HuGE: An Entropy-driven Approach to Efficient and Scalable Graph Embeddings

Peng Fang, Fang Wang†, Zhan Shi, Hong Jiang*, Dan Feng, and Lei Yang
Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System,
Engineering Research Center of data storage systems and Technology, Ministry of Education of China,
School of Computer Science and Technology, Huazhong University of Science and Technology, China
* *Department of Computer Science and Engineering, University of Texas at Arlington, USA*
{fangpeng, wangfang, zshi, dfeng, leiyang}@hust.edu.cn; *hong.jiang@uta.edu; †Corresponding author

***Abstract*—Graph embedding is becoming widely adopted as an efficient way to learn graph representations required to solve graph analytics problems. However, most existing graph embedding methods, owing to computation-efficiency challenges for large-scale graphs, generally employ a one-size-fits-all strategy to extract information, resulting in a large amount of redundant or inaccurate representations. In this work, we propose HuGE, an efficient and scalable graph embedding method enabled by an entropy-driven mechanism. Specifically, HuGE leverages hybrid-property heuristic random walk to capture node features, which considers both node degree and the number of common neighbors in each walking step. More importantly, to guarantee information effectiveness of sampling, HuGE adopts two heuristic methods to decide the random walk length and the number of walks per node, respectively. Extensive experiments on real-world graphs demonstrate that HuGE achieves both efficiency and performance advantages over recent popular graph embedding approaches. For link prediction and multi-label classification, our approach not only offers >10% average gains, but also exhibits 22×–126× speedup compared with existing sampling-based methods.**

***Index Terms*—Graph embedding, Efficiency, Scalability, Entropy-driven**

I. INTRODUCTION

Graph is an important data structure for representing connected entities and relationships existing in a wide variety of real-world scenarios, such as social networks, biological graphs, traffic networks, semantic graphs, etc. Over the last few years, graph analytics has received significant attention for its ability to extract meaningful insights from large-scale graphs representing the above real-world scenarios. Graph embedding (a.k.a. network embedding), as an effective technique of graph analytics, has become a key method to learn node representations from a graph [1]. Specifically, it aims to embed the nodes of a graph into a low-dimensional vector space, while preserving the inherent structural properties of the graph, and the representation vectors can serve a wide range of downstream graph tasks through machine learning methods.

In the current literature, existing approaches in graph embedding roughly fall into three categories: *sampling-based* techniques such as Deepwalk [2], Node2vec [3], LINE [4], Struc2vec [5], VERSE [6], and DiaRW [7]; *matrix factorization-based (MF-based)* techniques such as HOPE [8], NetMF [9], STRAP [10], ProNE [11], and NRP [12]; and *graph neural networks-based (GNN-based)* techniques such as GraphSage [13], Graph Attention [14], and GraphGAN [15]. However, most of the existing methods face efficiency and scalability challenges in the embedding process, especially for large-scale graphs that are increasingly common nowadays. For example, sampling-based techniques, such as node2vec, need to sample a large number of node pairs to ensure the quality of embedding, and thus require substantial computational resources, taking months to learn embeddings for a graph with 100 million nodes and 500 million edges by 20 threads on a modern server [11]. For MF-based techniques, the performance of embedding depends heavily on the DRAM size due to expensive matrix factorization operations, some recent works [10]–[12] attempt to address this challenge, especially NRP [12] efficiently handle a billion-edge Twitter graph on 100GB level RAM. NRP also reveals that the random walk based methods are restricted by the sheering number of possible walks while increasing the walk length, so is it possible to efficiently reduce the redundant walk by a heuristic random walk? For GNN-based techniques, training a neural network also incurs very high computational overhead and inevitably limits their scalability [1], [12]. Therefore, it is critically important to generate high-efficiency graph embeddings on large-scale graphs.

Furthermore, current graph embedding techniques adopt a one-size-fits-all strategy for all nodes and fail to meet the unique requirements of different individual real-world graphs [6]. For the sampling-based approach, which generally samples node pairs from an input graph, defining the proximity of pairwise nodes is the key to achieving effectiveness for various downstream tasks in diverse graphs. Deepwalk is a pioneer work extending word2vec [21] to graph feature learning, it generates sequences of nodes from a graph by uniform random walk but limited in preserving properties for complex graph structures. Considering the diversity of connectivity patterns, Node2vec attempts to ameliorate the flexibility in bias walk via two hyperparameters (p and q), but at the cost of increased space complexity due to repetitive search for the optimal configuration on a large-scale graph. DiaRW is recently proposed as a scalable graph embedding method based on a degree-biased random walk with variable length, yet the random walk length is only determined by

the degree of a source node, ignoring the rich information of proximity nodes. Consequently, the flexibility and effectiveness of strategy should be fully considered in the generated graph embeddings.

To address the above challenges, we propose HuGE, an entropy-driven approach to efficient and scalable graph embeddings. Specifically, HuGE leverages hybrid-property heuristic random walk, which takes into consideration both node degree and the number of common neighbors to exploit the proximity of pairwise nodes. Moreover, from the perspective of information effectiveness, we present heuristic methods based on information theory, primarily referring to information entropy and relative entropy in this case, for the random walk length and number of walks per node, improving the computing and memory efficiency without sacrificing precision.

II. Motivation

Sampling-based techniques inspired by the well-known natural language processing model word2vec [21], transform a graph into a set of random walk paths through sampling and then adopt word2vec (Skip-Gram) to generate graph embeddings from the sampling paths. However, random walk length (L) and the number of walks per node (r) in existing sampling-based techniques heavily rely upon an empirical value set (usually, $L = 80$ and $r = 10$). These static configurations for the sampling procedure may introduce an excessive amount of low-quality information, limiting the efficiency and scalability on large-scale graphs. To better understand the impact of random walk length and the number of walks per node on learning node representations, we quantitatively examine the relationship between information entropy and the walk length, in Fig. 1, and between relative entropy and number of walks per node, in Fig. 2.

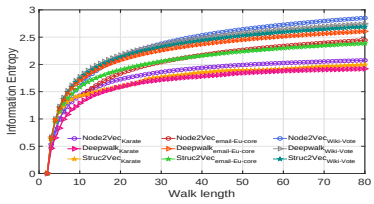

Fig. 1. Information entropy as a function of walk length L for short random walks on three real graphs, Karate, email-Eu-core and Wiki-Vote, with sampling procedures of Node2vec, Deepwalk and Struc2vec, respectively. To ensure comprehensiveness, the mean of information entropy for each short random walk is visualized.

As mentioned in previous approaches, information entropy can be used to measure global or local information in random walk process on graphs [16] and quantify effectiveness of text data [17]. Consider a path W_u^L generated by the walker with length L from source node u, suppose that the path W_u^L passes through the nodes $V_u^1, V_u^2, V_u^3, ..., V_u^i$, then the probability of V_u^i occurring in path W_u^L is $p_{(V_i)} = \frac{n(V_i)}{\sum_{V \in W_u^L} n(V)}$, the information entropy of all paths is $H_{W_u^L} = -\sum p_{(V_i)} log p_{(V_i)}$, indicating that the more evenly distributed the nodes in a path, the larger the H value will be. Fig. 1 shows how information entropy changes with an increasing L on three real graphs. For clarity of display, each curve plots the average H of paths starting from all nodes in the corresponding graph and sampling procedure. Since each path of short random walks corresponds to a sentence from corpus [2], we can find that the commonly used L value of 80 in existing models cannot support a concise and comprehensive representation due to the fact that H is already in a stable state, and will inevitably introduce a great amount of redundant information.

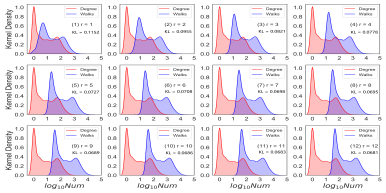

Fig. 2. For the Wiki-Vote graph, Kernel density distributions of node degree (red shaded area) and node occurrence in random walks based on Node2vec (blue shaded area), respectively. Kullback-Leibler divergence (KL) for the two distributions is labeled into subplots. The x axis represents the degree of nodes in graph and the counts of node occurrence in walks.

From a macro perspective, the corpus is generated by multiple rounds (r) of walk paths for each node, and the study in [2] reveals that the distribution of frequency with which nodes appear in corpus ($q(V_i)$) is similar to the node degree distribution of a graph ($p(V_i)$) because they both follow a power-law distribution. Inspired by this observation, we first leverage kernel density distribution [18] to estimate the two distributions and then use Kullback-Leibler divergence (a.k.a. relative entropy) [19] to quantify the discrepancy between $q(V_i)$ and $p(V_i)$, denoted as $KL = \sum p(V_i) log \frac{p(V_i)}{q(V_i)}$. Visually indicated in Fig. 2, as r increases, the first peak of the blue shade (i.e., the tallest on the left in $q(V_i)$) becomes more leptokurtic, and its density distribution more similar to that of the red shade ($p(V_i)$). It also can be found that the KL value labeled in Fig. 2 has a slight variation after $r = 7$, which is less than the empirical value for r = 10.

Note that the low-quality corpus will adversely affect the effectiveness of embedding training and ultimately interfere with the accuracy of downstream tasks [20]. Motivated by these insights, we propose in this paper an entropy-driven graph embedding approach, called HuGE that sheds new light on improving performance via efficient and scalable embedding.

III. Design and Analysis of HuGE

A. Sampling strategy

(1) Hybrid-Property Heuristic Random Walks

Sampling-based graph embedding techniques extract node features through random walks, which have capability of automatically capturing useful representations from complex graph structures [1]. As popularly used in graph statistics, common neighbors can identify the similarity between nodes in graph, besides, most real-world graphs are reported to be scale-free, so as to high-degree nodes in random walks tend to be revisited more and walks starting from them are likely to obtain richer information by traveling around the local

neighbors [7], [23]. Based on these considerations, we propose a hybrid-property heuristic random walk (HRW) for next-hop node selection. HRW differentiates the candidates by taking into account both the number of common neighbors and the degree of candidate nodes. Specifically, if a candidate node v ($v \in N(v_0)$) has more common neighbors with v_0 and a higher degree, then the walker moves to v with a higher probability. As shown in Fig. 3, the number of common neighbors between v_0 and $\{v_1, v_2, v_3, v_4\}$ is {1, 2, 1, 0}, and the degrees of $\{v_1, v_2, v_3, v_4\}$ are respectively {7, 4, 4, 4}. Intuitively, since v_2 has the most common neighbors with v_0, it has the highest similarity to v_0 among nodes in the candidate set. Nevertheless, the walker may have a high chance of walking back to previously visited nodes through common neighbors due to v_2's lower degree than v_1's. On the other hand, since v_1 and v_3 have the same number of common neighbors with v_0, the walker will more likely choose v_1, leading to a higher probability of walking to unvisited nodes. Since the transition probability from v_0 to v_4 will be the lowest due to the minimal similarity with the current node v_0, the proximity of v_0 and v_4 in the graph will likely be minimal. Therefore, by jointly considering common neighbors and degree skewness, HRW is arguably capable of both identifying similar properties and ensuring the exploration of richer information in the random walking.

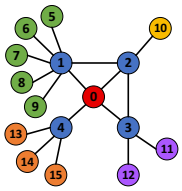

Fig. 3. An example for random walk.

To implement HRW, the key issue is to formulate the selection of the next-hop node with a mathematical model. Let $G = (V, E)$, $u, v \in V$, we first define the node similarity between u and v by the number of distinctive neighbors as:

$$Sim(u, v) = \frac{1}{deg(u) - Cm(u, v)}, if(u, v) \in E \tag{1}$$

where $deg(u)$ is the degree of u. From the above equation, it is obvious that $Sim(u, v)$ grows with the number of common neighbors $Cm(u, v)$ since $deg(u)$ is fixed. Instead of the single property biased random walk, HRW also considers weights in node transition probabilities based on degrees of nodes as follows. Suppose the source node is u, the weight of transition probability from u to v is:

$$\alpha_{(u,v)} = \frac{1}{deg(u) - Cm(u, v)} \times max\left\{\frac{deg(u)}{deg(v)}, \frac{deg(v)}{deg(u)}\right\} \tag{2}$$

the function max is able to exert influence of high degree nodes to obtain richer information by traveling around the local neighbors, reducing the redundancy in the future walking. Take Fig. 3 as an example to illustrate the walking process in one step. Suppose that the walker is current at v_0, $N(v_0)$=$\{v_1, v_2, v_3, v_4\}$, the weights of transition probabilities are $\alpha_{(0,1)} = 7/12$, $\alpha_{(0,2)} = 1/2$, $\alpha_{(0,3)} = 1/3$, $\alpha_{(0,4)} = 1/4$, suggesting that v_1 has a higher probability than other candidate nodes as the next-hop node, which is consistent with the intuition.

To limit the computation overhead, HRW leverages a commonly used strategy, walking-backtracking [7], [23], to determine the next-hop in each walking step. Specifically, at a source node u in each step, HRW randomly chooses v from $N(u)$ as a candidate node, the acceptance probability for v as the next-hop node is $P_{(u,v)}$, and if v is rejected, which will happen with probability $1 - P_{(u,v)}$, the walker will backtrack to u and repeat a random selection again from $N(u)$. Formally, the transition probability from u to v can be written as:

$$P_{(u,v)} = \begin{cases} Z\left(\alpha_{(u,v)}\right), & if(u, v) \in E \\ 0, & other \end{cases} \tag{3}$$

here we do a normalization on $\alpha_{(u,v)}$ by $Z = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, which is widely applied in machine learning and satisfies the walking-backtracking strategy.

(2) Heuristic walk length

In the current literature, most sampling-based graph embedding techniques adopt a one-size-fits-all strategy for all nodes, setting as a fixed walk length that relies on an empirical value set (usually, $L = 80$). As mentioned above, each path of short random walks corresponds to a sentence from the corpus in the Skip-Gram model. The representation of sentences needs to be concise and have good coverage. If L is too large, it will introduce significant redundancy; otherwise, too small an L will make it difficult to guarantee the quality of information. Moreover, an over-sized L would directly increase the sampling overhead and the storage and computation overheads in the learning phase, thereby limiting the scalability to large-scale graphs. Fortunately, information entropy can be used as a measure of how much information is contained in a given source, and thus we propose a heuristic walk length strategy (HWL) to measure the effectiveness of information during walking based on information entropy (H).

The main idea of HWL is to observe the variation of H for short random walks as a node is selected to the path, and if it becomes stable, the random walk is simply stopped. In other words, as shown in Fig. 1, a stable H means that the newly added node has little contribution to representation. HWL characterizes the correlation between the variations of H and L by linear regression and calculates the coefficient of determination (R^2) to determine the termination of a random walk process in the heuristic method. Mathematically, for any $u, v \in G$, the random walk process of the source node u is defined as:

$$W_u^L = \{v_u^1, v_u^2, v_u^3, ..., v_u^k\}, L = 1, 2, 3, \cdots, k \tag{4}$$

where v_u^k denotes k-th node of the path, L is the walk length. With the L variation, the probability of v_i occurrence in the path is $p_{v_i} = \frac{n(v_i)}{\sum_{v \in W_u^L} n(v)}$. Accordingly, information entropy for the random walk is given as follows:

$$H_{W_u^L} = -\sum_i^n \frac{n(v_i)}{\sum\limits_{v \in W_u^L} n(v)} log \frac{n(v_i)}{\sum\limits_{v \in W_u^L} n(v)} \tag{5}$$

Based on the above equation, the Pearson correlation coefficient for $H_{W_u^L}$ and L is computed by

$$r_{(H,L)} = \frac{\sum_{i=1}^{n}(H^i_{W_u^L} - \overline{H_{W_u^L}})(L^i - \overline{L})}{\sqrt{\sum_{i=1}^{n}(H^i_{W_u^L} - \overline{H_{W_u^L}})^2}\sqrt{\sum_{i=1}^{n}(L^i - \overline{L})^2}} \quad (6)$$

where $\overline{H_{W_u^L}}$ and $\overline{H_{W_u^L}}$ are the mean of the series $H^i_{W_u^L}$ and L^i, respectively. In the case of a linear model for H and L, $R^2_{(H,L)}$ is simply the square of $r_{(H,L)}$. The closer $R^2_{(H,L)}$ is to 1, the better linear relation between H and L. Here we set $R^2_{(H,L)} \geq \mu$ as the judgment condition to determine whether the walk stops. To guarantee a linear relationship for two datasets in a linear model, usually, we let $\mu \geq 0.99$.

(3) Heuristic number of walks per node

Sampling-based techniques usually perform multiple walks for each node to guarantee the quality of training. The number of walks per node determines the size of corpus formed by walk paths. As with the fixed walk length that depends on an empirical value set, the number of walks per node (usually set $r = 10$) is also an urgent problem that needs to be solved. The conciseness of corpus can not only ensure the effectiveness of information but also reduce the overhead of the sampling phase and improve efficiency in the learning phase. Based on the observation in Fig. 2, we try to quantify the quality of corpus through the difference between the node occurrence distribution in corpus and the degree distribution in graph. One of the most popular means of measuring the discrepancy between two distributions is to use relative entropy [22]. Accordingly, we present the heuristic number of walks per node (HWN) strategy, focusing on heuristically exploring the number of walks based on relative entropy ($D(p\|q)$). For a given graph G, where any $v_i \in G$, the degree of a node is denoted as $deg(v_i), i = 1, 2, 3, \cdots, n$, and thus the degree distribution is given by:

$$p(v_i) = \frac{deg(v_i)}{\sum_{i=1}^{n} deg(v_i)} \quad (7)$$

The occurrence counts of v_i in the generated corpus is $ocn(v_i), i = 1, 2, 3, \cdots, n$, and the probability for $ocn(v_i)$ is defined as:

$$q(v_i) = \frac{ocn(v_i)}{\sum_{i=1}^{n} ocn(v_i)} \quad (8)$$

Then the relative entropy for $p(v_i)$ and $q(v_i)$ is denoted as:

$$D(q(v_i)\|p(v_i)) = \sum_{i=1}^{n} \frac{deg(v_i)}{\sum deg(v_i)} log \frac{deg(v_i)\sum ocn(v_i)}{ocn(v_i)\sum deg(v_i)} \quad (9)$$

With r increasing, we can calculate the difference of $D(q\|p)$ as:

$$\Delta D_r(q\|p) = |D_r(q(v_i)\|p(v_i)) - D_{r-1}(q(v_i)\|p(v_i))|, r > 1 \quad (10)$$

HWN leverages $\Delta D_r(q\|p) \leq \delta$ as the termination condition for the sampling phase, utilizing a heuristic method to explore the number of walks per node. Taking as an example for the link prediction on Wiki-Vote graph, where $\delta = 0.001$, while the sampling phase needs 7 iterations per node, less than the traditional scheme $r = 10$, it does not reduce the accuracy of tasks. In other cases, HWN will introduce r that is greater than 10 for some graphs, such as CA-AstroPh.

B. Feature learning strategy

Feature learning in graphs can be treated as a maximum likelihood optimization problem [2], [3]. In general, feature learning can be modeled as a mapping function $\varphi : V \rightarrow R^d$ from nodes to feature representations, where d is the number of dimensions for feature representations and $\varphi(u)$ is the embedding vector of node u. For each source node $u \in V$, we define $N_S(u)$ as the neighbors of node u generated by a sampling strategy S. Since our scheme captures node representations based on the Skip-Gram model [21], the optimization for the objective function is given as:

$$\max_{\varphi} \sum_{u \in V} logPr(N_S(u) \mid \varphi(u)) \quad (11)$$

Assuming that the predicting nodes in a context are independent of one another, the conditional probability in Eq.(11) can be approximated by:

$$Pr(N_S(u) \mid \varphi(u)) = \prod_{n_i \in N_{S(u)}} Pr(n_i \mid \varphi(u)) \quad (12)$$

Since neighboring nodes are symmetrical to each other in the feature space, we use the softmax unit to model the conditional likelihood for each source-neighbor node pair as:

$$Pr(n_i \mid \varphi(u)) = \frac{\exp(\varphi(n_i) \cdot \varphi(u))}{\sum_{v \in V} \exp(\varphi(v) \cdot \varphi(u)} \quad (13)$$

The function $\sum_{v \in V} \exp(\varphi(v) \cdot \varphi(u)$ is expensive to compute for large graphs. To reduce this overhead, we speed up the training by approximating it via negative sampling [24].

IV. EVALUATION

A. Experimental Setup

Experiment Environment. We conduct evaluations on a 2.10 GHz Intel Xeon E7-4830 server equipped with 1T RAM and 4.5 T disk, running Ubuntu 18.04 and Linux 3.13.0-160-generic kernel. HuGE is implemented by Python 2.7.6.

Datasets. Eight widely-used real-world graph datasets are employed in our experiments. Table I lists the key properties of these datasets. According to the requirements of evaluation tasks, four graphs are selected for the multi-label classification task: PPI [1], Wiki [2], Flickr [3] and Youtube [3], and five graphs are chosen for the link prediction task: Wiki-Vote [4], CA-AstroPh [4], Youtube [3], LiveJournal [3], and Twitter [3]. Further more, we also generate a set of synthetic graphs [25] for evaluating the scalability of HuGE.

Baselines. We compare HuGE against eight state-of-the-art graph embedding methods, including five sampling-based techniques (Deepwalk [2], Node2vec [3], LINE [4], VERSE [6] and DiaRW [7]), two matrix factorization-based techniques

[1] http://www.thebiogrid.org
[2] https://cs.fit.edu/ mmahoney/compression/text.html
[3] http://socialcomputing.asu.edu
[4] http://snap.stanford.edu/

TABLE I

THE REAL-WORLD GRAPH DATASETS USED IN EXPERIMENTS ($K = 10^3, M = 10^6, B = 10^9$)

Graph	$\|V\|$	$\|E\|$	$Type$	$\|Label\|$	$Density$
PPI	3.89K	76.6K	undirected	50	5.0×10^{-4}
Wiki	4.78K	185K	directed	40	1.6×10^{-2}
Flickr	80.5K	5.90M	undirected	195	1.8×10^{-3}
Youtube	1.14M	2.99M	undirected	47	4.6×10^{-6}
Wiki-Vote	7.12K	104K	directed	–	2.9×10^{-3}
CA-AstroPh	18.72K	198K	undirected	–	1.2×10^{-3}
LiveJournal	2.24M	14.6M	directed	–	5.1×10^{-6}
Twitter	11.3M	85.3M	directed	–	9.9×10^{-7}

TABLE II

AUC OF HUGE AND BASELINES FOR LINK PREDICTION ON FIVE GRAPHS, WHERE A "–" SIGNIFIES THE FAILURE OF THE CORRESPONDING METHOD DUE TO COMPUTE-RESOURCE OR RUNNING-TIME (>5 DAYS) CONSTRAINS.

Method	Wiki-Vote	CA-AstroPh	Youtube	Live Journal	Twitter
Deepwalk	0.801	0.887	0.707	0.867	–
Node2vec	0.794	0.876	0.728	0.870	–
LINE	0.788	0.872	0.805	0.861	–
HOPE	0.854	0.924	–	–	–
VERSE	0.867	0.937	**0.827**	0.859	–
ProNE	0.853	0.949	0.766	0.881	0.881
DiaRW	0.902	0.913	0.772	0.891	–
GraphGAN	0.767	0.866	–	–	–
HuGE	**0.938**	**0.951**	0.812	**0.901**	**0.897**

(HOPE [8] and ProNE [11]), and a recent graph neural networks-based technique (GraphGAN [15]). In all the experiments, we choose the best embedding by parameter tuning for each method.

B. Link Prediction

Link prediction refers to the task of predicting the existence of a link between two nodes in a graph. To perform the link prediction task for a given graph G, we first randomly remove 50% of its edges as a positive edge set, and the rest as a training set. We also provide a negative edge set where the randomly selected edges are not in G. It should be noted that the size of the negative set is the same as that of the positive set, and thus the two sets form the task testing set. For a pair of nodes (u, v), $\varphi(u)$ and $\varphi(v)$ are the node representation vectors learned by embedding methods. Here the similarity score for u and v is measured in terms of the inner product $\varphi(u) \cdot \varphi(v)$, along with the AUC (Area Under Curve) metric, a value that is the higher the better, to evaluate the task performance. We repeat this procedure 50 times to offset the randomness of edge removal and report the estimated AUC.

Table II shows AUC for all the methods on five real-world graphs, respectively, where a "–" indicates that the method fails due to the limitation of computing resources or because its running time exceeds 5 days. HuGE outperforms all baselines on all graph datasets, except for VERSE on the Youtube dataset where HuGE comes in the second, by a clear margin from 3.1% to 13.7%. We note that VERSE is the best competitor on Youtube and achieves a considerable gain on CA-AstroPh; nevertheless it is mediocre on directed graphs, such as Wiki-Vote and LiveJournal. The study in [12] reveals that VERSE fails to capture the asymmetric transitivity (i.e., direction of edges) in directed graphs because it generates only one embedding vector per node. Among the recently proposed scalable methods ProNE and DiaRW, although ProNE can efficiently handle the large-scale graph Twitter, HuGE outperforms ProNE by an average of 5% on the evaluated graphs.

C. Multi-label Classification

Multi label classification task aims to predict one or more labels for each graph node and has been widely applied in modern applications ranging from text categorization to Bioinformatics. To perform this task, we use embedding vector and a one-vs-rest logistic regression classifier with L2 regularization (using the LIBLINEAR library), and evaluate accuracy by micro-averaged F1 ($Micro - F1$) and micro-averaged F1 ($Macro - F1$).

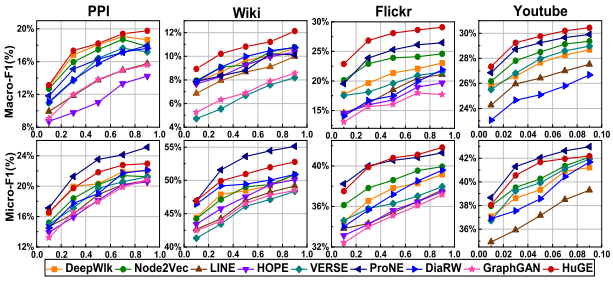

Fig. 4. Performance evaluation of benchmarks for multi-label classification. The x axis is the fraction of labeled data and y axis in the top and bottom rows denote the $Macro - F1$ and $Micro - F1$ scores respectively.

To train a classifier, nodes are randomly split into a training set and a test set, respectively. In the experiments, for each input graph G, we select 10%-90% training ratio on PPI, Wiki and Flickr, and 1%-9% training ratio on Youtube, and the remaining nodes for testing. We report the averaged $Macro-F1$ and $Micro - F1$ scores from 50 trials. Figures 4 show the $Macro-F1$ and $Micro-F1$ scores achieved by each method as a function of the training ratio variation, respectively. We observe that HuGE is comparable to and, in most cases, better than existing popular methods on four real-world graphs. In particular, thanks to its effective heuristic sampling strategies, HuGE consistently outperforms the sampling-based models on all graphs in $Macro - F1$ and $Micro - F1$ scores, gaining 22.1% and 7.3% average improvements, respectively. Note that on the large graph Youtube, HOPE and GraphGAN cannot efficiently handle this graph due to their complex computation operations. On directed graph Wiki, VERSE achieves unsatisfactory performance relative to the comparable gains on undirected graphs, which is consistent with the observation in the evaluation of the link prediction task. ProNE outperforms all competitors in $Micro - F1$ score, but the performance in $Macro - F1$ score is less than impressive. This is because ProNE is specifically designed for node classification by leveraging the spectral propagation technique, and its effectiveness is lower for other tasks, such as the link prediction task.

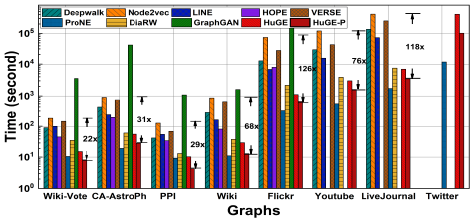

Fig. 5. Node embedding learning time required for each method on eight real-world graphs, where the y axis is in log-scale.

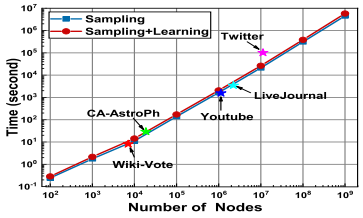

Fig. 6. Scalability of HuGE on synthetic graphs. The lines depict the running time required for sampling (blue line) and both sampling and feature learning (red line), respectively. Pentagrams show the time cost of five real graphs.

D. Efficiency and Scalability

Fig. 5 shows the time required by each method to generate node embedding. We observe that HuGE strikes the best balance between effectiveness and efficiency, and significantly outperforms the sampling-based methods on all graphs, by an average acceleration of $12.8\times$, and the parallel version–HuGE-P is up to $22\times$–$126\times$ faster than all competitors except ProNE. As illustrated in Table II and Fig. 4, ProNE is less effective compared to HuGE for the AUC and $Macro-F1$ score. While ProNE optimizes matrix factorization operation by graph partition techniques to meet the scalability for large-scale graphs, it still exerts a serious burden to DRAM, e.g., it consumes 135GB DRAM on Twitter. Note that GraphGAN and HOPE cannot efficiently handle large graphs, i.e., Youtube, LiveJournal, and Twitter. In particular, GraphGAN is up to 2–3 orders of magnitude slower than HuGE. We also note that in all methods, only HuGE and ProNE can successfully execute on the largest graph Twitter. To further test the scalability of HuGE, we generate synthetic graphs [25] with a fixed node degree of 10 and the number of nodes from 10^2 to 10^9. Fig. 6 presents the running time for sampling and feature learning on the randomly generated graphs with different sizes, suggesting that the running time increases linearly with the size of a graph, and HuGE has the capability of handling billion-node-scale graphs. Moreover, the running time for five real-world graphs is inserted into the plot, which is consistent with the trend of synthetic data.

V. CONCLUSION

In this work, we proposed HuGE, an entropy-driven graph embedding approach with improved efficiency and scalability, which takes into account both the degree of node and the number of common neighbors to measure the proximity of pairwise nodes, and adopts heuristic methods to provide concise and comprehensive representation in the sampling procedure. HuGE is experimentally shown to be more efficient and effective than recent popular graph embedding benchmarks, especially exhibits dozens of times faster for node embedding learning compared with sampling-based models.

ACKNOWLEDGMENT

This work is supported in part by National Defense Preliminary Research Project No. 31511010202, NSFC No. 61772216, No. 61821003, No. U1705261, National Science and Technology Major Project No.2017ZX01032-101, and National Key R&D Program of China No. 2018YFB1003305.

REFERENCES

[1] H. Cai, V. W. Zheng, K. C. Chang. A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. *IEEE TKDE*. 30(9): 1616-1637. 2018.
[2] B. Perozzi, R. Al-Rfou, S. Skiena. Deepwalk: Online learning of social representations. *KDD*. 701-710. 2014.
[3] A. Grover, J, Leskovec. node2vec: Scalable feature learning for networks. *KDD*. 855-864. 2016.
[4] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei. Line: Large-scale information network embedding. *WWW*. 1067-1077. 2015.
[5] L. F. Ribeiro, P. H. Saverese, D. R. Figueiredo, Struc2vec: Learning Node Representations from Structural Identity. *KDD*. 385-394. 2017.
[6] A. Tsitsulin, D. Mottin, P. Karras, E. Müller. VERSE: Versatile Graph Embeddings from Similarity Measures. *WWW*. 539-548. 2018.
[7] Y. Zhang, Z. Shi, D. Feng, X. Zhan. Degree-biased random walk for large-scale network embedding. *Future Generation Computer Systems*. 100: 198-209. 2019.
[8] M. Ou, P. Cui, Jian Pei, Z. Zhang, W. Zhu. Asymmetric Transitivity Preserving Graph Embedding. *KDD*. 1105-1114. 2016.
[9] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, J. Tang. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. *WSDM*. 459-467. 2018.
[10] Y. Yin, Z. Wei. Scalable Graph Embeddings via Sparse Transpose Proximities. *KDD*. 1429-1437. 2019.
[11] J. Zhang, Y. Dong, Y. Wang, J. Tang, M. Ding. ProNE: fast and scalable network representation learning. *IJCAI*. 4278-4284. 2019.
[12] R. Yang, J. Shi, X. Xiao, Y. Yang, S. S. Bhowmick. Homogeneous network embedding for massive graphs via reweighted personalized PageRank. *VLDB*. 13(5). 670-683. 2020.
[13] W. L. Hamilton, R. Ying, J. Leskovec.Inductive representation learning on large graphs. *NIPS*. 1024-1034. 2017.
[14] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, et al. Graph attention networks. *ICLR*. 2018.
[15] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xing, M. Guo. Graphgan: Graph representation learning with generative adversarial nets. *AAAI*. 2508-2515. 2018.
[16] R. H. Li, J. X. Yu, J. Liu. Link prediction: the power of maximal entropy random walk. *CIKM*. 2011.
[17] J. Li, Y. Rao, F. Jin, H. Chen, X. Xiang. Multi-label maximum entropy model for social emotion classification over short text. *Neurocomputing*. 247-25, 2016
[18] B. J. Worton. Kernel methods for estimating the utilization distribution in home-range studies. *Ecology*. 70(1): 164-168. 1989.
[19] S. Kullback, R. A. Leibler. On Information and Sufficiency. *Annals of Mathematical Statistics*. 22(1):79-86. 1951.
[20] Çano E., M. Morisio. Quality of Word Embeddings on Sentiment Analysis Tasks. *NLDB*. 2017.
[21] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean. Distributed representations of words and phrases and their compositionality. *NIPS*. 3111-3119. 2013.
[22] P. Fang, J. Gao, F. Fan and L. Yang. Identifying Political "hot" Spots Through Massive Media Data Analysis. *SBP-BRiMS*. 282-290. 2016.
[23] Y. Li, Z. Wu, S. Lin, H. Xie, M. Lv, Y. Xu, J. C. Lui. Walking with Perception: Efficient Random Walk Sampling via Common Neighbor Awareness. *ICDE*. 962-973. 2019.
[24] T. Mikolov, K. Chen, G. S. Corrado, J. Dean. Efficient estimation of word representations in vector space. *ICLR*. 2013.
[25] D. Chakrabarti, Y. Zhan, C. Faloutsos. R-MAT: A recursive model for graph mining. *SDM*. 442-446. 2004.