H}^{(k)}$;parameters $\Theta^{(k)}$.
Output: the optimized embedding $\boldsymbol{H}^{(k+1)}$ and parameters $\Theta^{(k+1)}$.
compute $p$ and the sampler $q_a$ by Equation (15);
for each $v_i$ do
    compute the estimator $\hat{\boldsymbol{g}}_i$ by Equation (16);
    compute the reconstructed $\tilde{\boldsymbol{h}}_i$ with $\{\hat{\boldsymbol{g}}_{i,r} | r \in \mathcal{R}\}$ by Equation (5);
end for
minimize $L_k(\boldsymbol{H}^{(k)}, \Theta^{(k)})$ and perform gradient updates;
output the optimized $\Theta^{(k+1)}$ and $\boldsymbol{H}^{(k+1)}$.
```

5 EXPERIMENTS

In this section, we evaluate the empirical performance of our method on five real-world heterogeneous graphs. More specifically, we study the effectiveness and efficiency of our sampling strategies in the context of node classification and link prediction tasks.

5.1 Datasets and Tasks

The statistics of our five datasets, namely, DBLP, Aminer, IMDB, Yelp, and the Alibaba graph, are summarized in Table 2. We describe these four datasets and their associated tasks in the following.

5.1.1 DBLP Bibliographic Network. This is a public bibliographic dataset,[1] which consists of three types of nodes, namely, authors (A), papers (P), and conferences (C), as well as five types of edges, namely, "co-authorship" (A-A), "attend" (A-C), "written by" (P-A), "publish" (P-C), and a composite meta-path (P-A-P). We treat the occurrence frequency of the five relations as edge features, and subsequently construct the bibliographic network as a heterogeneous graph. This is

[1] Available at https://dblp.uni-trier.de/db/.

Table 2. Description of Datasets

Dataset	# Nodes	# Edges	Relations (A-B)	# A-B	Feature	Training	Validation	Test
DBLP	18,405	212,190	Paper-Author	19,645	1	2,027	1,013	1,014
			Paper-Conf.	14,328				
			Paper-Paper	16,2440				
			Author-Author	6,572				
			Author-Conf.	9,205				
Aminer	41,523	199,429	Paper-Author	52,539	1	9,232	4,616	4,616
			Paper-Conf.	18,464				
			Author-Conf.	52,539				
			Author-Author	75,887				
IMDB	11,958	43,937	Moive-Director	4,349	49	1,793	896	896
			Moive-Actor	13,033				
			Actor-Actor	13,522				
			Actor-Director	13,033				
Yelp	971,258	7,159,671	User-review-Business	4,569,305	8	159,033	79,516	79,516
			User-tip-Business	1,619,108				
			User-User	971,258				
Alibaba	4,527,222	49,785,900	User-click-Item	44,664,880	30	1,698,375	646,364	646,364
			User-collect-Item	3,603,744				
			User-cart-Item	1,517,276				

a relatively small graph, enabling us to study various properties of our proposed strategies and comparing with some baselines that cannot scale to larger graphs. On this dataset, the authors are assigned to four research domains, and we are to predict the class of the given authors. The authors are randomly split into subsets of ratio 2:1:1 for training, validation and test.

5.1.2 Aminer Bibliographic Network. This is also a public benchmark dataset[2] [18], which consists of three types of nodes, namely, authors (A), papers (P), and venues (V), as well as four types of edges, namely, "co-authorship" (A-A), "publication" (P-V), "participation" (A-V) and "write" (A-P). We treat the times of collections as edge features and construct the heterogeneous graph similar to that on DBLP dataset. We adopt the research domains as classes, and perform multi-class node classification for papers on this dataset. The ratio of training, validation and test is 2:1:1 as well.

5.1.3 IMDB Movie Network. This is a movie information dataset,[3] which consists of three types of nodes, namely, actors (A), movies (M), and directors (D), as well as four types of edges, namely, "directed" (D-M), "participation" (A-M), "co-operation" (A-A), "participating" (A-M), and "reused" (A-D). On this dataset, the records of movie information are from the 1900s to 2016. We extract the records that appeared before 2014 to construct a heterogeneous movie graph and our goal is to predict whether the given actor will co-operate with the specific director from 2014 to 2016. We randomly generate negative labels five times as much as positive ones, and the ratio of training, validation and test is also 2:1:1. In addition, we adopt the meta-paths (A-D-A & A-A), (M-D-M & M-A-M), and (D-A-D) to respectively generate negative neighbors for actors, movies and directors.

[2] Available at http://resource.aminer.org/lab-datasets/crossdomain/.

[3] Available at http://www.imdb.com.

5.1.4 Yelp Business Graph. This is a public large-scale user review dataset,[4] recording users' reviews and tips as well as friendships. It consists of two types of nodes, namely, users (U) and businesses (B), as well as three types of relations, namely, "reviewed" and "tipped" between users and businesses, and "friendship" between users. Furthermore, all these relations contain several features, like rating scores and fans. We extract records before 24th Oct. 2019 to construct a HIG and our goal is to predict whether the pointed users will review the given items. We set the remainder reviews as positive labels and randomly generate negative labels four times more than positive ones, and the ratio of training, validation and test is also 2:1:1. Moreover, we randomly sample negative neighbors for users and items.

5.1.5 Alibaba E-commerce Graph. This is a public large-scale user activity dataset,[5] capturing 1 week's user actions on the Alibaba platform. It consists of two types of nodes, namely, users (U) and items (I), as well as multiple types of relations, namely, "click," "cart," "favorite," and "buy" between users and items. Between a pair of user and item, we concatenate their relation type encoding and attribute vectors to form the edge features, and subsequently construct a HIG. Notice that the Alibaba graph is a real Alibaba graph roughly 100 times larger than the Aminer graph, enabling us to study the scalability of different methods. On this dataset, we would like to predict users' purchase actions. This task can be formulated as a binary classification problem, where we only use the first 5 days of data for training, and the remaining 2 days of data for testing. In addition, we randomly sample five negative neighbors for each node. Since it could be quite difficult to judge whether two users/items are different, we respectively sample negative items for users and sample negative users for items.

We adopt ***Micro-F1*** and ***Macro-F1*** as the evaluation metrics for both DBLP and Aminer while adopting ***F1*** and ***AUC*** as the evaluation metrics for IMDB, Yelp, and the Alibaba graph. All the above metrics are positively related to the performance of methods.

5.2 Baselines and Experimental Settings

We first compare our various IS strategies with the general HIGE model and HAN [35] without any sampling. We also compare with state-of-the-art sampling algorithms on GCN [5, 19] to showcase that it has limited effectiveness and scalability, whereas the sampling-based models not only achieve superior accuracy, but also scale to large graphs. Finally, to study the utility of our proposed variance reduction, we also substitute our variance reduction sampler with a uniform sampler. These methods are summarized in the following.

- **HIGE:** This is the general heterogeneous embedding model on the whole graph. This can be understood as an extreme case where 100% neighbors are sampled. As all information is preserved, the effectiveness of this method intuitively represents an upper-bound for all sampling-based methods.
- **HIGE-Nil:** At the other extreme of HIGE, we sample 0% or none of the neighbors. This is also equivalent to setting $\alpha = 0$ in Equation (18). Intuitively, we consider the effectiveness of this method as a lower-bound; any method that utilizes neighborhood information should outperform it.
- **HAN [35]:** This is a heterogeneous graph representation model that aggregates information from different-typed neighborhoods by utilizing semantic-level and node-level attention mechanisms. Here we have implemented the batch-wise version which supports the one-hot features of nodes and can be used to deal with link prediction tasks.

[4] Available at https://www.yelp.com/dataset/.
[5] Available at https://tianchi.aliyun.com/dataset/dataDetail?dataId=9716.

- **GraphSAGE [16]:** This is an inductive graph representation model which aggregates information based on node-wise random sampling. In this model, per node on the higher layer aggregates information from its sampled lower-layer neighborhoods.
- **Fast-GCN [5]:** This is an accelerated GCN framework based on layer-wise IS. In this model, the neighborhood is sampled based their structural information. Here we utilize the transductive version which consider embedding vectors as node features and construct the homogeneous graph without types of relations.
- **AS-GCN [19]:** This is an accelerated GCN framework based on layer-wise adaptive sampling. This model optimizes parameters of the designed samplers in each iteration. The fundamental settings including node representation and graph construction are the same as those in Fast-GCN.
- **Unif-TF** and **Unif-TD:** methods prefixed with "Unif" substitute the variance reduction sampler in our proposed strategy with a uniform sampler.
- **VarR-TD**, **VarR-TF** , **VarR-TF-SN**, and **VarR-TF-AS:** Methods prefixed with "VarR" denote our variance reduction sampler under different policies, including type-dependent (TD) or type-fusion (TF), type-fusion with self-normalization (TF-SN), and adaptive type-fusion (TF-AS).

For our methods and all the baselines, we set $\beta = 0.1$, $\gamma = 0.1$, $\psi = 0.1$. We set the value of α of HIGE according to the performance on validations. Here we respectively set α as 0.4, 0.5, 1, 0.4, and 0.4 for the five datasets according to the performance on validations. For GraphSAGE, Fast-GCN, and AS-GCN, we adopt two layers, since the computational cost of AS-GCN increases rapidly when the layers become deeper and deeper. For HAN, we set all the remainder parameters the same in [35]. For DBLP, Aminer, IMDB, and Yelp, the batch size is set as 1,024 and the sizes of samples are 128, 256, 512, and 1,024. For Alibaba dataset, the batch size is set as 4,096 and the sizes of samples are 512, 1,024, 2,048, and 4,096. For DBLP, Aminer, IMDB, Yelp, and Alibaba, the maximum iteration is respectively set as 100, 100, 500, 5, and 5.

All programs are implemented in Python 3.6 using TensorFlow 1.12.0. The experiments are conducted on a Linux server with two Intel(R) Xeon(R) CPU E5-2682 v4 @ 2.50 GHz, two Nvidia Tesla M40 and 128 GB RAM. The codes of GraphSage, Fast-GCN, and AS-GCN are provided by the authors of the original papers.

5.3 Empirical Validation

We perform empirical validation on both the relatively small Aminer, DBLP, and IMDB datasets, as well as the much larger Yelp review graph and Alibaba interaction graph. Note that the non-HIGE-based baselines, GraphSAGE, Fast-GCN, and AS-GCN are to deal with node classification, and these baselines can only work on Aminer and DBLP; they cannot complete on IMDB, Yelp, and Alibaba where the task is link prediction. Furthermore, they are likely to suffer from insufficient memory on the large-scale Yelp and Alibaba graph even if we implement the modified models for the task of link prediction.

5.3.1 Effectiveness. We report the results in Tables 3–7, respectively. We progressively sample more neighbors per batch and the sampling rate ranges from about 1.5% to 24% on smaller DBLP, Aminer, and IMDB, 3% to 25% on the larger Yelp graph, and 1.25% to 10% on the larger Alibaba graph. We make the following observations.

- VarR-TF-AS generally achieves the best performance on the four datasets for node classification and link prediction, whereas VarR-TF-SN comes as a close competitor. In particular, VarR-TF-AS performs as well as or even better than the original HIGE. This phenomenon

Table 3. Micro/Macro-F1 Scores for Node Classification on DBLP

	Micro-F1				Macro-F1			
Sampling size per batch	128 ∼ 3%	256 ∼ 6%	512 ∼ 12%	1024 ∼ 24%	128 ∼ 3%	256 ∼ 6%	512 ∼ 12%	1024 ∼ 24%
HIGE-Nil	0.2411 (intuitive lower bound)				0.2403 (intuitive lower bound)			
HIGE	0.8821 (intuitive upper bound)				0.8766 (intuitive upper bound)			
HAN	0.8908				0.8825			
GraphSAGE	0.2060	0.2067	0.2485	0.2505	0.2011	0.2034	0.2346	0.2399
Fast-GCN	0.2187	0.2453	0.2601	0.2709	0.2097	0.2189	0.2350	0.2363
AS-GCN	0.2739	0.2778	0.2808	0.2813	0.2465	0.2498	0.2564	0.2604
Unif-TD	0.2779	0.2667	0.3170	0.3438	0.2425	0.2220	0.2344	0.3261
Unif-TF	0.3058	0.2857	0.2656	0.3103	0.2334	0.2327	0.2598	0.2634
VarR-TD	0.7645	0.8147	0.8426	0.8393	0.7558	0.8086	0.8316	0.8204
VarR-TF	0.8636	0.8795	0.8991	0.8944	0.8584	0.8730	0.8920	0.8882
VarR-TF-SN	0.8648	0.8694	0.8962	0.8976	0.8609	0.8662	0.8916	0.8906
VarR-TF-AS	**0.8828**	**0.9073**	**0.9017**	**0.9062**	**0.8778**	**0.9020**	**0.8962**	**0.9009**

Excluding HIGE and HAN, the best model is bolded and the second best is underlined.

Table 4. Micro/Macro-F1 Scores for Node Classification on Aminer

	Micro-F1				Macro-F1			
Sampling size per batch	128 ∼ 3%	256 ∼ 6%	512 ∼ 12%	1024 ∼ 24%	128 ∼ 3%	256 ∼ 6%	512 ∼ 12%	1024 ∼ 24%
HIGE-Nil	0.1990 (intuitive lower bound)				0.1961 (intuitive lower bound)			
HIGE	0.9646 (intuitive upper bound)				0.9593 (intuitive upper bound)			
HAN	0.9512				0.9508			
GraphSAGE	0.2060	0.2067	0.2125	0.2119	0.2011	0.2034	0.2036	0.2073
Fast-GCN	0.2117	0.2244	0.2318	0.2361	0.1850	0.1898	0.2116	0.2119
AS-GCN	0.2361	0.2307	0.2390	0.2390	0.2005	0.2028	0.2004	0.2068
Unif-TD	0.3043	0.4123	0.4256	0.4221	0.2638	0.3907	0.3553	0.3962
Unif-TF	0.1780	0.2229	0.3785	0.6296	0.1266	0.1952	0.3081	0.5727
VarR-TD	0.8086	0.7990	0.8971	0.9123	0.7913	0.7894	0.8945	0.9133
VarR-TF	0.9461	0.9671	**0.9712**	0.9675	0.9424	0.9651	**0.9696**	0.9664
VarR-TF-SN	**0.9659**	0.9643	0.9612	0.9684	0.9637	0.9629	0.9631	0.9603
VarR-TF-AS	0.9650	**0.9676**	0.9705	**0.9687**	**0.9649**	**0.9667**	0.9602	**0.9671**

Excluding HIGE and HAN, the best method is bolded and the second best is underlined.

is reasonable. On the one hand, HIGE aggregates information from whole neighbors which may introduce some noisy information while our sampling strategies are to sample valuable neighbors for training. On the other hand, we optimize the sampler by reducing the variance loss which enhances the similarity limitation between the sampled neighbors and its target nodes.

—Compared to the corresponding uniform samplers, the variance reduction samplers guarantee more stable estimators and thus produce better results. Furthermore, in the VarR-* methods, type-fusion strategies outperform type-dependent methods, as the former consider all types jointly rather than independently, and reduce the variance of the whole batch rather than a single type.

Table 5. F1 and AUC Scores for Link Prediction on the IMDB Graph

	F1				ROC-AUC			
Sampling size	128	256	512	1024	128	256	512	1024
per batch	~ 1.5%	~ 3%	~ 6%	~ 12%	~ 1.5%	~ 3%	~ 6%	~ 12%
HIGE-Nil	0.2484 (intuitive lower bound)				0.5225 (intuitive lower bound)			
HIGE	0.3325 (intuitive upper bound)				0.6240 (intuitive upper bound)			
HAN	0.3292				0.6215			
Unif-TD	0.2657	0.2500	0.2671	0.2535	0.5313	0.5556	0.5421	0.5127
Unif-TF	0.2500	0.2639	0.2639	0.2682	0.5323	0.5230	0.5279	0.5227
VarR-TD	0.2603	0.2614	0.2694	0.2637	0.5407	0.5536	0.5604	0.5690
VarR-TF	0.2983	**0.3163**	0.3212	**0.3390**	0.5959	0.6073	**0.6135**	**0.6244**
VarR-TF-SN	0.3022	0.3054	0.3205	0.3288	0.5986	0.5992	0.6097	0.6228
VarR-TF-AS	**0.3116**	0.3122	**0.3233**	0.3300	**0.6048**	**0.6077**	0.6128	0.6232

Excluding HIGE and HAN, the best method is bolded and the second best is underlined.

Table 6. F1 and AUC Scores for Purchase Prediction on the Yelp Graph

	F1				ROC-AUC			
Sampling size	128	256	512	1024	128	256	512	1024
per batch	~ 3%	~ 6%	~ 13%	~ 25%	~ 3%	~ 6%	~ 13%	~ 25%
HIGE-Nil	0.4128 (intuitive lower bound)				0.6329 (intuitive lower bound)			
HIGE	0.5034 (intuitive upper bound)				0.7041 (intuitive upper bound)			
HAN	0.4992				0.7005			
Unif-TD	0.4180	0.4144	0.4148	0.4414	0.5913	0.6487	0.6529	0.6575
Unif-TF	0.4226	0.4177	0.4154	0.4397	0.6466	0.6509	0.6531	0.6545
VarR-TD	0.4829	0.4970	0.5000	0.5067	0.6675	0.6779	0.6819	0.6950
VarR-TF	0.4786	0.5034	0.5124	0.5184	0.6789	0.6849	0.7010	0.7107
VarR-TF-SN	0.4793	0.4950	**0.5190**	**0.5201**	0.6866	0.6826	0.7027	0.7105
VarR-TF-AS	**0.4931**	**0.5036**	0.5128	0.5189	**0.6869**	**0.6914**	**0.7085**	**0.7117**

Excluding HIGE and HAN, the best method is bolded and the second best is underlined.

Table 7. F1 and AUC Scores for Purchase Prediction on the Alibaba Graph

	F1				ROC-AUC			
Sampling size	512	1024	2048	4096	512	1024	2048	4096
per batch	~ 1.25%	~ 2.5%	~ 5%	~ 10%	~ 1.25%	~ 2.5%	~ 5%	~ 10%
HIGE-Nil	0.3994 (intuitive lower bound)				0.5134 (intuitive lower bound)			
HIGE	0.5663 (intuitive upper bound)				0.7715 (intuitive upper bound)			
HAN	0.5618				0.7704			
Unif-TD	0.4017	0.4226	0.4352	0.4371	0.5768	0.5826	0.5908	0.5924
Unif-TF	0.4008	0.4122	0.4125	0.4451	0.5731	0.5790	0.5862	0.5977
VarR-TD	0.4841	0.5003	0.5274	0.5682	0.6207	0.6504	0.6925	0.7475
VarR-TF	0.5769	0.5908	0.5709	0.5833	0.7648	**0.7671**	0.7653	0.7796
VarR-TF-SN	**0.5780**	0.5883	0.5802	0.5798	0.7669	0.7625	0.7660	0.7742
VarR-TF-AS	0.5729	**0.5913**	**0.5806**	**0.5844**	**0.7674**	0.7641	**0.7676**	**0.7799**

Excluding HIGE and HAN, the best method is bolded and the second best is underlined.

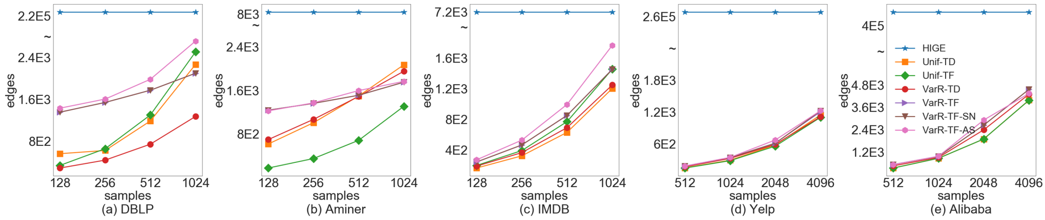


Fig. 3. Average sampled edges per batch of training.

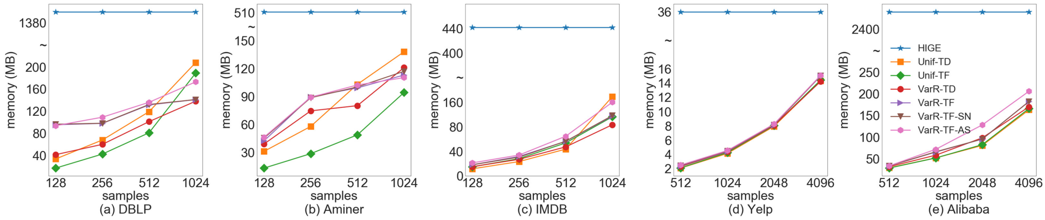


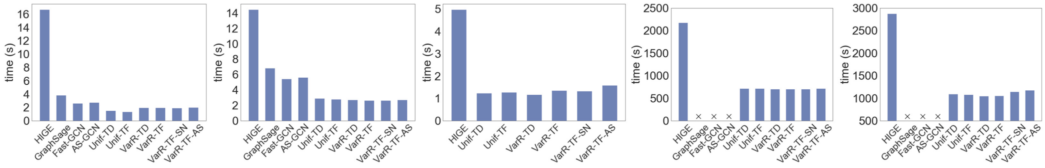
Fig. 4. Average memory cost per batch of training.

- As the sampling size increases, it is not surprising that almost all sampling strategies tend to perform better. In particular, when we sample smaller neighborhoods (like 128 on Aminer and 1,024 on Alibaba), VarR-TF-AS almost reaches better performance than the full HIGE model (in terms of micro-F1 and macro-F1 on Aminer and AUC on the Alibaba graph). Note that the full HIGE and HAN model does not resort to any sampling approximation, and thus can be deemed an intuitive upper bound.
- Our proposed general HIGE can perform competitively with or even better than HAN on the five datasets. However, HAN has higher time complexity. For example, for the large-scale Alibaba dataset, the time cost of HAN (about 1 day) is quite larger than HIGE (about 5 hours).
- All sampling strategies for HIGE, including those with uniform samplers, perform significantly better than GCN-based models (Fast-GCN and AS-GCN) and GraphSage. On the one hand, sampling for HIGE takes the graph heterogeneity into consideration, whereas the two GCN-based models do not make use of such information. On the other hand, the sampling size or number of layers of Fast-GCN and AS-GCN may be not enough. However, even under current settings, Fast-GCN and AS-GCN are already several times slower than our method, as we shall see in the efficiency study.

5.3.2 Efficiency. We first investigate the efficiency of our sampling strategies in the context of the *sampled edges*. As shown in Figure 3, the sampled edges increase when more neighbors are sampled. Nevertheless, even with a large sample size of 4,096 per batch, only very few edges are sampled. In particular, our sampling strategies VarR-TF-AS and VarR-TF-SN require 93.36% and 92.55% fewer edges than the full HIGE model on the large-scale Alibaba graph, respectively, and yet they achieve better performance.

Second, similar observations can be made on the memory cost shown in Figure 4, where VarR-TF-AS/VarR-TF-SN incur 93.22%/92.48% less memory cost on the Alibaba, respectively. Further note that the differences in both the number of edges and memory cost are more prominent on the larger Alibaba graph, indicating excellent scalability of our sampling strategies.

Third, in terms of the running time (as shown in Figure 5), VarR-TF-AS and VarR-TF-SN require less time than HIGE to attain close performance on the five datasets. In particular, on the



(a) DBLP (b) Aminer (c) IMDB (d) Yelp (e) Alibaba

Fig. 5. Average running time per iteration of training.

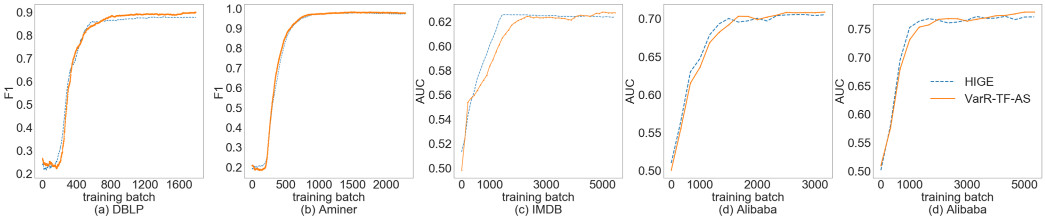


Fig. 6. The convergence of training.

smaller DBLP, Aminer, and IMDB datasets, VarR-TF-AS incur by up to 93.36% less memory cost, compared to HIGE, and almost half the time cost compared to GraphSage. On IMDB, Yelp, and Alibaba datasets, the original GraphSAGE, Fast-GCN, and AS-GCN fail to work since the three methods deal with node classification rather than link prediction. Furthermore, these sampling models which require all edges to be preloaded would suffer from the extremely large memory cost and time cost when dealing with large-scale graphs. In summary, compared to HIGE, our models incur by up to 86.25%, 84.39%, 67.31%, 65.24%, and 58.32% less time cost on DBLP, Aminer, IMDB, and the larger Yelp and Alibaba graph, respectively. Notice that since HAN is an supervised model where the time cost is related to labels, it could be unfair to compare the time cost of HAN and HIGE. HAN is faster than HIGE when dealing with smaller DBLP, Aminer and IMDB, while HIGE is quite faster than HAN on the large Alibaba dataset. For instance, when dealing with large-scale Alibaba dataset, the time cost is quite larger (about 1 day) than HIGE (about 5 hours).

Fourth, by analyzing the convergence of VarR-TF-AS and the original HIGE shown in Figure 6, it is obvious that our VarR-TF-AS has quick convergence, especially on the denser Aminer and DBLP datasets because of quite lower time cost when achieving similar performance.

6 CONCLUSIONS

In this article, we focus on the problem of accelerating large-scale heterogeneous graph embedding with IS. There are two main challenges, one of which is how to design an effective estimator that work with heterogeneous neighborhood, and the other is how to balance the structural information of each target node in its own neighborhood, with the common neighborhood for all target nodes in a batch. To address the challenges, in this article, we design various IS strategies, namely, type-dependent and type-fusion samplers, with self-normalized and adaptive estimator. Furthermore, we conduct extensive experiments on four public real-world dataset, including the Aminer dataset and the large-scale e-commerce interaction graph. We analyze the experimental results on various aspects including effectiveness, memory cost and time cost. The experimental results have shown the advantages of our strategies in both effectiveness and efficiency.

Apart from the static edge information of HIGs, the dynamics of interactions also indicate the importance where recent interactions play more important roles. We leave dynamic neighborhood sampling of HIGs as our future work.

ACKNOWLEDGMENTS

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agencies. This work was done when Yugang Ji was visiting Alibaba as a research intern.

REFERENCES

[1] Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. 2015. Variance reduction in SGD by distributed importance sampling. *arXiv preprint arXiv:1511.06481* (2015).

[2] Sezin Kircali Ata, Yuan Fang, Min Wu, Xiao-Li Li, and Xiaokui Xiao. 2017. Disease gene classification with metagraph representations. *Methods* 131 (2017), 83–92.

[3] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1616–1637. DOI: https://doi.org/10.1109/TKDE.2018.2807452

[4] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation learning for attributed multiplex heterogeneous network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'19), Anchorage, AK, USA, August 4-8, 2019*. ACM, 1358–1368. DOI: https://doi.org/10.1145/3292500.3330964

[5] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *Proceedings of the 6th International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=rytstxWAW.

[6] Long Chen, Fajie Yuan, Joemon M. Jose, and Weinan Zhang. 2018. Improving negative sampling for word representation using self-embedded features. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 99–107. DOI: https://doi.org/10.1145/3159652.3159695

[7] Dominik Csiba and Peter Richtárik. 2018. Importance sampling for minibatches. *Journal of Machine Learning Research* 19 (2018), 27:1–27:21. Retrieved from http://jmlr.org/papers/v19/16-241.html.

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the Advances in Neural Information Processing Systems*. 3837–3845. Retrieved from http://papers.nips.cc/paper/6081-convolutional-neural-networks-on-graphs-with-fast-localized-spectral-filtering.

[9] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 135–144. DOI: https://doi.org/10.1145/3097983.3098036

[10] Víctor Elvira, Luca Martino, David Luengo, and Mónica F. Bugallo. 2015. Efficient multiple importance sampling estimators. *IEEE Signal Processing Letters* 22, 10 (2015), 1757–1761.

[11] Yuan Fang, Wenqing Lin, Vincent W. Zheng, Min Wu, Kevin Chen-Chuan Chang, and Xiao-Li Li. 2016. Semantic proximity search on graphs with metagraph-based learning. In *Proceedings of the 2016 IEEE 32nd International Conference on Data Engineering (ICDE'16)*. IEEE, 277–288.

[12] Tao-Yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. HIN2Vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1797–1806. DOI: https://doi.org/10.1145/3132847.3132953

[13] Alberto García-Durán and Mathias Niepert. 2017. Learning graph representations with embedding propagation. In *Proceedings of the Advances in Neural Information Processing Systems*. 5125–5136. Retrieved from http://papers.nips.cc/paper/7097-learning-graph-representations-with-embedding-propagation.

[14] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks*. Vol. 2. IEEE, 729–734.

[15] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 855–864. DOI: https://doi.org/10.1145/2939672.2939754

[16] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the Advances in Neural Information Processing Systems*. 1025–1035. Retrieved from http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.

[17] Carsten Hartmann, Christof Schütte, Marcus Weber, and Wei Zhang. 2018. Importance sampling in path space for diffusion processes with slow-fast variables. *Probability Theory and Related Fields* 170, 1–2 (2018), 177–228.

[18] Dimitar Hristovski, Andrej Kastrin, and Thomas C. Rindflesch. 2016. Implementing semantics-based cross-domain collaboration recommendation in biomedicine with a graph database. In *Proceedings of the American Medical Informatics Association Annual Symposium.* Retrieved from http://knowledge.amia.org/amia-63300-1.3360278/t005-1.3362920/f005-1.3362921/2499243-1.3363975/2499720-1.3363972.

[19] Wen-bing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. 2018. Adaptive sampling towards fast graph representation learning. In *Proceedings of the Advances in Neural Information Processing Systems.* 4563–4572. Retrieved from http://papers.nips.cc/paper/7707-adaptive-sampling-towards-fast-graph-representation-learning.

[20] Xiao Huang, Jundong Li, Na Zou, and Xia Hu. 2018. A general embedding framework for heterogeneous information learning in large-scale networks. *ACM Transactions on Knowledge Discovery from Data* 12, 6 (2018), 70:1–70:24. DOI: https://doi.org/10.1145/3241063

[21] Zan Huang, Wingyan Chung, and Hsinchun Chen. 2004. A graph model for E-commerce recommender systems. *Journal of the Association for Information Science and Technology* 55, 3 (2004), 259–274. DOI: https://doi.org/10.1002/asi.10372

[22] Zhipeng Huang, Yudian Zheng, Reynold Cheng, Yizhou Sun, Nikos Mamoulis, and Xiang Li. 2016. Meta structure: Computing relevance in large heterogeneous information networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 1595–1604.

[23] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations.* Retrieved from https://openreview.net/forum?id=SJU4ayYgl.

[24] John Boaz Lee, Ryan A. Rossi, Sungchul Kim, Nesreen K. Ahmed, and Eunyee Koh. 2019. Attention models in graphs: A survey. *ACM Transactions on Knowledge Discovery from Data* 13, 6 (2019), 62.

[25] Luca Martino, Víctor Elvira, and Francisco Louzada. 2017. Effective sample size for importance sampling based on discrepancy measures. *Signal Processing* 131 (2017), 386–401.

[26] Art B. Owen. 2013. *Monte Carlo Theory, Methods and Examples.*

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14).* 701–710. DOI: https://doi.org/10.1145/2623330.2623732

[28] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2008), 61–80.

[29] Jingbo Shang, Meng Qu, Jialu Liu, Lance M. Kaplan, Jiawei Han, and Jian Peng. 2016. Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. *CoRR* abs/1610.09769 (2016). arxiv:1610.09769 http://arxiv.org/abs/1610.09769

[30] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and Philip S. Yu. 2019. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 31, 2 (2019), 357–370. DOI: https://doi.org/10.1109/TKDE.2018.2833443

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

[32] 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. *Proceedings of the VLDB Endowment* 4, 11 (2011), 992–1003.

[33] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web.* 1067–1077. DOI: https://doi.org/10.1145/2736277.2741093

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks. *CoRR* abs/1710.10903 (2017). arxiv:1710.10903 http://arxiv.org/abs/1710.10903

[35] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous graph attention network. In *Proceedings of the 28th International Conference on World Wide Web (WWW'19), San Francisco, CA, USA, May 13-17, 2019.* ACM, 2022–2032. DOI: https://doi.org/10.1145/3308558.3313562

[36] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596* (2019).

[37] Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th International Conference on World Wide Web.* 1271–1279. DOI: https://doi.org/10.1145/3038912.3052558

[38] Carl Yang, Jieyu Zhang, and Jiawei Han. 2019. Neural embedding propagation on heterogeneous networks. In *Proceedings of the 2019 IEEE International Conference on Data Mining (ICDM'19), Beijing, China, November 8-11, 2019.* IEEE, 698–707. DOI: https://doi.org/10.1109/ICDM.2019.00080

[39] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983. DOI: https://doi.org/10.1145/3219819.3219890

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

[41] Zi-Ke Zhang, Tao Zhou, and Yi-Cheng Zhang. 2010. Personalized recommendation via integrated diffusion on user–item–tag tripartite graphs. *Physica A: Statistical Mechanics and its Applications* 389, 1 (2010), 179–186.

[42] Vincent W. Zheng, Mo Sha, Yuchen Li, Hongxia Yang, Yuan Fang, Zhenjie Zhang, Kian-Lee Tan, and Kevin Chen-Chuan Chang. 2018. Heterogeneous embedding propagation for large-scale e-commerce user alignment. In *Proceedings of the IEEE International Conference on Data Mining*. 1434–1439. DOI: https://doi.org/10.1109/ICDM.2018.00198

[43] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. 2019. Layer-dependent importance sampling for training deep and large graph convolutional networks. In *Proceedings of the Advances in Neural Information Processing Systems*. 11247–11256.

Received December 2019; revised May 2020; accepted August 2020