ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10096660

SEMI-SUPERVISED GRAPH ULTRA-SPARSIFIER USING REWEIGHTED ℓ_1 OPTIMIZATION

Jiayu Li[1] *Tianyun Zhang*[2*] *Shengmin Jin*[3†] *Reza Zafarani*[1*]

[1] Data Lab, Syracuse University [2] Cleveland State University [3] Amazon USA

ABSTRACT

Graph representation learning with the family of graph convolution networks (GCN) provides powerful tools for prediction on graphs. As graphs grow with more edges, the GCN family suffers from sub-optimal generalization performance due to task-irrelevant connections. Recent studies solve this problem by using graph sparsification in neural networks. However, graph sparsification cannot generate ultra-sparse graphs while simultaneously maintaining the performance of the GCN family. To address this problem, we propose *Graph Ultra-sparsifier*, a semi-supervised graph sparsifier with dynamically-updated regularization terms based on the graph convolution. The graph ultra-sparsifier can generate ultra-sparse graphs while maintaining the performance of the GCN family with the ultra-sparse graphs as inputs. In the experiments, when compared to the state-of-the-art graph sparsifiers, our graph ultra-sparsifier generates ultra-sparse graphs and these ultra-sparse graphs can be used as inputs to maintain the performance of GCN and its variants in node classification tasks.

Index Terms— Graph sparsifier, graph neural network, reweighted optimization

1. INTRODUCTION

Inspired by the major success of neural networks in computer vision, graph neural networks (GNNs) [1] have been proposed for addressing various graph-based problems. There are two main types of GNNs: *spectral-based* methods and *spatial-based* methods. A well-established example from the spectral methods is the Graph Convolutional Network (GCN) [2], a semi-supervised model that takes the whole adjacency matrix as input in each neural network layer. Similarly, SGC [3] and SSGC [4], the variants of GCN, reduce the excess complexity of GCN by repeatedly removing the nonlinearities between layers and collapsing the resulting layers (i.e., functions) into a single linear transformation. However, task-irrelevant edges in graphs lead to the GCN family suffering from sub-optimal generalization performance [5].

Graph sparsification [6, 7] can solve the problem of task-irrelevant edges. The aim of graph sparsification is to find a sparse subgraph G_s from a graph G such that the sparsified subgraph G_s can serve as an approximation of the graph G in numerical computations for graph-based applications. For example, cut sparsifiers [8] ensure that the total weight of cuts in a sparsified graph approximates that of cuts in the original graph within some bounded distance. Spectral sparsifiers [9] can guarantee that a sparsified graph preserves the spectral properties of a graph Laplacian. Recent studies [10, 11, 12, 13] have shown that by integrating neural networks with different graph sparsifiers, one can improve node classification performance. However, to guarantee performance in GCN family, these studies only yield graphs with limited sparsity.

To address this problem, we propose *Graph Ultra-Sparsifier* (GU), which generates ultra-sparsified graphs, and the ultra-sparse graphs can be used as inputs in the GCN family without loss of accuracy (and at times, even improving) in the node classification task. GU achieves its performance by considering both (1) graph sparsification and (2) maintaining the graph filter. First, graph sparsification should reduce the ℓ_0 norm of the adjacency matrix. However, ℓ_0 norm is non-convex and discrete. To solve this issue, we approximate the solution to the ℓ_0 problem by solving the reweighted ℓ_1 problem [14, 15]. Second, we minimize the effect of graph sparsification on the graph filter in the graph convolutional network. Graph convolution on a graph G can be represented by $(\mathbf{X} * \mathbf{f})_G = \mathbf{U}((\mathbf{U}^T\mathbf{f}) \odot (\mathbf{U}^T\mathbf{X})) = \mathbf{U}\hat{\mathbf{f}}\mathbf{U}^T\mathbf{X}$, where $\hat{\mathbf{f}} = \mathbf{U}^T\mathbf{f}$. For graph signal $\mathbf{X}$, we have $\mathbf{X} = \mathbf{UC}$, where $\mathbf{U} = (u_1, \ldots, u_N)$ is the basis signal and $\mathbf{C} = (c_1, \ldots, c_N)$ is the coefficients for $\mathbf{U}$. Therefore, given a graph G with an adjacency matrix $\mathbf{A}$, we have the graph convolution $(\mathbf{X} * \mathbf{f})_G = \mathbf{U}\hat{\mathbf{f}}\mathbf{C}$. In the GCN family, $\hat{\mathbf{f}} = \Lambda$, where $\Lambda = \text{diag}(\lambda_i)$ is the eigenvalue matrix of the graph filter $\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}$, $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\widetilde{\mathbf{D}}$ is the degree matrix from $\widetilde{\mathbf{A}}$. We have $\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}} = \mathbf{I} - \widetilde{\mathbf{L}}$, where $\widetilde{\mathbf{L}}$ is a Laplacian matrix with eigenvalues $\Delta = \text{diag}(\delta_i)$. Hence, $(\mathbf{X} * \mathbf{f})_G = \mathbf{U}(\mathbf{I} - \Delta)\mathbf{C} = \sum_i (1 - \delta_i)c_i u_i$. In the process of generating a sparse graph, the graph sparsifier updates the eigenvalues δ_i to the eigenvalues δ_i'; hence, the filter function of the sparse graph will be affected, leading to performance drop in GCN family. Hence, the proposed graph

*Corresponding authors

†The work was done prior to the author joining Amazon

ultra-sparsifier aims to guarantee $\sqrt{\sum_i(\delta_i' - \delta_i)^2} \leq \|\mathbf{E}\|_F$, where $\mathbf{E}$ is a Hermitian matrix. We prove that the Hermitian matrix exists and that we can minimize $\|\mathbf{E}\|_F$ to maintain the filter function in the graph sparsification process.

To evaluate GU on multiple real-world graphs, we compare its performance with the state-of-the-art graph sparsifiers, such as Spectral Sparsifier [16], SGCN [12], and DropEdge [11]. We demonstrate that GU can generate ultra-sparse graphs while the GCN family can maintain the node classification performance when using these ultra-sparsified graphs from GU. In sum, our contributions can be summarized as:

- We propose *Graph Ultra-Sparsifier* (GU), a semi-supervised graph sparsifier that can be integrated with the GCN family;
- We prove that there exists a Hermitian matrix $\mathbf{E}$, which can be minimized to maintain the performance of the graph filter in the graph sparsification process; and
- We show that GU yields ultra-sparse graphs, and the performance of the GCN family is maintained (or improved) when using such graphs.

2. A SEMI-SUPERVISED GRAPH SPARSIFIER

2.1. Problem Definition

Given an undirected graph $G = (\mathbf{V}, \mathbf{K})$ with adjacency matrix $\mathbf{A}$, nodes $\mathbf{V} = \{v_1, \ldots, v_n\}$ and edges $\mathbf{K} = \{k_1, \ldots, k_m\}$, let $n = |\mathbf{V}|$ denote the number of nodes and $m = |\mathbf{K}|$ denote the number of edges. Graph ultra-sparsifier aims to reduce the number of edges m of the original graph G to m' in a subgraph G_s ($m' < m$), such that subgraph G_s, when used as input to the GCN family, results in classification performance similar to that of the original graph G in GCNs.

2.2. Graph Sparsification

2.2.1. Problem Formulation

The output of the GCN family is a function of $\mathbf{A}$ and the weight matrix $\mathbf{W}$. For semi-supervised multi-class classification, loss function of the GCN family with N layers related to $\mathbf{A}$ is the cross-entropy error over labeled examples:

$$f(\{\mathbf{A}_i\}_{i=1}^N, \{\mathbf{W}_i\}_{i=1}^N) = -\sum_{l\in\mathcal{Y}_L}\sum_f \mathbf{Y}_{lf}\ln(\mathbf{Z}_{lf}), \quad (1)$$

where $\mathcal{Y}_L$ is the set of node indices that have labels, $\mathbf{Y}_{lf}$ is a matrix of labels, and $\mathbf{Z}_{lf}$ is the output of the GCN family. When fixing the weight matrices in the GCN family, the ultra-sparsifier aims to train the adjacency matrix and reduce the number of non-zero elements in the adjacency matrix while maintaining the accuracy of the GCN family. Therefore, with pretrained models of the GCN family, we use $f(\{\mathbf{A}\}_{i=1}^N)$ to present the loss function (Eq. 1) and aim to minimize the summation of the loss function with the ℓ_0 regularization term:

$$\underset{\{\mathbf{A}_i\}}{\text{minimize}} \quad f(\{\mathbf{A}_i\}_{i=1}^N) + \lambda\sum_{i=1}^N \|\mathbf{A}_i\|_0, \quad (2)$$

where λ is the penalty parameter to adjust the relative importance of accuracy with respect to sparsity.

2.2.2. Problem Solution

The problem in Eq. (2) with the ℓ_0 norm is intractable. Therefore, we use a reweighted ℓ_1 method [14] to approximate the ℓ_0 norm. With the reweighted ℓ_1 method, we instead solve the following problem:

$$\underset{\{\mathbf{A}_i\}}{\text{minimize}} \quad f(\{\mathbf{A}_i\}_{i=1}^N) + \lambda\sum_{i=1}^N h(\mathbf{H}_i^{(l)}, \mathbf{A}_i), \quad (3)$$

where l represents the number of iterations of the reweighted method and $h(\mathbf{H}_i^{(l)}, \mathbf{A}_i) = \|\mathbf{H}_i^{(l)} \odot \mathbf{A}_i\|_{\ell_1}$, where the operator $\odot$ denotes the Hadamard product. $\mathbf{H}_i^{(l)}$ is the collection of penalties on an adjacency matrix of one layer in the GCN family, which is updated in every iteration to increase the degree of sparsity beyond the ℓ_1 norm regularization. In each iteration, we update the solution of $\mathbf{A}_i$ by $\mathbf{A}_i^{(l)}$ using gradient descent and update $\mathbf{H}_i^{(l+1)}$ with the following equation.:

$$\mathbf{H}_i^{(l+1)} = \frac{1}{|\mathbf{A}_i^{(l)} + \epsilon|}, \quad (4)$$

where $|\cdot|$ denotes the absolute value, and ϵ is a small parameter to avoid dividing by zero. All operations in Eq. (4) are element-wise.

2.3. Maintaining the Graph Filter

For maintaining the graph filter, given a Laplacian matrix $\widetilde{\mathbf{L}}$ from a graph G with eigenvalues δ_i, and a Laplacian matrix $\hat{\mathbf{L}}$ with eigenvalues δ_i' after updating the graph G, we first prove that the difference between δ_i's and δ_i''s can be bounded by $\|\mathbf{E}\|_F = \|\widetilde{\mathbf{L}} - \hat{\mathbf{L}}\|_F$. Then, we prove that the Laplacian matrix $\hat{\mathbf{L}}$ exists; hence, we can minimize the upper bound $\|\mathbf{E}\|_F$ to maintain the graph filter in the graph sparsification process.

Theorem 1. *There exists a Hermitian matrix $\boldsymbol{E} = \hat{\boldsymbol{L}} - \widetilde{\boldsymbol{L}}$ such that the eigenvalues δ_i and the updated eigenvalues δ_i' satisfy $\sqrt{\sum_i(\delta_i' - \delta_i)^2} \leq \|\boldsymbol{E}\|_F$.*

Proof. In the GCN family, $(\mathbf{X} * \mathbf{f})_G = \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X} = \mathbf{U}\Lambda\mathbf{U}^T\mathbf{U}\mathbf{C}$. We can define $\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}$ as $\mathbf{g}_\lambda$, which is called graph filter in [17]. After sparsifying a graph, the filter is updated to $\mathbf{g}_{\lambda'} = \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}$. The difference between two

filters can be written as

$$\begin{aligned}\mathbf{g}_\lambda - \mathbf{g}_{\lambda'} =& \mathbf{U}\Lambda\mathbf{U}^T - \mathbf{U}\Lambda'\mathbf{U}^T \\ =& \mathbf{U}(\mathbf{I}-\Delta)\mathbf{U}^T - \mathbf{U}(\mathbf{I}-\Delta')\mathbf{U}^T \\ =& \mathbf{U}\Delta'\mathbf{U}^T - \mathbf{U}\Delta\mathbf{U}^T \\ =& \hat{\mathbf{L}} - \widetilde{\mathbf{L}},\end{aligned} \tag{5}$$

where $\hat{\mathbf{L}}$ and $\widetilde{\mathbf{L}}$ are Laplacian matrices with eigenvalues $\Delta' = \text{diag}(\delta_i')$ and $\Delta = \text{diag}(\delta_i)$, respectively. Based on the Hermitian analogue of Hoffman-Wielandt theorem [18], given $\mathbf{E} = \mathbf{g}_\lambda - \mathbf{g}_{\lambda'} = \hat{\mathbf{L}} - \widetilde{\mathbf{L}}$, we can have $\sqrt{\sum_i(\delta_i' - \delta_i)^2} \leq \|\mathbf{E}\|_F$, where $\mathbf{E}$ is also a Hermitian matrix. □

Next, we prove that given a trainable Hermitian matrix α, there exists an updated Laplacian matrix $\hat{\mathbf{L}}$ related to a graph G with Laplacian matrix $\widetilde{\mathbf{L}}$.

Theorem 2. *There exists a Hermitian matrix α such that we can have the updated Laplacian matrix $\hat{\boldsymbol{L}} = (\boldsymbol{I}+\alpha)\widetilde{\boldsymbol{L}} + \alpha\widetilde{\boldsymbol{D}}$, satisfying $\sqrt{\sum_i(\delta_i' - \delta_i)^2} \leq \|\boldsymbol{E}\|_F = \|\alpha(\widetilde{\boldsymbol{L}} + \widetilde{\boldsymbol{D}})\|_F$*

Proof. Rayleigh quotient for a given complex Hermitian matrix $\mathbf{M}$ and nonzero vector $\mathbf{x}$ is defined as $R(\mathbf{M},\mathbf{x}) = \frac{\mathbf{x}^T\mathbf{M}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$. The Laplacian matrix and the degree matrix can be noted as $\widetilde{\mathbf{L}} = \mathbf{U}\Delta\mathbf{U}^T$ and $\widetilde{\mathbf{D}}$ (degrees are in range $[d_{\min}, d_{\max}]$) are also Hermitian matrices. Let $\mathbf{M} = \widetilde{\mathbf{L}} + \widetilde{\mathbf{D}}$, then

$$\begin{aligned}R(\mathbf{M},\mathbf{x}) =& \frac{\mathbf{x}^T(\widetilde{\mathbf{L}} + \widetilde{\mathbf{D}})\mathbf{x}}{\mathbf{x}^T\mathbf{x}} \\ =& \frac{(\mathbf{U}^T\mathbf{x})^T\Delta(\mathbf{U}^T\mathbf{x}) + \mathbf{x}^T\widetilde{\mathbf{D}}\mathbf{x}}{\mathbf{x}^T\mathbf{x}} \\ =& \frac{\mathbf{P}^T\Delta\mathbf{P} + \mathbf{x}^T\mathbf{D}\mathbf{x}}{\mathbf{x}^T\mathbf{x}} \\ =& \frac{\sum_i \delta_i|p_i|^2 + \sum_i d_i|x_i|^2}{\sum_i |x_i|^2}\end{aligned} \tag{6}$$

According to (6), we have

$$\frac{\delta_1\sum_i|p_i|^2 + d_{\min}\sum_i|x_i|^2}{\sum_i|x_i|^2} \leq R(\mathbf{M},\mathbf{x}) \tag{7}$$

and

$$R(\mathbf{M},\mathbf{x}) \leq \frac{\delta_n\sum_i|p_i|^2 + d_{\max}\sum_i|x_i|^2}{\sum_i|x_i|^2}. \tag{8}$$

As $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ and $\mathbf{I}_{ij} = \sum_i u_{ji}u_{ik}$, $j \neq k$ when $\mathbf{I}_{jk} = 0$ and $j = k$ when $\mathbf{I}_{jk} = 1$. With $\sum_i|p_i|^2 = \sum_j\sum_k(\sum_i u_{ki}u_{ij})x_jx_k$. Therefore, we can have

$$\sum_i|p_i|^2 = \sum_i|x_i|^2. \tag{9}$$

Combining (7) and (8) with (9), we have $\delta_1 + d_{\min} \leq R(\widetilde{\mathbf{L}} + \widetilde{\mathbf{D}}) \leq \delta_n + d_{\max}$, which means the eigenvalues μ of $\widetilde{\mathbf{L}} + \widetilde{\mathbf{D}}$ range from $\delta_1 + d_{\min}$ to $\delta_n + d_{\max}$. In the GCN family, the range of eigenvalues of $\widetilde{\mathbf{L}}$ is $[0, 2)$ while $d_{\min}$ is equal to 1 with adding a self-loop to each node. Therefore, we have $1 \leq \mu < 2 + d_{\max}$ and $\det(\widetilde{\mathbf{L}} + \widetilde{\mathbf{D}}) = \prod_i \mu_i > 0$, which means the Hermitian matrix $\widetilde{\mathbf{L}} + \widetilde{\mathbf{D}}$ can be invertible. Let $\mathbf{E} = \alpha(\widetilde{\mathbf{L}} + \widetilde{\mathbf{D}})$, we can obtain $\alpha = \mathbf{E}(\widetilde{\mathbf{L}} + \widetilde{\mathbf{D}})^{-1}$ and the updated Laplacian matrix $\hat{\mathbf{L}}$ can be represented as $\hat{\mathbf{L}} = (\mathbf{I}+\alpha)\widetilde{\mathbf{L}} + \alpha\widetilde{\mathbf{D}}$ according to Theorem 1. □

Matrices $\widetilde{\mathbf{L}}$ and $\hat{\mathbf{L}}$ are related to $\widetilde{\mathbf{A}}$ and the sparsified $\widetilde{\mathbf{A}}$, respectively. Thus, from Theorems 1 and 2, we can add $\|\hat{\mathbf{L}} - \widetilde{\mathbf{L}}\|_F$ in the Eq.(3), which is equal to finding an optimal α to minimize the upper bound $\|\mathbf{E}\|_F$ for maintaining the filters.

Table 1. Dataset statistics

Dataset	Type	Number of Nodes	Number of Edges
Cora	Citation network	2,708	5,429
Citeseer	Citation network	3,327	4,732
Pubmed	Citation network	19,717	44,338
NELL	Knowledge graph	65,755	266,144

3. EXPERIMENTS

We evaluate the performance of our graph ultra-sparsifier by (a) comparing the sparsity of the sparsified graphs provided by different graph sparsifiers; and (b) the node classification performance of the GCN family with the sparsified graphs.

3.1. Experimental Setup

Datasets. We conduct our experiments on four classical graph datasets, which have been utilized for evaluation in previous studies. `Citeseer`, `Cora`, and `Pubmed` are from [19] while `NELL` is extracted from a knowledge graph introduced by [20]. Table 1 provides the statistics of the datasets.

Baselines. We evaluate our results using node classification "backbone" models from the GCN family including GCN [2], SGC [3] and SSGC [4]. We thoroughly evaluate the performance of different graph sparsifiers on the GCN family using the state-of-the-art sparsifiers: Spectral Sparsifier (SS) [16], SGCN [12], and DropEdge [11]. The SS sparsifies graphs in near linear-time. SGCN sparsifies the graph in GCN by formulating and solving it by ADMM [21]. DropEdge randomly removes a certain number of edges from an input graph at each training epoch.[1]

3.2. Results and Performance Analysis

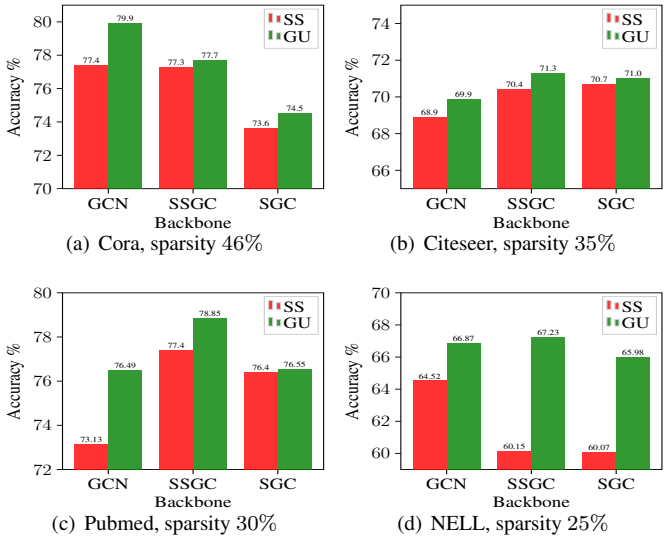
In Table 2, we identify the sparsest graphs (i.e., the maximum sparsity) that can be achieved by the GU such that when those sparse graphs are integrated with a backbone (the GCN family), we can achieve a similar performance to that of the backbone without any sparsifier. We use the same sparsity in

[1]The code and appendix have been released on `https://github.com/Code4Graph/Ultra-sparsifier`.

Table 2. Performance of node classification of the GCN family using sparsified graphs from graph sparsifiers.

Backbone	Sparsifier	Cora		Citeseer		Pubmed		NELL	
		Accuracy	Sparsity	Accuracy	Sparsity	Accuracy	Sparsity	Accuracy	Sparsity
GCN	None	81.5%	0%	70.5%	0%	**78.7%**	0%	**67.7%**	0%
	SGCN	80.5%	15%	69.3%	28%	77.2%	20%	63.8%	20%
	DropEdge	80.6%	15%	68.6%	28%	76.8%	20%	63.0%	20%
	GU (ours)	**81.7%**	15%	**70.7%**	28%	**78.7%**	20%	**67.7%**	20%
SSGC	None	82.4%	0%	73.0%	0%	80.0%	0%	**68.1%**	0%
	SGCN	80.8%	15%	71.6%	20%	78.8%	20%	65.4%	20%
	DropEdge	81.6%	15%	69.6%	20%	77.6%	20%	66.1%	20%
	GU (ours)	**82.6%**	15%	**73.1%**	20%	**80.0%**	20%	**68.1%**	20%
SGC	None	80.7%	0%	71.9%	0%	**77.7%**	0%	**67.9%**	0%
	SGCN	79.2%	15%	71.8%	25%	76.4%	20%	63.8%	20%
	DropEdge	78.7%	15%	71.9%	25%	77.2%	20%	54.3%	20%
	GU (ours)	**80.8%**	15%	**72.1%**	25%	**77.7%**	20%	**67.9%**	20%

other sparsifiers with the GCN family and observe their performance in node classification. Note that we denote backbone without any sparsifier as None in the sparsifier column of Table 2 (i.e., sparsity is equal to 0). On `Cora` dataset, the max sparsity of the graph in GU is up to 15%, and the GCN family using the sparse graphs from GU have the best performance (81.7%, 82.6% and 80.8% accuracy rates). We observe that the GCN family using sparse graphs from other sparsifiers with the same sparsity cannot achieve this performance. When integrated with backbone GCN, SSGC and SGC on `Citeseer` dataset, the GU achieves a max sparsity of up to 28%, 20% and 25%, respectively and helps the backbones outperform other models in the node classification task. On `Pubmed` and `NELL` datasets, even though removing 20% of the edges, GU integrated with the GCN family still achieve the same performance as the GCN family using original graphs while other sparsifiers using the same sparsity with the backbones have worse performance. As in SS sparsifier, we cannot set as input a fix sparsity, for fair comparison, we compare the performance of the GCN family using the sparse graphs from GU and SS with the same sparsity decided by SS. The results in Figures 1(a), 1(b), 1(c) and 1(d) show that the GCN family using sparse graphs from GU have better performance than these backbones using sparsified graphs from SS.



(a) Cora, sparsity 46% (b) Citeseer, sparsity 35%

(c) Pubmed, sparsity 30% (d) NELL, sparsity 25%

Fig. 1. Performance comparison of the GCN family with Graph Ultra-sparsifier (GU) and Spectral Sparsifier (SS) under the same sparsity level given by Spectral Sparsifier (SS).

4. CONCLUSION

Graph sparsification removes task-irrelevant connections in graphs and prevents the GCN family from falling into suboptimal generalization performance. In this paper, we propose *Graph Ultra-sparsifier* (GU), a semi-supervised graph sparsifier that can generate ultra-sparse graphs and can be integrated with the GCN family. In GU, we formulate sparsification as an ℓ_0 optimization problem and approximate the solution of the ℓ_0 problem by solving the reweighted ℓ_1 problem. When sparsifying a graph, GU maintains the graph filter of the GCN family such that the generated ultra-sparse graphs from GU can be used in the GCN family, achieving the same (or better) performance as that of the GCN family using original graphs. Experimental results on real-world datasets demonstrate that the proposed ultra-sparsifier can generate ultra-sparse graphs. Simultaneously, the ultra-sparse graphs can be used as inputs to the GCN family, which achieves the best classification accuracy when compared to that of the GCN family using sparse graphs from other graph sparsifiers.

5. ACKNOWLEDGEMENTS

This research was supported by the National Science Foundation under awards CAREER IIS-1942929.

6. REFERENCES

[1] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[2] Thomas N. Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[3] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the 36th International Conference on Machine Learning*, Kamalika Chaudhuri and Ruslan Salakhutdinov, Eds., Long Beach, California, USA, 09–15 Jun 2019, vol. 97 of *Proceedings of Machine Learning Research*, pp. 6861–6871, PMLR.

[4] Hao Zhu and Piotr Koniusz, "Simple spectral graph convolution," in *International Conference on Learning Representations*, 2021.

[5] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang, "Learning to drop: Robust graph neural network via topological denoising," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, New York, NY, USA, 2021, WSDM '21, p. 779–787, Association for Computing Machinery.

[6] Jiayu Li, Tianyun Zhang, Shengmin Jin, Makan Fardad, and Reza Zafarani, "Adversparse: An adversarial attack framework for deep spatial-temporal graph neural networks," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 5857–5861.

[7] Jiayu Li, Tianyun Zhang, Hao Tian, Shengmin Jin, Makan Fardad, and Reza Zafarani, "Graph sparsification with graph convolutional networks," *International Journal of Data Science and Analytics*, vol. 13, no. 1, pp. 33–46, 2022.

[8] András A. Benczúr and David R. Karger, "Approximating *s-t* minimum cuts in $\tilde{O}(n^2)$ time," in *STOC*, 1996.

[9] Daniel A. Spielman and Shang-Hua Teng, "Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems," 2006.

[10] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang, "Robust graph representation learning via neural sparsification," in *ICML*, 2020.

[11] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *ICLR*, 2020.

[12] Jiayu Li, Tianyun Zhang, Hao Tian, Shengmin Jin, Makan Fardad, and Reza Zafarani, "Sgcn: A graph sparsifier based on graph convolutional networks," in *PAKDD*, 2020.

[13] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang, "A unified lottery ticket hypothesis for graph neural networks," in *Proceedings of the 38th International Conference on Machine Learning*, Marina Meila and Tong Zhang, Eds. 18–24 Jul 2021, vol. 139 of *Proceedings of Machine Learning Research*, pp. 1695–1706, PMLR.

[14] Emmanuel J. Candes, Michael B. Wakin, and Stephen P. Boyd, "Enhancing sparsity by reweighted l1 minimization," 2007.

[15] Tianyun Zhang, Xiaolong Ma, Zheng Zhan, Shanglin Zhou, Caiwen Ding, Makan Fardad, and Yanzhi Wang, "A unified dnn weight pruning framework using reweighted optimization methods," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 2021, p. 493–498, IEEE Press.

[16] Daniel A. Spielman and Shang-Hua Teng, "Spectral sparsification of graphs," *CoRR*, vol. abs/0808.4134, 2008.

[17] Qimai Li, Xiao-Ming Wu, and Zhichao Guan, "Generalized label propagation methods for semi-supervised learning," *CoRR*, vol. abs/1901.09993, 2019.

[18] G. W. Stewart and Ji guang Sun, *Matrix Perturbation Theory*, Academic Press, 1990.

[19] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–106, 2008.

[20] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell, "Toward an architecture for never-ending language learning," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. 2010, AAAI'10, p. 1306–1313, AAAI Press.

[21] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.