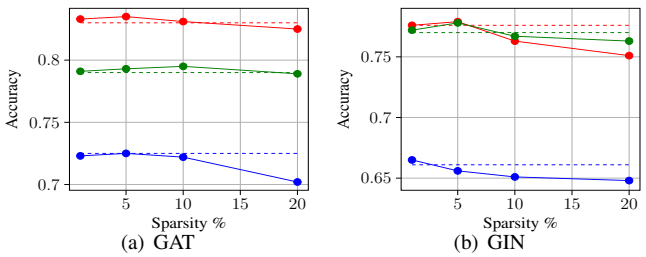
APPENDIX OF ULTRA-SPARSIFIER

Jiayu Li[⋆] *Tianyun Zhang*[†] *Shengmin Jin*[‡] *Reza Zafarani*[⋆]

[⋆] Data Lab, Syracuse University
[†] Cleveland State University
[‡] Amazon USA

I. SPARSIFIER USING REWEIGHTED ℓ_1 OPTIMIZATION WITH NON-SPECTRAL GRAPH NEURAL NETWORKS

In this paper, we use the reweighted ℓ_1 optimization to generate ultra-sparse graphs. Our goal was to derive theorems that can ensure that the ultra-sparse graphs can be constructed as accurate spectral-based filters so that we can guarantee the performance of the GCN family with the filter in node classification tasks. As the most widely used spectral-based filter in the GCN family is related to the symmetrical normalized Laplacian matrix $\widetilde{\mathbf{L}} = \mathbf{I} - \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}$ and $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, we consider the symmetrical normalized Laplacian matrix to maintain the spectral-based filter when sparsifying. The symmetrical normalized Laplacian matrix is widely applied as the spectral-based filter in the GCN family (spectral-based GNN models) such as GCN, SSGC and SGC (the experiment results are shown in our paper). However, non-spectral GNN models do not include the spectral-based filter. Therefore, Theorem 1 and Theorem 2 are not applicable to such non-spectral GNN models. Having said that, we can still apply the reweighted ℓ_1 optimization to sparsify graphs for non-spectral GNN models. Here, we just show the experiment results on Cora and Citeseer datasets using reweighted ℓ_1 optimization based on the GAT and GIN. In Fig. 1, we obtain sparsified graphs and maintain the performance of GAT and GIN by using the sparsified graphs as inputs. As the figures show, we can outperform or remain relatively competitive even with significant sparsification compared to the baselines.



(a) GAT (b) GIN

Fig. 1. Performance comparison of the GAT and GIN only using reweighted reweighted ℓ_1 optimization for graph sparsification. Solid lines represent the performance on the Cora and Citeseer and Pubmed while dash line represents the performance of baselines without sparsification.

II. TRAINING OF ULTRA-SPARSIFIER

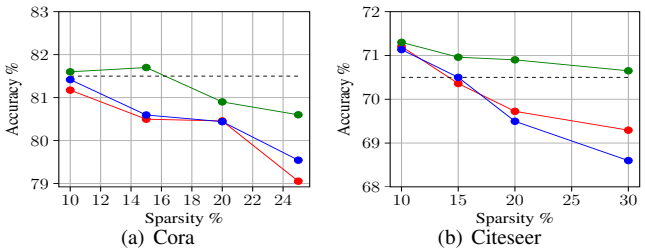
The complete objective function with $f(\{\mathbf{A}_i^{(l)}\}_{i=1}^N)$, $h(\mathbf{H}_i^{(l-1)}, \mathbf{A}_i^{(l)})$ and $\|\hat{\mathbf{L}}_i^{(l)} - \widetilde{\mathbf{L}}\|_F$ can be defined as

$$\underset{\{\mathbf{A}_i\}}{\text{minimize}} \quad f(\{\mathbf{A}_i^{(l)}\}_{i=1}^N) + \|\hat{\mathbf{L}}_i^{(l)} - \widetilde{\mathbf{L}}\|_F + \lambda \sum_{i=1}^{N} h(\mathbf{H}_i^{(l-1)}, \mathbf{A}_i^{(l)}), \tag{1}$$

where $\widetilde{\mathbf{L}}$ and $\hat{\mathbf{L}}_i^{(l)}$ are related to the original adjacency matrix $\mathbf{A}$ (fixed) and the trainable adjacency matrix $\mathbf{A}_i^{(l)}$ (l represents the number of iterations), respectively. Note that $\mathbf{A}_i^{(1)}$ is initialized with $\mathbf{A}$, and $\mathbf{H}_i^{(0)}$ is initialized by all 1s. In the training step, we fix the weights from the pretrained model and apply stochastic gradient descent to update the trainable adjacency matrix $\mathbf{A}_i^{(l)}$ based on Eq.(1). When updating $\mathbf{A}_i^{(l)}$, we ensure that the non-existing edges (disconnected nodes) in the original graph cannot be updated in $\mathbf{A}_i^{(l)}$. A matrix m is defined using the original adjacency matrix $\mathbf{A}$ to mask zero entries, which will not be updated in the gradient descent. The trainable variable $\mathbf{A}_i^{(l)}$ at each step is updated by $\mathbf{A}_i^{(l+1)} = \mathbf{A}_i^{(l)} - \gamma(m \odot \frac{\partial f'(\mathbf{A}_i^{(l)})}{\partial \mathbf{A}_i^{(l)}})$, where γ is the learning rate, and the operator $\odot$ denotes the Hadamard product. For maintaining the symmetry of the trainable adjacency matrix, we set $\mathbf{A}_i^{(l)} = \frac{1}{N}\sum_i^N \frac{\mathbf{A}_i^{(l)} + (\mathbf{A}_i^{(l)})^\intercal}{2}$ at the end of each iteration.

We update $\mathbf{H}_i^{(l+1)}$ with equation $\mathbf{H}_i^{(l+1)} = \frac{1}{|\mathbf{A}_i^{(l)} + \epsilon|}$. The Hermitian matrix α is solved in each optimization step. This is because $\mathbf{E}_i = \hat{\mathbf{L}}_i^{(l)} - \widetilde{\mathbf{L}}$ from Theorem 1 and $\alpha = \mathbf{E}_i(\widetilde{\mathbf{L}} + \widetilde{\mathbf{D}})^{-1}$ from Theorem 2. $\widetilde{\mathbf{D}}$ and $\widetilde{\mathbf{L}}$ are related to the original matrix $\mathbf{A}$ (fixed). $\hat{\mathbf{L}}_i^{(l)}$ is updated in every training step because it is related to the trainable adjacency matrix $\mathbf{A}_i^{(l)}$. Therefore, solving the adjacency matrix $\mathbf{A}_i^{(l)}$ is equal to updating α. We will include a pseudo code to clarify the steps in the revised manuscript.

The entries in the symmetric Laplacian matrix ($\mathbf{I} - \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}$) can be non-integers (floats). The entries of the trainable adjacency matrix can be floats in training, but the entries in the trainable adjacency matrix (corresponding to connections in the original graph) are projected to 0 or 1 after training is completed. For example, when the sparsity is set to 5%, we find the 5th percentile of the values (corresponding to connections in the original graph) in the trainable matrix as the threshold. The values below this threshold are set to 0, and the rest of the values (95% of the values) are set to 1.



(a) Cora (b) Citeseer

Fig. 2. Solid line represents the performance GCN with different sparsifiers inclduing SGCN, DropEdge and GU. **Dash line** represents the performance of GCN without any sparsifiers.

III. ABOUT EXPERIMENTS OF ULTRA-SPARSIFIER

Here we only show accuracy of applying GCN when applying different sparsifiers with various levels of sparsity on the Cora and Citeseer datasets. Fig.2 shows that the proposed GU achieves the best performance. In the table of our paper, we show the ultra-sparse graphs from GU can be used as inputs to maintain the performance of the GCN family in the node classification tasks.

In our paper, the spectral sparsifier with the GCN family performs worse. This is because the spectral sparsifier yields a subgraph of the original whose Laplacian quadratic form is approximately the same as that of the original graph. Therefore, the main purpose of the spectral sparsifier is to preserve the spectral properties rather than focusing on node classification. Moreover, the spectral sparsification method cannot maintain the graph filter of the GCN family in the node classification tasks.