Heterogeneous Graph Sparsification for Efficient Representation Learning

Chandan Chunduru*, Chun Jiang Zhu*, Blake Gains†, Jinbo Bi†

*Department of Computer Science, University of North Carolina Greensboro, Greensboro, NC, USA
†Department of Computer Science and Engineering, University of Connecticut, Storrs, CT, USA
{c.chunduru, chunjiang.zhu}@uncg.edu
{blake, jinbo.bi}@uconn.edu

Abstract—Graph sparsification is a powerful tool to approximate an arbitrary graph and has been used in machine learning over homogeneous graphs. In heterogeneous graphs such as knowledge graphs, however, sparsification has not been systematically exploited to improve efficiency of learning tasks. In this work, we initiate the study on heterogeneous graph sparsification and develop sampling-based algorithms for constructing sparsifiers that are provably sparse and preserve important information in the original graphs. We have performed extensive experiments to confirm that the proposed method improves time and space complexities of representation learning while achieving comparable, or even better performance in subsequent graph learning tasks based on the learned embedding.

Index Terms—Heterogeneous Graph, Knowledge Graph, Network Representation Learning, Graph Sparsification

I. INTRODUCTION

Heterogeneous graphs (which subsume knowledge graphs) can model complex relationships among multiple types of real-world entities and have been widely used in studying biological and biomedical data. In a heterogeneous graph (e.g., [6], [11], [21]), there can be multiple types of nodes (e.g., some for compounds and others for targets, gene or diseases) and edges (e.g., a compound treats a disease, a compound resembles another compound, or a gene interacts with another gene), making discovery of deep insights (e.g., perform drug repurposing or identify drug-target interactions) become possible. Graph representation learning aims to construct low-dimensional embeddings (vectors) for all nodes in a graph, and has been a standard approach to further knowledge discovery and development on graphs. For instance, both predicting biomedical network links based on their structural and feature information and clustering for drug-target interaction and drug discovery are based on high-quality heterogeneous network representation. Recently, various heterogeneous graph representation learning methods have been proposed, e.g., random-walk based methods Metapath2Vec [4] and HIN2Vec [7], message-passing based graph neural networks HAN [19] and HGT [9], and relation learning based methods TransE [2], ComplEx [18], and RotatE [17]. For a systematic review on heterogeneous graph embedding, readers are referred to a good survey [20].

However, modern heterogeneous graphs can be of very large size in terms of the number of nodes or the number of edges. For instance, the benchmarking knowledge graph Freebase has over 12 millions nodes across 8 node types and about 63 millions edges of 36 edge types [20]. Processing such large-scale heterogeneous graphs imposes high computational and storage demands. We identify two directions to meet the requirements: one approach is to develop efficient heterogeneous embedding algorithms. But this direction may not offset the effects of ever growing big heterogeneous graphs. The other is to sparsify/compress big heterogeneous graphs while approximating some important network property and not significantly impacting the quality of graph embedding. In this work, we focus on the latter topic of sparsifying heterogeneous graphs.

Graph sparsification deals with approximating an arbitrary graph by a sparse graph that usually has a linear number of edges in the number of vertices of the original graph while preserving some important property. Several notions of graph sparsification have been proposed, e.g., graph spanners [12] that approximate the shortest path distance between every pair of vertices, cut sparsifiers [1] that approximately preserve all graph cut values, and spectral sparsifiers [16] that approximate the graph spectrum/eigenvalues. Graph sparsification has been used to improve the computational cost of learning over graphs, e.g., graph semi-supervised learning and spectral clustering [13], [22], [23]. Although heterogeneous graphs can have much larger size since heterogeneity allows more nodes or edges to be included in one network, however, there has been no systematic study on sparsification techniques for heterogeneous graphs yet.

In this paper, we for the first time study the problem of heterogeneous graph sparsification and its algorithmic applications in graph representation learning. We propose an efficient algorithm for constructing provably smaller sparsifiers for heterogeneous graphs, which, when feeding into a heterogeneous graph embedding algorithm, can save significant computational and storage cost. However, interestingly, the quality of the learned representations using sparsifiers is comparable or even better than those based on the original graphs.
We have provided theoretical analysis of the saving in time and space complexities of heterogeneous graph representation learning. Extensive experiments have also been performed to demonstrate that the developed method can improve the efficiency of heterogeneous graphs while not significantly affecting the accuracy of downstream machine learning tasks based on the learned embedding.

Related Work. Sparsification over homogeneous graphs has been extensively studied, e.g., [1], [12], [16] and used in machine learning [3], [13], [24]. In heterogeneous graphs, however, how to sparsify various types of vertices and edges while preserving graph properties has received limited studies [10], [20]. Jiang et al. [10] developed self-supervised pre-training methods for heterogeneous graphs with relation-based sparsification for efficient pre-training. But their focus is on designing both the node- and schema-level pre-training tasks to preserve semantic and structural properties, instead of a comprehensive study on the sparsification method. To the best of our knowledge, this work is the first systematic study on heterogeneous graph sparsification and its use in representation learning.

II. HETEROGENEOUS GRAPH SPARSIFIERS

In this section, we develop a sampling-based sparsification technique for heterogeneous graphs, analyze its computational and space complexities, and then discuss other potential methods.

Notations and Definitions. A heterogeneous graph can be defined as $G(V, E, \phi, \pi, X, R)$ where $V$ is the vertex set, $E$ is the edge set, $\phi$ specifies the type for each vertex, $\pi$ assigns the type of each edge, and $X$ and $R$ include the node and edge features respectively. $\phi$, $\pi$, $X$, and $R$ can be omitted from the presentation if they are clear from the context. Let $E_{\text{Out}}(u)$ be the set of $u$’s out-edges and $E_{\text{Out}}^t(u)$ be the set of $u$’s out-edges of type $t$. Similarly, let $E_{\text{In}}(u)$ be the set of $u$’s in-edges and $E_{\text{In}}^t(u)$ be the set of $u$’s in-edges of type $t$. Let $|G(V, E)| = |E|$.

Motivation. Graph sparsification in general/homogeneous graphs has received considerable attention and many sparsification algorithms have been developed. However, they can be separated into two types, importance based sampling methods and heuristic based sampling methods. In the former, graph edges are sampled according to some important measure. For example, spectral sparsifiers and cut sparsifiers can be obtained by non-uniformly sampling each edge based on its effective resistance and edge connectivities [8], [15], respectively. This type of approaches is often sound and has nice theoretical guarantees. However, such a design for heterogeneous graphs can be highly challenging considering the complexity in defining an importance measure across different types of edges and computing that measure in large-scale heterogeneous graphs.

In the latter, sampling is performed on all edges independently using only heuristics instead of complex importance measures. For example, $k$-neighbors sparsifiers are constructed by randomly sampling at most $k$ edges for each vertex where $k > 0$ is a tuneable parameter. Though simple, heuristic based sparsifiers have performed practically good, e.g., comparable to spectral sparsifiers in machine learning tasks over graphs [13]. In this paper, we will work towards sparsifying heterogeneous graphs using simple yet effective heuristics.

Details. As a warm-up, we consider the simple method of sampling a fixed number of edges and including in the sparsifier. Since node degrees can be quite different for different nodes, it is possible that a node of low degree has none of its edges sampled and become isolated in the sparsifier. Unfortunately, some heterogeneous embedding algorithms do not allow isolated nodes (through reporting errors such as [14], [19]) and thus this type of sparsifiers is not robust to embedding algorithms. Furthermore, the number of edges of different types for a node can be very different because of their semantic difference. Consider an extreme case: a vertex $u$ with $10K$ edges of type 1 and only one edge of type 2. The information carried by the edge of type 2 is supposed to be greater than that in an individual edge of type 1. When all edges are considered together, the single edge of type 2 can be easily sampled away, resulting in information loss.

We propose to perform random sampling within each type of edges and then aggregate all sampled edges as the sparsifier. In this way, the isolation problem is avoided and the distribution of edge types cannot affect the quality of the constructed sparsifier. Specifically, we sparsify the neighborhood of each vertex according to ascending order of vertex degree. Since heterogeneous graphs are often directed, we sparsify out-edges and in-edges of a vertex separately. Consider out-edges $E_{\text{Out}}(u)$ of vertex $u$. For each edge type $t$ in $E_{\text{Out}}(u)$, if the number of edges of type $t$ is at most $k$, we include all of them to the sparsifier $H$. Here $k$ is an input parameter to control the size of the output sparsifier. Otherwise, we first calculate the number of edges in $E_{\text{Out}}^t(u)$ that are already included in the current $H$. Note that we process vertices one by one and the processing of some vertex before the current vertex may already add a few edges in $E_{\text{Out}}(u)$ to $H$. Let $x$ be the number of such edges, $x = |E_{\text{Out}}^t(u) \cap H|$. Then we sample $k - x$ edges from $E_{\text{Out}}^t(u) - H$ uniformly at random and include the sampled edges in $H$. Since in heterogeneous graphs there can be many edges without edge weight, we keep the weight of a sampled edge unchanged, if any. The processing for in-edges is symmetric. The algorithms are summarized in Algorithms 1 and 2.

The computational and space complexities of Algorithm 1 are $O(ktn)$ and the size (number of edges) of the resulting
sparsifier is also $O(ktn)$, where $t$ is the total number of edge types. Let the computational complexity of a machine learning task in a heterogeneous graph $G$ be $T'(G)$. It is generally believed to be a function of the number of edges $m = |E|$, i.e., $T'(G) = T(m)$. Although there are few studies on the time complexity of learning in heterogeneous graphs, in homogeneous graphs, the time complexities of graph-based learning, e.g., supervised learning such as Laplacian smoothing, semi-supervised learning, and spectral clustering, have shown to be linear in the number of edges $[3],[24]$. We believe that similar relations appear in heterogeneous graphs. Based on this assumption, our approach can reduce the computational cost from $T(m)$ to $O(ktn) + T(ktn)$. The improvement is significant because $m \gg n$ usually holds for heterogeneous graphs, and $k$ and $t$ are only a small value. This analysis is applicable to the space complexity as well. We will confirm the improvement through empirical studies in Section III.

Discussion. We develop another method that also does not suffer from the isolation problem. The principle is to first randomly select, for every edge type $t$ of vertex $u$, one edge from $E_{Out}(u)$ (and $E_{In}(u)$, respectively) to include in the sparsifier. Let the number of edges included be $y$. We then randomly keep $k - y$ edges out of all $u$’s edges, regardless of edge types. This method guarantees that each edge type $t$ of $u$’s edges has at least one sampled edges and thus does not lose important information. This algorithm constructs sparsifiers of size $O(\max(k, t) \cdot n)$, which is smaller than Alg. 1 for the same value of $k$. Consider sparsifiers of similar size produced by both methods, their empirical performance in subsequent machine learning tasks are comparable. As will be shown in Section III, sparsifiers produced by Alg. 1 generally performs better in the link prediction task while sparsifiers output by this method works better in node classification, although the margins are relatively small.

III. EXPERIMENTS

In this section, we perform extensive experiments to evaluate the proposed algorithms for sparsifying heterogeneous graphs and performance of the constructed sparsifiers in representation learning. The algorithms were implemented in Python, and all experiments were performed using 3 cores in a compute node, which is equipped with 24 cores, 2.50 GHz Intel processors, and 256GB RAM.

Experimental Setting. We use three different real-world datasets provided by the benchmark of [20] and their statistics are summarized in Table I. PubMed is a network of genes, diseases, chemicals, and species from NIH PubMed project (not the traditional publication network). Yelp is a review network of businesses, users, locations, and reviews. Freebase is a knowledge base of books, films, music, sports, people and so on. Since this is the first work on heterogeneous graph sparsification for representation learning, we also implement the algorithm in the discussion for comparison, which is denoted as All-Types (ALT).

The tested graphs are first processed by a sparsification algorithm to get sparsified graphs, which are then fed into a representation learning algorithm to get node embedding. To show the robustness of the proposed sparsification, five diverse algorithms including random-walk based HIN2Vec [7] and AspEm [14], message-passing based HAN [19] and HGT [9], and relation learning based ComplEx [18], are employed to generate the embedding. Finally, link prediction and node classification tasks are performed based on the learned embedding following experimental setup in the benchmark [20]. In link prediction, $20\%$ links are held out and the remaining $80\%$ links are used as the training set. A linear Support Vector Machine (LinearSVC) [5] is trained on the training links. For node classification, we train a LinearSVC based on the learned embedding on $80\%$ of the labeled nodes and then predict on the remaining $20\%$.

The efficiency for graph learning tasks are measured by the training time in seconds and memory consumption in term of the maximal residual size during model training. The effectiveness of link prediction is measured by the $AUC$ (area under the ROC curve) and $MRR$ (mean reciprocal rank). The effectiveness of node classification is measured by $macro-F1$ (across all labels) and $micro-F1$ (across all nodes). These measures are averaged over five runs and reported together with their standard deviation.

Baseline Setting. In the baseline setting, we focus on unattributed representation learning that does not exploit node attributes and use the embedding size 50. We first show effectiveness of the developed algorithm in reducing the size of heterogeneous graphs, and then compare performance of unattributed representation learning that does not exploit node attributes and use the embedding size 50. We first show effectiveness of the developed algorithm in reducing the size of heterogeneous graphs, and then compare performance of

| Dataset   | $n$  | $m$  | $m/n$ | Node Types | $t$ |
|-----------|------|------|-------|------------|-----|
| PubMed    | 63,109 | 236,458 | 3.7 | 4 | 10 |
| Freebase  | 12,164,758 | 62,058,902 | 5.1 | 8 | 36 |
| Yelp      | 82,465 | 16,274,179 | 197 | 4 | 4  |

TABLE I: Statistics of the Tested Real-world Networks

| $k$ | PubMed | Freebase | Yelp |
|-----|--------|----------|------|
| 1   | 0.45   | 0.39     | 0.15 |
| 2   | 0.59   | 0.55     | 0.28 |
| 3   | 0.67   | 0.65     | 0.42 |
| 5   | 0.75   | 0.73     | 0.59 |
| 10  | 0.86   | 0.81     | 0.81 |

TABLE II: The Sparsification Ratio ($|H|/|G|$)
learning tasks in the sparsified graphs and the full graph. Table II shows the sparsification ratio, i.e., the ratio of the size of a sparsifier $H$ and the size of the full graph $G$, for varied values of $k$. It can be seen that our method consistently reduces the number of edges in the sparsifiers. The reduction becomes small with an increase in the value of $k$ since more edges are kept in the sparsifiers then. In general, the empirical performance matches our theory that the constructed sparsifiers have size $O(ktn)$.

Figure 1 plots performance of link prediction and node classification when applying different representation learning algorithms in the sparsified graphs and the full graph. The training times for the sparsifiers in the PubMed dataset are significantly improved to lower than 70% of the training time required in the full graph except for AspEm, as shown in Figure 1a. The training times for HIN2Vec even drop to lower than 44% for all values of $k$. Interestingly, the reductions in

---

1HAN reported out-of-memory errors in Yelp and Freebase and AspEm did not complete within 2 weeks in Freebase.
training times are larger than the sparsification ratio in Table II, e.g., 85% when \( k = 10 \). It means that the reductions in computational time are remarkably larger than those in the graph size. The reductions in larger graphs such as Yelp are even greater and drop to lower than 60% of the original time. The running time of ComplEx are consistently lower than 40% while maintaining a good performance as will be shown later. The improvement in memory consumption during model training is not that obvious as the training time in Figure 1b. This may be due to that the additional overhead in the training excluding loading the graph data is comparable to the size of PubMed. But the improvement of memory consumption in larger graphs Yelp and Freebase become much clearer as in Figures 1f and 1j and can be lower than 70% of the original consumption.

Importantly, as in Figures 1c, 1g, and 1k, the performance AUC and MRR of link prediction are not significantly affected by the sparsification, especially for large values of \( k \). Interestingly, the performance of ComplEx and HIN2Vec for quite a few settings can be even better than that based on the full graph. We analyze that it could be because noises in the original graph are also removed during the sparsifying process. Among the tested embedding algorithms, ComplEx consistently achieved the highest AUC and MRR values in different datasets. In Figures 1d, 1h, and 1l, the F1 values are generally comparable to the values in the full graph, and even better in some settings. Furthermore, graph (edge) sparsification appears to have more impacts on the performance of link prediction than on node classification. This seems to suggest that the task of node classification depends less on the edge connectivity compared to link prediction.

Due to superior performance of ComplEx, we choose it as the default embedding model for comparing Alg. 1 and ALT. Figure 2d plots the performance of ComplEx based on the two types of sparsified graphs with respect to different sparsification ratios. It can be seen that AUC and MRR values of Alg. 1 are slightly better than those of ALT, while both F1 values of ALT are larger by a small margin. In practice, users
may select a sparsification method that is most suitable for their target application.

**Additional Experiments.** We first vary the size of the learned embedding from 10 to 200 ([10, 25, 50, 75, 100, 200]) in PubMed. We observe that improvements in the training time and memory consumption are always significant, similar to the baseline setting. As expected, with an increase in the embedding size, the AUC and MRR are gradually improved due to a better learned embedding, as in Figures 2b, 2a and 2c. For HIN2Vec, multiple sparsifies achieve higher AUC/MRR than the full graph, those red lines in Figure 2a. For HAN and ComplEx, the performance of the sparsifies with larger values of $k$, e.g., $k = 10$, are comparable to that of the full graph as shown in Figures 2b and 2c.

Furthermore, since HAN and HGT can incorporate node attributes into the embedding learning, we perform attributed model training by taking node attributes into consideration. The training times and memory consumption in the attributed setting become slightly larger due to the processing of more information. But they are still much lower than training times and memory consumption in the full graph, as shown in Figures 2e and 2g. For HAN, the accuracy performance in attributed graphs (red and black lines) are better than in unattributed graphs, especially for $F1$ values as shown in Figure 2f. HGT appears to be strong in exploiting graph connectivity to generate accurate prediction even without the attribute information, as in Figure 2h. Consistent with the baseline setting, sparsification always brings great benefits in reducing computational resources while producing good accuracy performance.

**IV. Conclusion and Future Work**

In this work, we study the problem of sparsifying heterogeneous graphs while well approximating the original graphs for subsequent graph learning tasks. We develop an efficient sampling method which takes the distribution of different types of edges into consideration and perform computational and storage complexity analysis. Extensive experiments validate the efficiencies of our proposed algorithms while attaining promising performance in downstream graph learning tasks. As the future work, we will study sophisticated algorithms for sparsifying heterogeneous graphs, e.g., by sampling edges according to edge (betweenness) centrality measures and keeping only important edges with high centrality. It is also an interesting direction to develop deep learning models to perform end-to-end heterogeneous graph sparsification and graph learning tasks.

**Acknowledgment**

We thank graduate students Chandra Malgari and Divya Sanga for their help in the experiments and anonymous reviewers for their constructive comments.

**References**

[1] A. Benczur and D. Karger. Approximating s-t minimum cuts in $O(n^2)$ time. In *Proceedings of STOC Conference*, pages 47–55, 1996.

[2] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[3] D. Calandriello, I. Koutis, A. Lazaric, and M. Valko. Improved large-scale graph learning through ridge spectral sparsification. In *Proceedings of ICML Conference*, pages 688–697, 2018.

[4] Y. Dong, N. V. Chawla, and A. Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135–144, 2017.

[5] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[6] S. Fan, J. Zhu, X. Han, C. Shi, L. Hu, B. Ma, and Y. Li. Metapath-guided heterogeneous graph neural network for intent recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2478–2486, 2019.

[7] T.-Y. Fu, W.-C. Lee, and Z. Lei. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1797–1806, 2017.

[8] W. Fang, R. Haribaran, N. J. Harvey, and D. Panigrahi. A general framework for graph sparsification. In *Proceedings of STOC Conference*, pages 71–80, 2011.

[9] Z. Hu, Y. Dong, K. Wang, and Y. Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, pages 2704–2710, 2020.

[10] X. Jiang, T. Jia, Y. Fang, C. Shi, Z. Lin, and H. Wang. Pre-training on large-scale heterogeneous graph. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 756–766, 2021.

[11] D. N. Nicholson and C. S. Greene. Constructing knowledge graphs and their biomedical applications. *Computational and structural biotechnology journal*, 18:1414–1428, 2020.

[12] D. Peleg and A. Schaffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

[13] V. Sadhanala, Y.-X. Wang, and R. J. Tibshirani. Graph sparsification approaches for Laplacian smoothing. In *Proceedings of AISTATS Conference*, pages 1250–1259, 2016.

[14] Y. Shi, H. Gui, Q. Zhu, L. Kaplan, and J. Han. Aspem: Embedding learning by aspects in heterogeneous information networks. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 144–152. SIAM, 2018.

[15] D. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[16] D. Spielman and S.-H. Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

[17] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

[18] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR, 2016.

[19] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, pages 2022–2032, 2019.

[20] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han. Heterogeneous network representation learning: A unified framework with survey and benchmark. *IEEE Transactions on Knowledge and Data Engineering*, 32, 2020.

[21] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 793–803, 2019.

[22] C. Zhu, Q. Liu, and J. Bi. Communication Efficient Distributed Hyper-graph Clustering. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 2131–2135, 2021.

[23] C. Zhu, Q. Liu, and J. Bi. Spectral Vertex Sparsifiers and Pair-Wise Spanners over Distributed Graphs. In *Proceedings of ICML Conference*, pages 12890–12900, 2021.

[24] C. Zhu, S. Storandt, K.-Y. Lam, S. Han, and J. Bi. Improved dynamic graph learning through fault-tolerant sparsification. In *Proceedings of ICML Conference*, pages 7624–7633, 2019.