ASGNN: GRAPH NEURAL NETWORKS WITH ADAPTIVE STRUCTURE

Anonymous authors
Paper under double-blind review

ABSTRACT

The graph neural network (GNN) has presented impressive achievements in numerous machine learning tasks. However, many existing GNN models are shown to be extremely vulnerable to adversarial attacks, which makes it essential to build robust GNN architectures. In this work, we propose a novel interpretable message passing scheme with adaptive structure (ASMP) to defend against adversarial attacks on graph structure. Layers in ASMP are derived based on optimization steps that minimize an objective function that simultaneously learns the node feature and the graph structure. ASMP is adaptive in the sense that the message passing process in different layers is able to be carried out over different graphs. Such a property allows more fine-grained handling of the noisy graph structure and hence improves the robustness. Integrating ASMP with neural networks can lead to a new family of GNNs with adaptive structure (ASGNN). Extensive experiments on semi-supervised node classification tasks demonstrate that the proposed ASGNN outperforms the state-of-the-art GNN architectures with respect to classification performance under various graph adversarial attacks.

1 INTRODUCTION

Graphs, or networks, are ubiquitous data structures in many fields of science and engineering (Newman, 2018), like molecular biology, computer vision, social science, financial markets, etc. In the past few years, due to its appealing capability of learning representations through message passing over the graph structure, graph neural network (GNN) models have become popular choices for processing graph-structured data and have achieved astonishing success in various applications (Kipf and Welling, 2017; Bronstein et al., 2017; Wu et al., 2020; Zhou et al., 2020; Wu et al., 2022). However, existing GNN backbones such as the graph convolutional network (GCN) (Kipf and Welling, 2017) and the graph attention network (Veličković et al., 2018) are shown to be extremely vulnerable to carefully designed adversarial attacks on the graph structure (Sun et al., 2018; Jin et al., 2021; Günnemann, 2022). With unnoticeable malicious manipulations of the graph, the performance of GNNs significantly drops and may even be worse than the performance of a simple baseline that ignores all the relational information among data features (Dai et al., 2018; Zügner et al., 2018; Zügner and Günnemann, 2019; Zhang and Zitnik, 2020). With the increasing deployments of GNN models in various real-world applications, it is of vital importance to ensure their reliability and robustness, especially in scenarios, such as medical diagnosis and credit scoring, where a deflected model can lead to dramatic consequences (Günnemann, 2022), making it essential to build robust GNN architectures.

To improve the robustness of GNNs, a natural idea is to "purify" the given graph structure that is potentially noisy. Existing works in this line can be roughly categorized into two categories. The first category of robustifying GNNs can be viewed as a two-stage approach. First, a purified graph is obtained by "pre-processing" the input graph leveraging on information from the node features. Next, a GNN model is trained based on this purified graph. For example, in the GNN-Jaccard method (Wu et al., 2019b), a new graph is obtained by removing the edges with small "Jaccard similarity". In Entezari et al. (2020), observing that adversarial attacks can scale up the rank of the graph adjacency matrix, the authors proposed to use a low-rank approximation of the initial graph adjacency matrix as a substitute. In the second category, taking the graph adjacency matrix in the GNN as an unknown, a purified graph of a parameterized form will be "learned" through optimizing the supervised training loss (Zhu et al., 2022). For example, in Franceschi et al. (2019), the graph adjacency matrix is jointly learned with a GNN in a bilevel optimization way, which can be seen as a full parametrization of the graph adjacency matrix. Moreover, under this full parametrization setting, structural regularizers are adopted in Jin et al. (2020); Luo et al. (2021) as augmentations on the training loss function to promote certain properties of the purified graph. Besides the full parametrization approach, a multi-head weighted cosine similarity metric function (Chen et al., 2020) and a GNN model (Yu et al., 2020) are also used to parameterize the graph adjacency matrix for structure learning.

Going beyond purifying the graph structures to robustify the GNN models, there are also efforts on designing robust GNN architectures. Under the observation that aggregation functions such as sum, weighted mean, or the max operations can be arbitrarily distorted by only a single outlier node, Geisler et al. (2020); Wang et al. (2020); Zhang and Lu (2020) try to design robust GNN models via designing robust aggregation functions. Moreover, some other works apply the attention mechanism (Veličković et al., 2018) to mitigate the influence of the noisy edges. For example, Zhu et al. (2019) consider the representation of node features as Gaussian distribution and use the variance information to determine the attention scores; in Tang et al. (2020), clean graph information and their adversarial counterparts are utilized to train an attention mechanism so that the model can "learn to penalize" the perturbed edges; in Zhang and Zitnik (2020), the authors define an attention mechanism based on neighboring nodes' similarity directly.

Different from the existing approaches to robustify the GNNs, in this work, we propose a novel robust and interpretable message passing scheme with adaptive structure (ASMP). Then, a family of GNNs with adaptive structure (ASGNNs) will be introduced. Based on prior works revealing that the message passing processes in a class of GNNs are actually (unrolled) gradient steps for solving a graph signal denoising problem (Zhu et al., 2021; Ma et al., 2021; Zhang and Zhao, 2022), ASMP is generated by an alternating (proximal) gradient descent algorithm for simultaneously denoising the graph signal and the graph structure. Designed in such a principled way, ASMP is not only friendly to back-propagation training but also achieves the desired structure adaptivity with a theoretical convergence guarantee. Conceptually different from the existing robustified GNNs with *fixed* graph structure, ASGNN interweaves the graph purification process and the message passing process, which makes it possible to conduct message passing over different graph structures at different layers, i.e., in an *adaptive* graph structure fashion. Thus, an edge might be excluded in some layers but included in other layers, depending on the adaptive structure learning process. Such a property allows more fine-grained handling of perturbations than previous graph purification methods that use a single graph in the entire GNN. Once trained, ASMP can be naturally interpreted as a parameter-optimized iterative algorithm. This work falls into the category of GNN architecture designs. To be more specific, the major contributions of this work are highlighted as follows:

- To the best of our knowledge, ASMP is the first message passing scheme with adaptive structure that is designed based on an optimization problem, with thorough interpretations, convergence guarantee, and specifications. ASMP is adaptive in the sense that the graph structure over which the message passing process is conducted varies at different layers.
- Based on the proposed ASMP, a family of GNNs with adaptive structure, named ASGNN, is further introduced. The adaptive structure in ASGNN allows more fine-grained handling of noisy graph structures and advances the model's robustness against adversarial attacks.
- Extensive experiments under various adversarial attack scenarios showcase the superiority of the proposed ASGNN. The numerical results corroborate that the adaptive structure property inherited in ASGNN can truly help mitigate the impact of abnormal edges.

2 Preliminaries and Background

An unweighted graph with self-loops is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V}$ and $\mathcal{E}$ being the node set and the edge set, respectively. The graph adjacency matrix is given by $\mathbf{A} \in \mathbb{R}^{N\times N}$. We denote by $\mathbf{1}$ and $\mathbf{I}$ the all-one column vector and the identity matrix, respectively. Given $\mathbf{D} = \mathrm{Diag}\left(\mathbf{A1}\right) \in \mathbb{R}^{N\times N}$ as the diagonal degree matrix, the Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. We denote by $\mathbf{A}_{\mathrm{rw}} = \mathbf{D}^{-1}\mathbf{A}$ the random walk (or row-wise) normalized adjacency matrix and by $\mathbf{A}_{\mathrm{sym}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ the symmetric normalized adjacency matrix. Subsequently, the random walk normalized and symmetric normalized Laplacian matrices are defined as $\mathbf{L}_{\mathrm{rw}} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$ and $\mathbf{L}_{\mathrm{sym}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$. $\mathbf{X} \in \mathbb{R}^{N\times M}$ (M is assumed as the dimension of the node feature) is a node feature matrix or a graph signal, and its i-th row $\mathbf{X}_{i,:}$ represents the feature vector at the i-th node where $i = 1, \ldots, N$. $\mathbf{X}_{ij}$ (or $[\mathbf{X}]_{ij}$) denotes the (i, j)-th element of $\mathbf{X}$ where $i, j = 1, \ldots, N$. For vector $\mathbf{X}_{i,:}$, $\mathbf{X}_{i,:}^{-1}$ represents its element-wise inverse.

2.1 GNNs as Graph Signal Denoising

In the literature (Yang et al., 2021; Pan et al., 2021; Zhu et al., 2021; Zhang and Zhao, 2022), it has been realized that the message passing layers for feature learning in many GNN models could be uniformly interpreted as gradient steps for minimizing certain energy functions, which carries a meaning of graph signal denoising (Ma et al., 2021). Taking the approximate personalized propagation of neural predictions (APPNP) model (Klicpera et al., 2019) as an example, the initial node feature matrix $\mathbf{Z}$ is first pre-propcessed by a neural network $g_{\boldsymbol{\theta}}(\cdot)$ (e.g., a multilayer perceptron) with model parameter $\boldsymbol{\theta}$ producing an output $\mathbf{X} = g_{\boldsymbol{\theta}}(\mathbf{Z})$, and then $\mathbf{X}$ is fed into a K-layer message passing scheme given as follows:

$$\mathbf{H}^{(0)} = \mathbf{X}, \quad \mathbf{H}^{(k+1)} = (1-\alpha)\,\mathbf{A}_{\mathrm{sym}}\mathbf{H}^{(k)} + \alpha\mathbf{X}, \ \text{ for } k = 0, \ldots, K, \tag{1}$$

where $\mathbf{H}^{(0)}$ denotes the input feature of the message passing process, $\mathbf{H}^{(k)}$ represents the learned feature after the k-th layer, and α is the teleport probability. Therefore, the message passing of an APPNP model is fully specified by two parameters, namely, a graph structure matrix $\mathbf{A}_{\mathrm{sym}}$ and a parameter α, in which $\mathbf{A}_{\mathrm{sym}}$ assumes to be known beforehand and α is treated as a hyperparameter.

From an optimization perspective, the message passing process in (1) can be seen as executing K steps of gradient descent to solve a graph signal denoising problem below with initialization $\mathbf{H}^{(0)} = \mathbf{X}$ and step size 0.5 (Zhu et al., 2021; Zhang and Zhao, 2022):

$$\underset{\mathbf{H}\in\mathbb{R}^{N\times M}}{\mathsf{minimize}} \quad \alpha\left\|\mathbf{H}-\mathbf{X}\right\|_{\mathrm{F}}^{2} + (1-\alpha)\operatorname{Tr}\big(\mathbf{H}^{\top}\mathbf{L}_{\mathrm{sym}}\mathbf{H}\big), \tag{2}$$

where $\mathbf{X}$ and α are given and share the same meaning as in (1). In (2), the first term is a fidelity term forcing the recovered graph signal $\mathbf{H}$ to be as close as possible to a noisy graph signal $\mathbf{X}$, and the second term is the symmetric normalized Laplacian smoothing term measuring the variation of the graph signal $\mathbf{H}$, which can be explicitly expressed as

$$\operatorname{Tr}\left(\mathbf{H}^{\top}\mathbf{L}_{\mathrm{sym}}\mathbf{H}\right) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{A}_{ij}\left\|\frac{\mathbf{H}_{i,:}}{\sqrt{\mathbf{D}_{ii}}} - \frac{\mathbf{H}_{j,:}}{\sqrt{\mathbf{D}_{jj}}}\right\|_{2}^{2}. \tag{3}$$

For more technical discussions on relationships between GNNs with iterative optimization algorithms for solving graph signal denoising problems, please refer to Ma et al. (2021); Zhang and Zhao (2022). Apart from using the lens of optimization to interpret existing GNN models, there are also literature (Liu et al., 2021b; Chen et al., 2021; Fu et al., 2022) working on building new GNN architectures based on designing novel optimization problems and the corresponding iterative algorithms (more discussions are provided in Appendix A).

2.2 Graph Learning with Structural Regularizers

Structural regularizers are commonly adopted to promote certain desirable properties when learning a graph (Kalofolias, 2016; Pu et al., 2021). In the following, we discuss several widely used graph structural regularizers which will be incorporated into the design of ASMP. We denote the learnable graph adjacency matrix as $\mathbf{S}$ satisfying $\mathbf{S} \in \mathcal{S}$, where the constraint

$$\mathcal{S} = \left\{\mathbf{S} \in \mathbb{R}^{N\times N} \mid 0 \le \mathbf{S}_{ij} \le 1, \ \text{for } i, j = 1, \ldots, N\right\}$$

defines the class of adjacency matrices. Under the assumption that node feature changes smoothly between adjacent nodes (Ortega et al., 2018), the Laplacian smoothing regularization term is commonly considered in graph structure learning. (3) is the symmetric normalized Laplacian smoothing term, and a random walk normalized alternative can be similarly defined by replacing $\mathbf{L}_{\mathrm{sym}}$ in (3) by $\mathbf{L}_{\mathrm{rw}}$.

Real-world graphs are normally sparsely connected, which can be represented by sparse adjacency matrices. Moreover, it is also observed that singular values of these adjacency matrices are commonly small (Zhou et al., 2013; Kumar et al., 2020). However, a noisy adjacency matrix (e.g., one perturbed by adversarial attacks) tends to be dense and to gain singular values in larger magnitudes (Jin et al., 2020). In view of this, graph structural regularizers for promoting sparsity and/or suppressing the singular values are widely adopted as priors in the literature of graph learning (Kalofolias, 2016; Egilmez et al., 2017; Dong et al., 2019). Specifically, the ℓ_1-norm of the adjacency matrix is often used to promote sparsity, defined as $\|\mathbf{S}\|_1 = \sum_{i,j=1}^{N} |\mathbf{S}_{ij}|$. For penalizing the singular values, the ℓ_1-norm and the ℓ_2-norm on the singular value vector of the adjacency matrix $\mathbf{S}$ can help. Equivalently, they can be translated to be the nuclear norm and the Frobenius norm on $\mathbf{S}$, which are given by $\|\mathbf{S}\|_* = \sum_{i=1}^{N} \sigma_i(\mathbf{S})$ and $\|\mathbf{S}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{N} \sigma_i^2(\mathbf{S})}$, where $\sigma_1(\mathbf{S}) \geq \cdots \geq \sigma_N(\mathbf{S})$ denote the ordered singular values of $\mathbf{S}$. These two regularizers both restrict the scale of the singular values while the nuclear norm also promotes low-rankness. A recent study Deng et al. (2022) points out that graph learning methods with low-rank promoting regularizers may lose a wide range of the clean graph spectrum corresponding to the important structure in the spatial domain. Thus, these low-rank based methods may impair the quality of the reconstructed graph and therefore limit the performance of GNNs. Besides, the nuclear norm is not amicable for back-propagation and incurs high computational complexity (Luo et al., 2021). Arguably, the Frobenius norm of $\mathbf{S}$ is a more friendly regularizer for graph structure learning in comparison with the nuclear norm.

3 The Proposed Graph Neural Networks

In this section, we first motivate the design principle based on jointly node feature learning and graph structure learning. Then, we develop an efficient optimization algorithm for solving this optimization problem, which eventually leads to a novel message passing scheme with adaptive structure (ASMP). After that, we provide interpretations, convergence guarantees, and specifications of ASMP. Finally, integrating ASMP with deep neural networks ends up with a new family of GNNs with adaptive structure, named ASGNNs.

3.1 A Novel Design Principle with Adaptive Graph Structure

As discussed in Section 2.1, the message passing procedure in many popular GNNs can be viewed as performing graph signal denoising (or node feature learning) (Zhu et al., 2021; Ma et al., 2021; Pan et al., 2021; Zhang and Zhao, 2022) over a prefixed graph. Unfortunately, if some edges in the graph are task-irrelevant or even maliciously manipulated, the node features learned may not be appropriate for the downstream task. Motivated by this, we propose a new design principle for message passing, that is, to learn the node features and the graph structure simultaneously, which can enable learning an adaptive graph structure from the feature for the message passing procedures. Hence, such a message passing scheme can potentially improve robustness against noisy input graph structures.

Specifically, we first construct an optimization objective by augmenting the graph signal denoising objective in (2) (we have used a random walk normalized graph Laplacian smoothing term) with a structural fidelity term $\|\mathbf{S}-\mathbf{A}\|_{\mathrm{F}}^2$, where $\mathbf{A}$ is the given initial graph adjacency matrix, and structural regularizers $\|\mathbf{S}\|_1$ and $\|\mathbf{S}\|_{\mathrm{F}}^2$. Then we obtain the following optimization problem:

$$\underset{\mathbf{H}\in\mathbb{R}^{N\times M},\, \mathbf{S}\in\mathcal{S}}{\text{minimize}} \quad \overbrace{\|\mathbf{H}-\mathbf{X}\|_{\mathrm{F}}^2 + \underbrace{\lambda \mathrm{Tr}\left(\mathbf{H}^\top \mathbf{L}_{\mathrm{rw}} \mathbf{H}\right)}^{\text{feature learning}} + \gamma\|\mathbf{S}-\mathbf{A}\|_{\mathrm{F}}^2 + \mu_1\|\mathbf{S}\|_1 + \mu_2\|\mathbf{S}\|_{\mathrm{F}}^2}_{\text{structure learning}}, \tag{4}$$

where $\mathbf{H}$ is the feature variable, $\mathbf{S}$ is the structure variable, and γ, λ, μ_1, and μ_2 are parameters balancing different terms. To enable the interplay between feature learning and structure learning, the Laplacian smoothing term is concerned with $\mathbf{S}$ rather than $\mathbf{A}$, i.e., $\mathbf{L}_{\mathrm{rw}} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{S}$ with $\mathbf{D} = \mathrm{Diag}(\mathbf{S}\mathbf{1})$. We assume that $\mathbf{D}$ does not have zero diagonal elements and $\min_i [\mathbf{D}]_{ii} = c > 0$. When adversarial attacks exist, a perturbed adjacency matrix $\mathbf{A}$ will be generated. Since attacks are generally manipulated to be unnoticeable (Jin et al., 2021), the perturbed graph adjacency matrix is largely similar to the original graph matrix in value. In view of this, we also include a structural fidelity term $\|\mathbf{S}-\mathbf{A}\|_{\mathrm{F}}^2$. The motivation for introducing the last two regularizers is elaborated in Section 2.2.

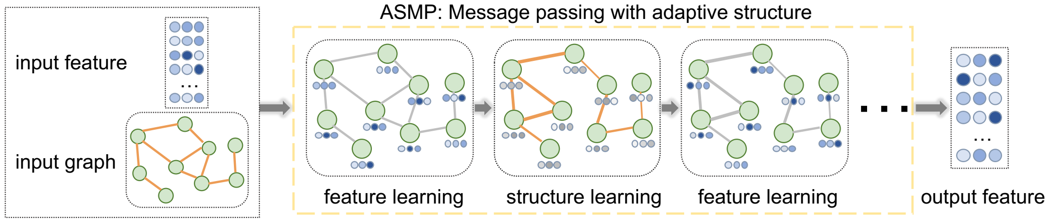

Figure 1: Illustration of ASMP. (ASMP takes the feature matrix $\mathbf{X}$ and the graph structure $\mathbf{A}$ as inputs, where the different colors of the features indicate different feature values. The ASMP updates the feature matrix through message passing and the graph structure in an alternating way, and the width of edges indicate edge weights.)

3.2 ASMP: Message Passing with Adaptive Structure

Following the idea that the message passing of a GNN model can be derived based on the optimization of a certain energy function (Ma et al., 2021; Zhang and Zhao, 2022), we can obtain a message passing scheme from (4). Different from the traditional GNN model with only the feature variable, problem (4) is nonconvex and much more challenging. For efficient problem resolution, we propose to use the alternating (proximal) gradient descent method (Parikh and Boyd, 2014) to solve it, i.e., alternatingly optimizing one variable at a time with the other variable fixed. (Note that a joint optimization approach is eligible, however, it would lead to slower convergence than the alternating optimization approach, more details can be found in Appendix D.

We denote by $\mathbf{H}^{(k)}$ and $\mathbf{S}^{(k)}$ the variables at the k-th iteration ($k = 0, \ldots, K$). In the following, the update rules for $\mathbf{H}$ and $\mathbf{S}$ will be discussed, respectively.

Updating node feature matrix $\mathbf{H}$: Given $\left\{\mathbf{H}^{(k)}, \mathbf{S}^{(k)}\right\}$, the subproblem with respect to feature matrix $\mathbf{H}$ is given by

$$\underset{\mathbf{H}\in\mathbb{R}^{N\times M}}{\mathsf{minimize}} \quad \|\mathbf{H}-\mathbf{X}\|_{\mathrm{F}}^2 + \lambda \mathrm{Tr}\left(\mathbf{H}^\top \mathbf{L}_{\mathrm{rw}}^{(k)} \mathbf{H}\right), \tag{5}$$

where $\mathbf{L}_{\mathrm{rw}}^{(k)} = \mathbf{I} - \mathrm{Diag}\left(\mathbf{S}^{(k)}\mathbf{1}\right)^{-1}\mathbf{S}^{(k)}$. One gradient step for $\mathbf{H}$ is computed as

$$\begin{aligned}
\mathbf{H}^{(k+1)} &= \mathbf{H}^{(k)} - \eta_1\left(2\mathbf{H}^{(k)} - 2\mathbf{X} + 2\lambda\mathbf{L}_{\mathrm{rw}}^{(k)}\mathbf{H}^{(k)}\right)\\
&= \mathbf{H}^{(k)} - \eta_1\left(2\mathbf{H}^{(k)} - 2\mathbf{X} + 2\lambda\left(\mathbf{I} - \mathrm{Diag}\left(\mathbf{S}^{(k)}\mathbf{1}\right)^{-1}\mathbf{S}^{(k)}\right)\mathbf{H}^{(k)}\right)\\
&= \left(1 - 2\eta_1 - 2\eta_1\lambda\right)\mathbf{H}^{(k)} + 2\eta_1\lambda\mathrm{Diag}\left(\mathbf{S}^{(k)}\mathbf{1}\right)^{-1}\mathbf{S}^{(k)}\mathbf{H}^{(k)} + 2\eta_1\mathbf{X},
\end{aligned}$$

where η_1 denotes the step size.

Updating graph structure matrix $\mathbf{S}$: Given $\left\{\mathbf{H}^{(k+1)}, \mathbf{S}^{(k)}\right\}$ and $\mathrm{Tr}\left(\mathbf{H}^{(k+1)\top}\mathbf{L}_{\mathrm{rw}}\mathbf{H}^{(k+1)}\right) = \mathrm{Tr}\left(\mathbf{H}^{(k+1)\top}\mathbf{H}^{(k+1)}\right) - \mathrm{Tr}\left(\mathbf{H}^{(k+1)\top}\mathrm{Diag}\left(\mathbf{S}\mathbf{1}\right)^{-1}\mathbf{S}\mathbf{H}^{(k+1)}\right)$, the subproblem for $\mathbf{S}$ becomes

$$\underset{\mathbf{S}\in\mathcal{S}}{\mathsf{minimize}} \quad \gamma\|\mathbf{S}-\mathbf{A}\|_{\mathrm{F}}^2 - \lambda\mathrm{Tr}\left(\mathbf{H}^{(k+1)\top}\mathrm{Diag}\left(\mathbf{S}\mathbf{1}\right)^{-1}\mathbf{S}\mathbf{H}^{(k+1)}\right) + \mu_1\left\|\mathbf{S}\right\|_1 + \mu_2\left\|\mathbf{S}\right\|_{\mathrm{F}}^2. \tag{6}$$

Due to the non-smoothness of the objective function, we apply one step of the proximal gradient descent method (Parikh and Boyd, 2014) for this subproblem. Define

$$\begin{aligned}
\mathbf{T}^{(k)} = &\left(2\gamma + 2\mu_2\right)\mathbf{S}^{(k)} - 2\gamma\mathbf{A} - \lambda\mathrm{Diag}\left(\mathbf{S}^{(k)}\mathbf{1}\right)^{-1}\mathbf{H}^{(k+1)}\left(\mathbf{H}^{(k+1)}\right)^\top\\
&+ \lambda\mathrm{Diag}\left(\mathrm{Diag}\left(\mathbf{S}^{(k)}\mathbf{1}\right)^{-1}\mathbf{S}^{(k)}\mathbf{H}^{(k+1)}\left(\mathbf{H}^{(k+1)}\right)^\top\mathrm{Diag}\left(\mathbf{S}^{(k)}\mathbf{1}\right)^{-1}\right)\mathbf{1}^\top.
\end{aligned}$$

One step of proximal gradient descent is given as follows (details are given in Appendix B):

$$\mathbf{S}^{(k+1)} = \mathrm{prox}_{\eta_2\left(\mu_1\|\cdot\|_1 + \mathbb{I}_{\mathcal{S}}(\cdot)\right)}\left(\mathbf{S}^{(k)} - \eta_2\mathbf{T}^{(k)}\right), \tag{7}$$

where η_2 is the step size and $\mathbb{I}_{\mathcal{S}}\left(\mathbf{S}\right)$ denotes the indicator function taking value 0 if $\mathbf{S}\in\mathcal{S}$ and $+\infty$ otherwise. Moreover, the proximal operator (7) can be computed analytically as follows:

$$\mathbf{S}^{(k+1)} = \min\left\{1, \mathrm{ReLU}\left(\mathbf{S}^{(k)} - \eta_2\mathbf{T}^{(k)} - \eta_2\mu_1\mathbf{1}\mathbf{1}^\top\right)\right\}.$$

In conclusion, the overall procedure of ASMP can be summarized as follows:

$$\begin{cases} \mathbf{H}^{(k+1)} = (1-2\eta_1-2\eta_1\lambda)\,\mathbf{H}^{(k)} + 2\eta_1\lambda\mathrm{Diag}\big(\mathbf{S}^{(k)}\mathbf{1}\big)^{-1}\mathbf{S}^{(k)}\mathbf{H}^{(k)} + 2\eta_1\mathbf{X} \\ \mathbf{S}^{(k+1)} = \min\left\{1, \mathrm{ReLU}\left(\mathbf{S}^{(k)} - \eta_2\mathbf{T}^{(k)} - \eta_2\mu_1\mathbf{1}\mathbf{1}^\top\right)\right\}, \end{cases} \quad k = 1,\ldots,K. \tag{ASMP}$$

The ASMP can be interpreted as the standard message passing (i.e., the update step of $\mathbf{H}$) with extra operations that adaptively adjust the graph structure (i.e., the update step of $\mathbf{S}$). Therefore, an edge of the graph included in some layers may be excluded or down-weighted in other layers. A pictorial illustration of the ASMP procedure is provided in Figure 1. The K-layer ASMP can be fully specified by $\mathbf{X}$, $\mathbf{A}$, γ, λ, μ_1, μ_2, η_1, and η_2, which we generally denote as $\mathrm{ASMP}_K\big(\mathbf{X},\mathbf{A},\gamma,\lambda,\mu_1,\mu_2,\eta_1,\eta_2\big)$.

Note that ASMP is general enough to cover several existing propagation rules as the special cases.

Remark 1 (Special cases). If we use a fixed graph structure $\mathbf{S}^{(0)} = \cdots = \mathbf{S}^{(K)} = \mathbf{A}$ in ASMP, i.e., $\mu_1 = \mu_2 = \gamma = 0$, the ASMP reduces to a standard message passing procedure that only performs feature learning. Specifically, with $\eta_1 = \frac{1}{2+2\lambda}$ and adopting the symmetric normalized adjacency matrix, ASMP can be written as

$$\mathbf{H}^{(k+1)} = \frac{\lambda}{1+\lambda}\mathbf{A}_{\mathrm{sym}}\mathbf{H}^{(k)} + \frac{1}{1+\lambda}\mathbf{X}. \tag{8}$$

Case I: when $\lambda = \frac{1}{\alpha} - 1$, the operation in (8) becomes the message passing rule of APPNP (Klicpera et al., 2019):

$$\mathbf{H}^{(k+1)} = (1-\alpha)\,\mathbf{A}_{\mathrm{sym}}\mathbf{H}^{(k)} + \alpha\mathbf{X}.$$

Case II: when $\lambda = \infty$, the operation in (8) becomes the simple aggregation in many GNN models such as the GCN model (Kipf and Welling, 2017) and the simple graph convolution (SGC) model (Wu et al., 2019a):

$$\mathbf{H}^{(k+1)} = \mathbf{A}_{\mathrm{sym}}\mathbf{H}^{(k)}.$$

Instead of updating both $\mathbf{S}$ and $\mathbf{H}$ once, we can also choose to update them in several steps. The convergence of ASMP can be guaranteed with proper selections of the step sizes as demonstrated below in Theorem 1.

Theorem 1. *The objective function in the H-block problem* (5) *and the smooth part of the objective in the S-block problem* (6) *are provable L-smooth with Lipschitz constants*

$$L_H = 2 + 4\lambda \quad \textit{and} \quad L_S = 2\gamma + 2\mu_2 + \frac{2\lambda}{c^2}N^2B^2 + \frac{2\lambda}{c^3}N^3B^2\sqrt{N},$$

respectively. Let $\mathbf{H}^{(0)} = \mathbf{X}$ and $\mathbf{S}^{(0)} = \mathbf{A}$, and denote $\left\{\mathbf{H}^{(k)}, \mathbf{S}^{(k)}\right\}_{k=1}^{K}$ as the sequence generated by ASMP. Under the assumption that feature vectors of all nodes are uniformly bounded by a constant, i.e., $\left\|\mathbf{H}_{i,:}^{(k)}\right\|_2 \leq B$ for $i = 1,\ldots,N$ and $k = 0,\ldots,K$, the sequence generated by (ASMP) *converge to a first-order stationary point of Problem* (4) *with step sizes satisfying $\eta_1 < \frac{2}{L_H}$ and $\eta_2 < \frac{2}{L_S}$.*

Proof. The proof for Theorem 1 is in Appendix C. Note that if multiple updating steps are used for $\mathbf{S}$ and $\mathbf{H}$ in ASMP, the convergence result still hold (Bolte et al., 2014; Nikolova and Tan, 2017). □

There have been some existing graph structure learning methods designed based on an optimization problem (Jin et al., 2020; Luo et al., 2021; Zhu et al., 2022). However, our design principle is conceptually different from theirs. Specifically, these prior works augment structural regularizers to the loss function, while ours amounts to the modification of the GNN architecture itself. As a result, these prior works need alternating training and the model performance of these methods are highly affected by the augmented structural regularizers in the training loss. Besides, the Laplacian smoothing term adopted in (4) promotes the smoothness of the denoised signal on the optimized graph, which is more natural than the one used in previous works that promote the smoothness of the input signal on the optimized graph.

3.3 ASGNN: Graph Neural Networks with Adaptive Structure

In this section, we introduce a family of GNNs leveraging the ASMP scheme. Integrating (ASMP) with neural network $g_{\boldsymbol{\theta}}(\cdot)$, i.e., let $\mathbf{X} = g_{\boldsymbol{\theta}}(\mathbf{Z})$, a K-layer ASGNN model is defined as follows:

$$\mathbf{H}^{(K+1)} = \mathrm{ASMP}_K\Big(g_{\boldsymbol{\theta}}(\mathbf{Z}), \mathbf{A}, \gamma, \lambda, \mu_1, \mu_2, \eta_1, \eta_2\Big).$$

We have chosen the decoupled architecture similar to APPNP (Klicpera et al., 2019) and deep adaptive GNN (DAGNN) (Liu et al., 2021a). Specially, the model $g_{\boldsymbol{\theta}}$ will first transform the initial node features as $\mathbf{X} = g_{\boldsymbol{\theta}}\big(\mathbf{Z}\big)$. Then ASMP takes $g_{\boldsymbol{\theta}}\big(\mathbf{Z}\big)$ as input, and performs K steps of message passing.

The coefficients in ASGNN, i.e., γ, λ, μ_1, and μ_2, are set to be learnable parameters. For example, in semi-supervised node classification tasks, the loss function is chosen as the cross-entropy classification loss on the labeled nodes and the whole model can be trained in an end-to-end way. Since ASMP is derived from the alternating (proximal) gradient descent algorithm, a trained ASMP is naturally a parameter-optimized iterative algorithm. The step sizes η_1 and η_2 in ASMP can be chosen according to the results in Theorem 1. However, such choices seem to be too conservative in practice and may lead to slow convergence. Thus, we also consider the step sizes η_1 and η_2 as learnable parameters. The convergence property of ASMP with the learned step sizes will be showcased in the experiments. In conclusion, there are six parameters in ASMP considered during the learning process.

4 Experiments

In this section, we conduct experiments to validate the effectiveness of the proposed ASGNN model. First, we introduce the experimental settings. Then, we assess the performance of ASGNN on semi-supervised node classifications tasks and investigate the benefits of introducing adaptive structure into GNNs against global attacks and targeted attacks. Finally, we analyze the structure denoising ability and the convergence property of ASMP with the learned step sizes.

4.1 Experiment Settings

Datasets: We perform numerical experiments on 4 real-world citation graphs, i.e., Cora, Citeseer (Sen et al., 2008), Cora-ML (Bojchevski and Günnemann, 2018), and ACM (Wang et al., 2019), and only consider the largest connected component in each dataset.

Baselines: To evaluate the effectiveness of ASGNN, we compare it with GCN and several benchmarks that are designed from different perspectives to robustify the GNNs, including GCN-Jaccard (Wu et al., 2019b) that pre-processes the graph by eliminating edges with low Jaccard similarity of node features, GCN-SVD (Entezari et al., 2020) that applies the low-rank approximation of the given graph adjacency matrix, Pro-GNN (Jin et al., 2020) that jointly learns a graph structure and a GNN model guided by some predefined structural priors, and Elastic GNN (Liu et al., 2021b) that utilizes trend filtering instead of Laplacian smoothing to promote robustness. The code is implemented based on PyTorch Geometric (Fey and Lenssen, 2019). For GCN-Jaccard, GCN-SVD, and Pro-GNN, we use the implementation provided in DeepRobust (Li et al., 2020). For Elastic GNN, we follow the implementation provided in the original paper (Liu et al., 2021b).

Parameter settings: For all the experiment results, we give the average performance and standard variance with 10 independent trials. For each graph, we randomly select 10%/10%/80% of nodes for training, validation, and testing. The Adam optimizer is used in all experiments. The models' hyperparameters are tuned based on the results of the validation set. The search space of hyperparameters are as follows: 1) learning rate: {0.005, 0.01, 0.05}; 2) weight decay: {0, 5e-5, 5e-4}; 3) dropout rate: {0.1, 0.5, 0.8}; 4) model depth: {2, 4, 8, 16}. For GCN-Jaccard, the threshold of Jaccard similarity for removing dissimilar edges is chosen from {0.01, 0.02, 0.03, 0.04, 0.05, 0.1}. For GCN-SVD, the reduced rank of the graph is tuned from {5, 10, 15, 50, 100, 200}. For Elastic GNN, the regularization coefficients are chosen from {3, 6, 9}. For Pro-GNN, we adopt the hyperparameters provided in their paper (Jin et al., 2020).

Table 1: Node classification performance (accuracy ± std) under global attack (**Bold**: the best model; wavy: the runner-up model)

Dataset	Ptb. rate (%)	GCN	GCN-Jaccard	GCN-SVD	Pro-GNN	Elastic GNN	ASGNN
Cora	0	85.34 ± 0.39	81.75 ± 0.49	75.15 ± 0.64	82.94 ± 0.28	84.80 ± 0.58	**85.38 ± 0.24**
	5	79.71 ± 0.48	77.81 ± 0.52	73.71 ± 0.42	82.20 ± 0.35	82.26 ± 0.69	**82.31 ± 0.53**
	10	74.28 ± 0.79	74.38 ± 0.30	65.85 ± 0.39	79.30 ± 0.64	79.47 ± 1.52	**80.31 ± 0.61**
	15	69.05 ± 0.77	72.54 ± 0.31	65.33 ± 0.47	77.69 ± 0.74	77.84 ± 1.08	**78.11 ± 0.76**
	20	57.76 ± 1.01	71.76 ± 0.48	60.85 ± 0.74	74.16 ± 1.02	63.68 ± 0.27	**77.04 ± 0.59**
	25	52.67 ± 1.00	69.67 ± 0.46	59.31 ± 0.47	71.19 ± 1.27	62.90 ± 3.37	**75.18 ± 0.97**
Citeseer	0	73.97 ± 0.54	72.09 ± 0.49	68.34 ± 0.39	73.35 ± 0.47	73.82 ± 0.43	**73.99 ± 0.93**
	5	72.57 ± 0.93	70.79 ± 0.30	67.59 ± 0.43	73.16 ± 0.42	73.30 ± 0.37	**73.35 ± 0.41**
	10	71.21 ± 1.44	70.27 ± 0.62	67.38 ± 0.65	72.78 ± 0.79	72.78 ± 0.66	**72.83 ± 0.56**
	15	68.00 ± 1.04	69.97 ± 1.49	66.47 ± 0.51	71.55 ± 0.73	71.73 ± 1.03	**71.85 ± 1.83**
	20	59.75 ± 0.83	69.49 ± 0.71	65.83 ± 0.69	70.07 ± 1.12	61.55 ± 1.82	**71.06 ± 3.09**
	25	59.98 ± 0.98	68.14 ± 0.36	62.34 ± 0.61	69.73 ± 0.93	63.98 ± 2.17	**70.03 ± 3.45**
Cora-ML	0	86.59 ± 0.07	84.68 ± 0.32	82.96 ± 0.27	79.48 ± 0.40	**87.01 ± 0.28**	86.68 ± 0.43
	5	80.99 ± 0.50	81.80 ± 0.37	81.78 ± 0.46	78.57 ± 0.16	84.68 ± 0.25	**84.80 ± 0.80**
	10	74.57 ± 0.75	80.35 ± 0.24	81.75 ± 0.33	78.74 ± 0.84	82.01 ± 0.64	**83.09 ± 0.59**
	15	54.69 ± 0.52	**76.53 ± 0.29**	74.76 ± 0.44	73.62 ± 0.85	64.59 ± 2.69	73.71 ± 1.82
	20	40.24 ± 1.97	**76.46 ± 0.58**	53.94 ± 0.45	72.72 ± 0.88	52.18 ± 0.71	73.65 ± 1.42
	25	44.13 ± 3.42	**75.95 ± 0.50**	71.98 ± 0.17	74.91 ± 0.56	53.05 ± 0.36	75.36 ± 1.34
ACM	0	91.75 ± 0.10	89.62 ± 0.41	87.51 ± 0.42	90.11 ± 0.57	91.45 ± 0.21	**92.56 ± 0.42**
	5	84.29 ± 0.57	84.64 ± 0.27	85.29 ± 1.13	88.25 ± 1.19	90.10 ± 0.27	**90.60 ± 0.28**
	10	81.71 ± 0.61	81.12 ± 0.31	84.59 ± 0.68	88.14 ± 0.60	89.45 ± 0.41	**90.10 ± 0.35**
	15	79.65 ± 1.00	74.66 ± 0.94	83.81 ± 0.81	87.59 ± 0.74	89.23 ± 0.34	**89.93 ± 0.51**
	20	79.95 ± 0.50	74.26 ± 0.75	82.35 ± 1.64	87.83 ± 1.03	88.65 ± 0.35	**90.61 ± 0.28**
	25	79.55 ± 1.16	74.12 ± 0.81	82.04 ± 0.99	88.06 ± 0.85	88.15 ± 0.58	**90.15 ± 0.33**

4.2 Performance Under Adversarial Attack

The performance of the compared models is evaluated under the training-time adversarial attacks (Wang and Gong, 2019; Zügner and Günnemann, 2019), i.e., the graph is first attacked, and then the GNN models are trained on the perturbed graph. In the following, we conduct experiments under both the global attack and the targeted attack. Specifically, the global attack aims to reduce the overall performance of GNNs (Zügner and Günnemann, 2019) while the targeted attack aims to fool GNNs on some specific nodes (Zügner et al., 2018).

4.2.1 Global Attack

We first test the node classification performance of ASGNN and other baselines under global attack using a representative global attack method called meta-attack (Zügner and Günnemann, 2019). We vary the perturbation rate, i.e., the ratio of changed edges, from 0% to 25% with an increasing step of 5%. The results are reported in Table 1. From the table, we observe that the proposed ASGNN model outperforms other methods in most cases. For instance, ASGNN improves GCN over 30% on the Cora-ML dataset at a 20% perturbation rate, whereas ASGNN improves GCN over 20% on the Cora dataset at a 25% perturbation rate. On Cora, Citeseer, and ACM datasets, ASGNN beats other baselines at various perturbation rates by a large margin. The GCN-Jaccard method slightly outperforms ASGNN on the Cora-ML dataset at a 15%-25% perturbation rate, while it performs poorly on other datasets. Specifically, on the other three datasets under the 25% perturbation rate, ASGNN outperforms GCN-Jaccard by 22%, 10%, and 10%, respectively. Such inspiring results demonstrate that ASGNN can better resist global attack than other baseline methods.

4.2.2 Targeted Attack

For the targeted attack, we use a representative method called NETTACK (Zügner et al., 2018). Following existing works (Zhu et al., 2019; Jin et al., 2020), we vary the perturbation number made on every node, i.e., the number of edge removals/additions, from 0 to 5 with an increasing step of 1. The results are reported in Table 2. We choose the nodes in the test set with degrees larger than 10 as targeted nodes and the reported classification performance is evaluated on target nodes. Thus, the results in Table 2 is not directly comparable with the results in Table 1. From the table, we can see that the proposed ASGNN attains better performance than other baselines in most cases. For instance, on the Citeseer dataset with 5 perturbations per targeted node, ASGNN improves GCN by 25% and

Table 2: Node classification performance (accuracy ± std) under targeted attack (**Bold**: the best model; wavy: the runner-up model)

Dataset	Ptb. number	GCN	GCN-Jaccard	GCN-SVD	Pro-GNN	Elastic GNN	ASGNN
Cora	0	82.53 ± 1.45	81.95 ± 0.29	77.35 ± 1.40	82.92 ± 0.29	**84.93 ± 2.28**	83.01 ± 1.57
	1	78.19 ± 1.66	75.30 ± 1.54	75.18 ± 1.80	81.48 ± 0.91	81.44 ± 1.81	**81.57 ± 1.18**
	2	71.33 ± 1.29	70.24 ± 1.52	71.81 ± 1.63	**79.03 ± 1.80**	76.74 ± 1.97	78.80 ± 1.03
	3	66.63 ± 1.53	69.04 ± 0.94	65.18 ± 1.65	72.75 ± 1.32	73.97 ± 2.67	**75.30 ± 1.35**
	4	61.45 ± 2.16	61.68 ± 1.05	58.79 ± 2.14	70.11 ± 2.45	68.31 ± 3.50	**70.24 ± 4.70**
	5	56.75 ± 1.37	59.52 ± 1.88	59.16 ± 2.71	66.98 ± 1.63	65.78 ± 2.51	**68.55 ± 3.21**
Citeseer	0	81.27 ± 0.95	80.31 ± 1.26	80.47 ± 1.01	81.24 ± 1.01	81.42 ± 0.76	**81.90 ± 1.95**
	1	80.63 ± 0.63	80.00 ± 1.45	78.57 ± 2.67	80.52 ± 0.85	80.79 ± 1.17	**81.21 ± 1.11**
	2	79.84 ± 1.02	76.98 ± 1.77	73.02 ± 6.77	80.63 ± 0.95	81.01 ± 0.50	**81.11 ± 1.32**
	3	66.51 ± 3.36	74.76 ± 1.31	76.03 ± 3.71	79.36 ± 4.76	80.31 ± 1.10	**80.32 ± 1.90**
	4	62.54 ± 1.62	76.34 ± 1.49	62.22 ± 3.31	75.71 ± 4.87	72.06 ± 5.60	**80.16 ± 1.28**
	5	52.70 ± 1.98	72.85 ± 1.65	60.16 ± 6.67	73.95 ± 7.13	73.96 ± 3.90	**77.94 ± 7.08**
Cora-ML	0	88.33 ± 0.56	84.91 ± 0.24	83.06 ± 0.30	86.64 ± 0.80	88.84 ± 0.67	**88.88 ± 0.31**
	1	83.85 ± 0.47	83.82 ± 0.33	80.51 ± 0.28	83.77 ± 0.96	85.87 ± 1.01	**86.20 ± 1.21**
	2	79.18 ± 1.31	83.75 ± 0.33	78.73 ± 0.34	82.29 ± 0.13	84.34 ± 1.02	**84.41 ± 1.51**
	3	76.19 ± 0.82	82.18 ± 0.48	78.23 ± 0.20	80.58 ± 1.06	81.41 ± 1.62	**83.50 ± 1.08**
	4	70.61 ± 1.56	**81.90 ± 0.45**	76.25 ± 0.51	79.92 ± 1.44	78.31 ± 1.20	80.81 ± 0.99
	5	64.52 ± 1.29	**81.35 ± 0.34**	75.47 ± 0.41	77.63 ± 1.68	74.08 ± 1.89	80.17 ± 3.85
ACM	0	90.33 ± 0.09	89.76 ± 0.39	86.21 ± 0.46	90.22 ± 0.60	90.31 ± 0.71	**92.11 ± 0.41**
	1	89.70 ± 0.22	83.62 ± 0.47	83.50 ± 0.71	87.09 ± 0.65	90.03 ± 0.43	**91.08 ± 1.36**
	2	82.06 ± 1.12	80.47 ± 0.63	82.09 ± 0.73	87.01 ± 0.53	87.72 ± 1.27	**90.55 ± 0.47**
	3	80.26 ± 1.17	77.07 ± 0.67	81.09 ± 0.78	87.07 ± 1.14	84.75 ± 2.07	**89.54 ± 0.49**
	4	76.86 ± 1.46	77.45 ± 0.44	80.76 ± 0.54	87.04 ± 1.16	83.49 ± 2.01	**89.45 ± 0.51**
	5	73.32 ± 1.77	74.38 ± 0.59	79.74 ± 0.77	86.42 ± 0.71	81.67 ± 1.53	**88.44 ± 0.71**

Table 3: Testing loss on the clean graph of GNN models trained on perturbed graphs with various perturbation numbers under targeted attack.

Ptb. number	0	1	2	3	4	5
GCN	1.0315	1.3923	1.6703	2.0372	2.6108	3.3270
ASGNN	0.5534	0.7577	0.8323	0.8534	0.9040	1.0273

outperforms other baselines by around 4%. The reported results demonstrate that ASGNN can also effectively resist the targeted attack.

4.2.3 Robustness of ASGNN

The message passing scheme in ASGNN is designed from a principle of simultaneous graph signal and graph structure denoising. To validate that ASGNN can help purify (i.e., denoise) the structure, we train ASGNN models on perturbed graphs with various perturbation numebrs under NETTACK and evaluate their testing loss with the clean graph. From the results in Table 3, we observe that ASGNN achieves lower losses than GCN in all cases. Besides, the loss increases slower as the perturbation number grows in ASGNN than in GCN. These results indicate that ASGNN can help denoise the perturbed structure.

As proved in Theorem 1, the convergence of ASMP is guaranteed with proper choices of step sizes. Since we choose to set the two step sizes in ASMP as learnable parameters, we provide an additional experiment to evaluate the convergence of ASMP with learned step sizes empirically in Appendix F.

5 Conclusion

In this work, we have developed an interpretable robust message passing scheme with adaptive structure following the simultaneous graph signal and graph structure denoising principle named ASMP. Integrating ASMP with neural network components, we have obtained a family of robust graph neural networks with adaptive structure. Extensive experiments on real-world datasets with various adversarial attack settings corroborate the effectiveness and the robustness of the proposed graph neural network architecture.

References

Hongjoon Ahn, Youngyi Yang, Quan Gan, David Wipf, and Taesup Moon. Descent steps of a relation-aware energy produce heterogeneous graph neural networks. *arXiv preprint arXiv:2206.11081*, 2022.

Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *Proceedings of the International Conference on Learning Representations*, 2018.

Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1):459–494, 2014.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Siheng Chen, Yonina C Eldar, and Lingxiao Zhao. Graph unrolling networks: Interpretable neural networks for graph signal denoising. *IEEE Transactions on Signal Processing*, 69:3699–3713, 2021.

Yu Chen, Lingfei Wu, and Mohammed Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Proceedings of Advances in Neural Information Processing Systems*, 33:19314–19326, 2020.

Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Society, 1997.

Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *Proceedings of the International Conference on Machine Learning*, pages 1115–1124. PMLR, 2018.

Chenhui Deng, Xiuyu Li, Zhuo Feng, and Zhiru Zhang. Garnet: Reduced-rank topology learning for robust and scalable graph neural networks. *arXiv preprint arXiv:2201.12741*, 2022.

Xiaowen Dong, Dorina Thanou, Pascal Frossard, and Pierre Vandergheynst. Learning laplacian matrix in smooth graph signal representations. *IEEE Transactions on Signal Processing*, 64(23): 6160–6173, 2016.

Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard. Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, 36(3):44–63, 2019.

Hilmi E Egilmez, Eduardo Pavez, and Antonio Ortega. Graph learning from data under laplacian and structural constraints. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):825–841, 2017.

Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 169–177, 2020.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 1972–1982. PMLR, 2019.

Guoji Fu, Peilin Zhao, and Yatao Bian. p-laplacian based graph neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 6878–6917. PMLR, 2022.

Simon Geisler, Daniel Zügner, and Stephan Günnemann. Reliable graph neural networks via robust aggregation. *Proceedings of Advances in Neural Information Processing Systems*, 33:13272–13284, 2020.

Stephan Günnemann. Graph neural networks: Adversarial robustness. In Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao, editors, *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 149–176. Springer Singapore, Singapore, 2022.

Zhimeng Jiang, Xiaotian Han, Chao Fan, Zirui Liu, Na Zou, Ali Mostafavi, and Xia Hu. Fmp: Toward fair graph message passing against topology bias. *arXiv preprint arXiv:2202.04187*, 2022.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 66–74, 2020.

Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. Adversarial attacks and defenses on graphs: A review, a tool and empirical studies. *ACM SIGKDD Explorations Newsletter*, 22(2):19–34, 2021.

Vassilis Kalofolias. How to learn a graph from smooth signals. In *Proceedings of Artificial Intelligence and Statistics*, pages 920–929. PMLR, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations*, 2017.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *Proceedings of the International Conference on Learning Representations*, 2019.

Sandeep Kumar, Jiaxi Ying, José Vinícius de Miranda Cardoso, and Daniel P Palomar. A unified framework for structured graph learning via spectral constraints. *Journal of Machine Learning Research*, 21(22):1–60, 2020.

Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020.

Xiaorui Liu, Jiayuan Ding, Wei Jin, Han Xu, Yao Ma, Zitao Liu, and Jiliang Tang. Graph neural networks with adaptive residual. *Proceedings of Advances in Neural Information Processing Systems*, 34, 2021a.

Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. Elastic graph neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 6837–6849. PMLR, 2021b.

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*, pages 779–787, 2021.

Yao Ma, Xiaorui Liu, Tong Zhao, Yozen Liu, Jiliang Tang, and Neil Shah. A unified view on graph neural networks as graph signal denoising. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1202–1211, 2021.

Mark Newman. *Networks*. Oxford University Press, 2018.

Mila Nikolova and Pauline Tan. Alternating proximal gradient descent for nonconvex regularised problems with multiconvex coupling terms. *HAL*, 2017.

Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106 (5):808–828, 2018.

Xuran Pan, Shiji Song, and Gao Huang. A unified framework for convolution-based graph neural networks. In *URL https://openreview. net/forum*, 2021.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3): 127–239, 2014.

Xingyue Pu, Tianyue Cao, Xiaoyun Zhang, Xiaowen Dong, and Siheng Chen. Learning to learn graph topologies. *Proceedings of Advances in Neural Information Processing Systems*, 34:4249–4262, 2021.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–93, 2008.

Harsh Shrivastava, Xinshi Chen, Binghong Chen, Guanghui Lan, Srinivas Aluru, Han Liu, and Le Song. Glad: Learning sparse graph recovery. In *Proceedings of the International Conference on Learning Representations*, 2020.

Lichao Sun, Yingtong Dou, Carl Yang, Ji Wang, Philip S Yu, Lifang He, and Bo Li. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*, 2018.

Xianfeng Tang, Yandong Li, Yiwei Sun, Huaxiu Yao, Prasenjit Mitra, and Suhang Wang. Transferring robustness for graph neural network against poisoning attacks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 600–608, 2020.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Proceedings of the International Conference on Learning Representations*, 2018.

Binghui Wang and Neil Zhenqiang Gong. Attacking graph-based classification via manipulating the graph structure. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 2023–2040, 2019.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *Proceedings of the World Wide Web Conference*, pages 2022–2032, 2019.

Yiwei Wang, Shenghua Liu, Minji Yoon, Hemank Lamba, Wei Wang, Christos Faloutsos, and Bryan Hooi. Provably robust node classification via low-pass message passing. In *Proceedings of the IEEE International Conference on Data Mining*, pages 621–630. IEEE, 2020.

Yu-Xiang Wang, James Sharpnack, Alex Smola, and Ryan Tibshirani. Trend filtering on graphs. In *Proceedings of Artificial Intelligence and Statistics*, pages 1042–1050. PMLR, 2015.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, pages 6861–6871. PMLR, 2019a.

Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: deep insights into attack and defense. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 4816–4823, 2019b.

Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Singapore, Singapore, 2022.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020.

Yongyi Yang, Tang Liu, Yangkun Wang, Jinjing Zhou, Quan Gan, Zhewei Wei, Zheng Zhang, Zengfeng Huang, and David Wipf. Graph neural networks inspired by classical iterative algorithms. In *Proceedings of the International Conference on Machine Learning*, pages 11773–11783. PMLR, 2021.

Donghan Yu, Ruohong Zhang, Zhengbao Jiang, Yuexin Wu, and Yiming Yang. Graph-revised convolutional network. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 378–393. Springer, 2020.

Hongwei Zhang, Tijin Yan, Zenjun Xie, Yuanqing Xia, and Yuan Zhang. Revisiting graph convolutional network on semi-supervised node classification from an optimization perspective. *arXiv preprint arXiv:2009.11469*, 2020.

Li Zhang and Haiping Lu. A feature-importance-aware and robust aggregator for gcn. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1813–1822, 2020.

Xiang Zhang and Marinka Zitnik. Gnnguard: Defending graph neural networks against adversarial attacks. In *Proceedings of Advances in Neural Information Processing Systems*, pages 9263–9275, 2020.

Zepeng Zhang and Ziping Zhao. Towards understanding graph neural networks: An algorithm unrolling perspective. *arXiv preprint arXiv:2206.04471*, 2022.

Lingxiao Zhao and Leman Akoglu. PairNorm: Tackling oversmoothing in GNNs. In *Proceedings of the International Conference on Learning Representations*, 2020.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

Ke Zhou, Hongyuan Zha, and Le Song. Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes. In *Proceedings of Artificial Intelligence and Statistics*, pages 641–649. PMLR, 2013.

Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1399–1407, 2019.

Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. Interpreting and unifying graph neural networks with an optimization framework. In *Proceedings of the Web Conference 2021*, pages 1215–1226, 2021.

Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. A survey on graph structure learning: Progress and opportunities. *arXiv preprint arXiv:2103.03036*, 2022.

Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *Proceedings of the International Conference on Learning Representations*, 2019.

Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856, 2018.

A RELATED WORK ON OPTIMIZATION-INDUCED GRAPH NEURAL NETWORKS

With the observation that a lot of existing GNN models can be interpreted as (unrolling) iterative algorithms for solving a graph signal denoising optimization problem (Ma et al., 2021; Zhu et al., 2021; Zhang and Zhao, 2022), there is a line of works which are proposed to strengthen the capability of GNNs by designing the underlying optimization problem and the corresponding iterative algorithms. For example, inspired by the idea of trend filtering (Wang et al., 2015), Liu et al. (2021b) replace the Laplacian smoothing term (in the form of ℓ_2 norm) with an $\ell_{2,1}$ norm to promote robustness against abnormal edges. Also for robustness pursuit, Yang et al. (2021) replace the Laplacian smoothing term with some robustness promoting nonlinear functions over pairwise node distances. Besides promoting smoothness over connected nodes, Zhang et al. (2020); Zhao and Akoglu (2020) suggest further promoting the non-smoothness over the disconnected nodes, which is achieved by deducting the sum of distances between disconnected pairs of nodes from the graph signal denoising objective. Moreover, Jiang et al. (2022) augment the graph signal denoising objective with a fairness term to fight against large topology bias. Most recently, Fu et al. (2022) propose p-Laplacian message passing and ^{p}GNN, which is capable of dealing with heterophilic graphs and is robust to noisy edges. Whereas Ahn et al. (2022) design a novel regularization term to build heterogeneous GNNs. These optimization-induced GNNs share a similar design philosophy with ours, while none of them consider improving the robustness of GNNs by using adaptive structures.

B DERIVATION OF THE PROXIMAL GRADIENT STEP (7)

Since $\mathbf{D} = \mathrm{Diag}(\mathbf{S1})$, the differential of scalar function $\mathrm{Tr}\left(\mathbf{H}^\top\mathbf{D}^{-1}\mathbf{SH}\right)$ of matrix $\mathbf{S}$ can be computed as follows:

$$\begin{aligned}
\mathrm{d}\left(\mathrm{Tr}\left(\mathbf{H}^\top\mathbf{D}^{-1}\mathbf{SH}\right)\right) &= \mathrm{Tr}\left(\mathbf{H}^\top\mathrm{d}\left(\mathbf{D}^{-1}\right)\mathbf{SH} + \mathbf{H}^\top\mathbf{D}^{-1}\mathrm{d}\left(\mathbf{S}\right)\mathbf{H}\right) \\
&= \mathrm{Tr}\left(-\mathbf{H}^\top\mathbf{D}^{-1}\mathrm{d}\big(\mathrm{Diag}\left(\mathbf{S1}\right)\big)\mathbf{D}^{-1}\mathbf{SH} + \mathbf{H}^\top\mathbf{D}^{-1}\mathrm{d}\left(\mathbf{S}\right)\mathbf{H}\right) \\
&= \mathrm{Tr}\left(-\mathbf{H}^\top\mathbf{D}^{-1}\mathrm{d}\left(\mathbf{S11}^\top \odot \mathbf{I}\right)\mathbf{D}^{-1}\mathbf{SH} + \mathbf{H}^\top\mathbf{D}^{-1}\mathrm{d}\left(\mathbf{S}\right)\mathbf{H}\right) \\
&= \mathrm{Tr}\left(-\mathbf{D}^{-1}\mathbf{SHH}^\top\mathbf{D}^{-1}\mathrm{d}\left(\mathbf{S11}^\top \circ \mathbf{I}\right) + \mathbf{HH}^\top\mathbf{D}^{-1}\mathrm{d}\left(\mathbf{S}\right)\right) \\
&= \mathrm{Tr}\left(-\left(\mathbf{D}^{-1}\mathbf{SHH}^\top\mathbf{D}^{-1} \circ \mathbf{I}\right)^\top\mathrm{d}\left(\mathbf{S11}^\top\right) + \mathbf{HH}^\top\mathbf{D}^{-1}\mathrm{d}\left(\mathbf{S}\right)\right) \\
&= \mathrm{Tr}\left(\left(-\mathbf{11}^\top\left(\mathbf{D}^{-1}\mathbf{SHH}^\top\mathbf{D}^{-1} \circ \mathbf{I}\right)^\top + \mathbf{HH}^\top\mathbf{D}^{-1}\right)\mathrm{d}\left(\mathbf{S}\right)\right).
\end{aligned}$$

Since

$$\mathrm{d}\left(\mathrm{Tr}\left(\mathbf{H}^\top\mathbf{D}^{-1}\mathbf{SH}\right)\right) = \mathrm{Tr}\left(\left(\frac{\partial\mathrm{Tr}\left(\mathbf{H}^\top\mathbf{D}^{-1}\mathbf{SH}\right)}{\partial\mathbf{S}}\right)^\top\mathrm{d}\left(\mathbf{S}\right)\right),$$

we can get

$$\begin{aligned}
\frac{\partial\mathrm{Tr}\left(\mathbf{H}^\top\mathbf{D}^{-1}\mathbf{SH}\right)}{\partial\mathbf{S}} &= \left(-\mathbf{11}^\top\left(\mathbf{D}^{-1}\mathbf{SHH}^\top\mathbf{D}^{-1} \circ \mathbf{I}\right)^\top + \mathbf{HH}^\top\mathbf{D}^{-1}\right)^\top \\
&= \mathbf{D}^{-1}\mathbf{HH}^\top - \mathrm{Diag}\left(\mathbf{D}^{-1}\mathbf{SHH}^\top\mathbf{D}^{-1}\right)\mathbf{1}^\top.
\end{aligned}$$

Therefore, we can have

$$\frac{\partial f_{\mathbf{S}}(\mathbf{S})}{\partial\mathbf{S}} = 2\gamma\left(\mathbf{S}-\mathbf{A}\right) - \lambda\left(\mathbf{D}^{-1}\mathbf{HH}^\top - \mathrm{Diag}\left(\mathbf{D}^{-1}\mathbf{SHH}^\top\mathbf{D}^{-1}\right)\mathbf{1}^\top\right) + 2\mu_2\mathbf{S},$$

where $f_{\mathbf{S}}(\mathbf{S})$ denotes the smooth part of the objective function of the $\mathbf{S}$-block, i.e.,

$$f_{\mathbf{S}}(\mathbf{S}) = \gamma\|\mathbf{S}-\mathbf{A}\|_{\mathrm{F}}^2 - \lambda\mathrm{Tr}\left(\mathbf{H}^\top\mathbf{D}^{-1}\mathbf{SH}\right) + \mu_2\left\|\mathbf{S}\right\|_{\mathrm{F}}^2. \tag{9}$$

Then, the k-th proximal step can be obtained as follows:

$$\mathbf{S}^{(k+1)} = \mathrm{prox}_{\eta_2\left(\mu_1\|\mathbf{S}\|_1 + \mathbb{I}_{\mathcal{S}}(\mathbf{S})\right)}\left(\mathbf{S}^{(k)} - \eta_2\frac{\partial f_{\mathbf{S}}(\mathbf{S}^{(k)})}{\partial\mathbf{S}^{(k)}}\right),$$

where η_2 is the step size and $\mathbb{I}_{\mathcal{S}}(\mathbf{S})$ denotes the indicator function taking value 0 if $\mathbf{S} \in \mathcal{S}$ and $+\infty$ otherwise.

The proximal operator above can be computed analytically as follows:

$$\mathbf{S}^{(k+1)} = \min\left\{1, \mathrm{ReLU}\Big(\mathbf{S}^{(k)} - \eta_2 \frac{\partial f_{\mathbf{S}}(\mathbf{S}^{(k)})}{\partial \mathbf{S}^{(k)}} - \eta_2\mu_1 \mathbf{1}\mathbf{1}^\top\Big)\right\}, \tag{10}$$

in which we first perform soft-thresholding and then project the solution to the constraint set $\mathcal{S}$. In the following, we give the proof that $\mathbf{S}^{(k)}$ in (10) is indeed the analytical expression of the proximal operator, i.e., it is the optimal solution to the non-smooth convex optimization problem :

$$\underset{\mathbf{S}}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{S} - \mathbf{M}\|_{\mathrm{F}}^2 + \eta_2\big(\mu_1\|\mathbf{S}\|_1 + \mathbb{I}_{\mathcal{S}}(\mathbf{S})\big),$$

where we denote $\mathbf{M} = \mathbf{S}^{(k)} - \eta_2 \frac{\partial f_{\mathbf{S}}(\mathbf{S}^{(k)})}{\partial \mathbf{S}^{(k)}}$ for notational simplicity. This optimization problem is decoupled over different elements in $\mathbf{S}$. Thus, we can individually optimize each $\mathbf{S}_{ij}, i,j = 1,\ldots,N$, by solving the following optimization problem:

$$\underset{\mathbf{S}_{ij}\in\mathcal{S}_{ij}}{\text{minimize}} \quad h(\mathbf{S}_{ij}) = \frac{1}{2}\|\mathbf{S}_{ij} - \mathbf{M}_{ij}\|_{\mathrm{F}}^2 + \eta_2\mu_1|\mathbf{S}_{ij}|, \tag{11}$$

where $\mathcal{S}_{ij} = \{\mathbf{S}_{ij} \in \mathbb{R} \mid 0 \le \mathbf{S}_{ij} \le 1\}$. The subgradient of $h(\mathbf{S}_{ij})$ can be computed as follows:

$$\partial h(\mathbf{S}_{ij}) = \begin{cases} \mathbf{S}_{ij} - \mathbf{M}_{ij} + \eta_2\mu_1 & \mathbf{S}_{ij} > 0 \\ \mathbf{S}_{ij} - \mathbf{M}_{ij} + \eta_2\mu_1\epsilon & \mathbf{S}_{ij} = 0, \end{cases}$$

where ϵ can be any constant satisfying $-1 \le \epsilon \le 1$.

According to the analytical expression (10), we can get

$$\mathbf{S}_{ij}^{(k+1)} = \min\{1, \mathrm{ReLU}(\mathbf{M}_{ij} - \eta_2\mu_1)\} = \begin{cases} 1 & \mathbf{M}_{ij} \ge 1 + \eta_2\mu_1 \\ \mathbf{M}_{ij} - \eta_2\mu_1 & 1 + \eta_2\mu_1 \ge \mathbf{M}_{ij} \ge \eta_2\mu_1 \\ 0 & \mathbf{M}_{ij} < \eta_2\mu_1. \end{cases}$$

In the following, we will show the optimality of $\mathbf{S}_{ij}^{(k+1)}$ by showing there exists a subgradient $\psi \in \partial h\big(\mathbf{S}_{ij}^{(k+1)}\big)$ such that $\psi\big(\mathbf{S}_{ij} - \mathbf{S}_{ij}^{(k+1)}\big) \ge 0$ for all $\mathbf{S}_{ij} \in \mathcal{S}_{ij}$. Observe that

$$\partial h(\mathbf{S}_{ij}^{(k+1)}) = \begin{cases} 1 - \mathbf{M}_{ij} + \eta_2\mu_1 & \mathbf{M}_{ij} \ge 1 + \eta_2\mu_1 \\ 0 & 1 + \eta_2\mu_1 \ge \mathbf{M}_{ij} \ge \eta_2\mu_1 \\ -\mathbf{M}_{ij} + \eta_2\mu_1\epsilon & \mathbf{M}_{ij} < \eta_2\mu_1. \end{cases}$$

Below, we will show that the optimality condition holds in each case.

1) For $\mathbf{M}_{ij} \ge 1 + \eta_2\mu_1$, we have $\psi = 1 - \mathbf{M}_{ij} + \eta_2\mu_1 \le 0$. Since $\mathbf{S}_{ij} - 1 \le 0$ for all $\mathbf{S}_{ij} \in \mathcal{S}_{ij}$, we can get $\psi\big(\mathbf{S}_{ij} - \mathbf{S}_{ij}^{(k+1)}\big) \ge 0$ for all $\mathbf{S}_{ij} \in \mathcal{S}_{ij}$.

2) For $1 + \eta_2\mu_1 \ge \mathbf{M}_{ij} \ge \eta_2\mu_1$, we have $\psi = 0$ and hence, $\psi\big(\mathbf{S}_{ij} - \mathbf{S}_{ij}^{(k+1)}\big) = 0$ for all $\mathbf{S}_{ij} \in \mathcal{S}_{ij}$.

3) For $\mathbf{M}_{ij} < \eta_2\mu_1$, we have $\psi = -\mathbf{M}_{ij} + \eta_2\mu_1\epsilon$ with ϵ being any constant satisfying $-1 \le \epsilon \le 1$. Thus, we can choose $\epsilon = 1$, which leads to $\psi > 0$. Since $\mathbf{S}_{ij} \ge 0$ for all $\mathbf{S}_{ij} \in \mathcal{S}_{ij}$, we can get $\psi\big(\mathbf{S}_{ij} - \mathbf{S}_{ij}^{(k+1)}\big) \ge 0$ for all $\mathbf{S}_{ij} \in \mathcal{S}_{ij}$.

In conclusion, we have $\psi\big(\mathbf{S}_{ij} - \mathbf{S}_{ij}^{(k+1)}\big) \ge 0$ for $\mathbf{S}_{ij} \in \mathcal{S}_{ij}$ in all cases and the optimality of $\mathbf{S}_{ij}^{(k)}$ for $\forall i,j = 1,\ldots,N$ is readily proved.

C Proof of Theorem 1 (Convergence of ASMP)

To ensure the monotonically decreasing property of (ASMP), the step size in $\mathbf{H}$-block must satisfy $\eta_1 < \frac{2}{L_H}$ and the step size in $\mathbf{S}$-block must satisfy $\eta_2 < \frac{2}{L_S}$ (Parikh and Boyd, 2014), where L_H and

L_S are the Lipschitz constants of $\nabla f_{\mathbf{H}}(\mathbf{H})$ and $\nabla f_{\mathbf{S}}(\mathbf{S})$, where $f_{\mathbf{H}}(\mathbf{H})$ is the objective function at the $\mathbf{H}$-block optimization problem, i.e.,

$$f_{\mathbf{H}}(\mathbf{H}) = \|\mathbf{H} - \mathbf{X}\|_{\mathrm{F}}^2 + \lambda \mathrm{Tr}\left(\mathbf{H}^\top \mathbf{L}_{\mathrm{rw}} \mathbf{H}\right),$$

and $f_{\mathbf{S}}(\mathbf{S})$ is defined in (9). Under such condition, the convergence of (ASMP) to a stationary point of (4) is readily obtained based on the results in Bolte et al. (2014); Nikolova and Tan (2017). In the following, we will derive the Lipschitz constants of $\nabla f_{\mathbf{H}}(\mathbf{H})$ and $\nabla f_{\mathbf{S}}(\mathbf{S})$ and give the conditions to ensure convergence of ASMP.

Denote $\mathbf{H}_1$ and $\mathbf{H}_2$ as two different variables for $f_{\mathbf{H}}$, we have

$$\begin{aligned}
\|\nabla f_{\mathbf{H}}(\mathbf{H}_1) - \nabla f_{\mathbf{H}}(\mathbf{H}_2)\|_{\mathrm{F}} &= \|(2(\mathbf{H}_1 - \mathbf{X}) + 2\lambda \mathbf{L}_{\mathrm{rw}}\mathbf{H}_1) - (2(\mathbf{H}_2 - \mathbf{X}) + 2\lambda \mathbf{L}_{\mathrm{rw}}\mathbf{H}_2)\|_{\mathrm{F}} \\
&= \|(2\mathbf{I} + 2\lambda \mathbf{L}_{\mathrm{rw}})(\mathbf{H}_1 - \mathbf{H}_2)\|_{\mathrm{F}} \\
&\leq 2\|\mathbf{I} + \lambda \mathbf{L}_{\mathrm{rw}}\|_2 \|\mathbf{H}_1 - \mathbf{H}_2\|_{\mathrm{F}}.
\end{aligned}$$

Since the largest eigenvalue of the normalized Laplacian matrix $\|\mathbf{L}_{\mathrm{rw}}\|_2 \leq 2$ (Chung, 1997), we can conclude that

$$\|\nabla f_{\mathbf{H}}(\mathbf{H}_1) - \nabla f_{\mathbf{H}}(\mathbf{H}_2)\|_{\mathrm{F}} \leq (2 + 4\lambda)\|\mathbf{H}_1 - \mathbf{H}_2\|_{\mathrm{F}}.$$

Therefore, function $\nabla f_{\mathbf{H}}(\mathbf{H})$ is L-smooth with Lipschitz constant $L_H = 2 + 4\lambda$.

Denote $\mathbf{S}_1$ and $\mathbf{S}_2$ as two different variables for $f_{\mathbf{S}}$. We have

$$\begin{aligned}
&\|\nabla f_{\mathbf{S}}(\mathbf{S}_1) - \nabla f_{\mathbf{S}}(\mathbf{S}_2)\|_{\mathrm{F}} \\
=&\Big\|\left(2\gamma(\mathbf{S}_1 - \mathbf{A}) + 2\mu_2\mathbf{S}_1 - \lambda \mathbf{D}_1^{-1}\mathbf{H}\mathbf{H}^\top + \lambda \mathrm{Diag}\left(\mathbf{D}_1^{-1}\mathbf{S}_1\mathbf{H}\mathbf{H}^\top\mathbf{D}_1^{-1}\right)\mathbf{1}^\top\right) \\
&- \left(2\gamma(\mathbf{S}_2 - \mathbf{A}) + 2\mu_2\mathbf{S}_2 - \lambda \mathbf{D}_2^{-1}\mathbf{H}\mathbf{H}^\top + \lambda \mathrm{Diag}\left(\mathbf{D}_2^{-1}\mathbf{S}_2\mathbf{H}\mathbf{H}^\top\mathbf{D}_2^{-1}\right)\mathbf{1}^\top\right)\Big\|_{\mathrm{F}},
\end{aligned}$$

where $\mathbf{D}_1 = \mathrm{Diag}(\mathbf{S}_1\mathbf{1})$ and $\mathbf{D}_2 = \mathrm{Diag}(\mathbf{S}_2\mathbf{1})$. Then, it can be upper bounded by

$$\begin{aligned}
\|\nabla f_{\mathbf{S}}(\mathbf{S}_1) - \nabla f_{\mathbf{S}}(\mathbf{S}_2)\|_{\mathrm{F}} \leq& \|(2\gamma + 2\mu_2)(\mathbf{S}_1 - \mathbf{S}_2)\|_{\mathrm{F}} + \left\|\lambda\left(\mathbf{D}_1^{-1} - \mathbf{D}_2^{-1}\right)\mathbf{H}\mathbf{H}^\top\right\|_{\mathrm{F}} + \\
&\left\|\lambda \mathrm{Diag}\left(\mathbf{D}_1^{-1}\mathbf{S}_1\mathbf{H}\mathbf{H}^\top\mathbf{D}_1^{-1} - \mathbf{D}_2^{-1}\mathbf{S}_2\mathbf{H}\mathbf{H}^\top\mathbf{D}_2^{-1}\right)\mathbf{1}^\top\right\|_{\mathrm{F}} \\
\leq& (2\gamma + 2\mu_2)\|\mathbf{S}_1 - \mathbf{S}_2\|_{\mathrm{F}} + \lambda\left\|\mathbf{H}\mathbf{H}^\top\right\|_2 \left\|\mathbf{D}_1^{-1} - \mathbf{D}_2^{-1}\right\|_{\mathrm{F}} \\
&+ \lambda\sqrt{N}\left\|\mathrm{Diag}\left(\mathbf{D}_1^{-1}\mathbf{S}_1\mathbf{H}\mathbf{H}^\top\mathbf{D}_1^{-1} - \mathbf{D}_2^{-1}\mathbf{S}_2\mathbf{H}\mathbf{H}^\top\mathbf{D}_2^{-1}\right)\right\|_{\mathrm{F}}.
\end{aligned}$$

First, observe that

$$\begin{aligned}
\left\|\mathbf{D}_1^{-1} - \mathbf{D}_2^{-1}\right\|_{\mathrm{F}} &= \left\|\mathrm{Diag}\Big((\mathbf{S}_1\mathbf{1})^{-1} - (\mathbf{S}_2\mathbf{1})^{-1}\Big)\right\|_{\mathrm{F}} \\
&= \sqrt{\sum_i \left(\frac{[\mathbf{S}_1]_{i,:}\mathbf{1} - [\mathbf{S}_2]_{i,:}\mathbf{1}}{\left([\mathbf{S}_1]_{i,:}\mathbf{1}\right)\left([\mathbf{S}_2]_{i,:}\mathbf{1}\right)}\right)^2} \\
&\leq \frac{1}{c^2}\sqrt{\sum_i \left(N \max_j \left|[\mathbf{S}_1]_{ij} - [\mathbf{S}_2]_{ij}\right|\right)^2} \\
&\leq \frac{1}{c^2} N \|\mathbf{S}_1 - \mathbf{S}_2\|_{\mathrm{F}},
\end{aligned}$$

in which we use the assumption $\min_i [\mathbf{D}]_{ii} = c > 0$. Since $\|\mathbf{H}_{i,:}\|_2 \leq B$, we have

$$\left\|\mathbf{H}\mathbf{H}^\top\right\|_2 \leq \left\|\mathbf{H}\mathbf{H}^\top\right\|_{\mathrm{F}} = \sqrt{\sum_{i,j}^{N}\left(\mathbf{H}_{i,:}^\top\mathbf{H}_j\right)^2} \leq \sqrt{\sum_{i,j}^{N} B^4} = NB^2.$$

Therefore, $\|\nabla f_{\mathbf{S}}(\mathbf{S}_1) - \nabla f_{\mathbf{S}}(\mathbf{S}_2)\|_{\mathrm{F}}$ can be further upper bounded by

$$\begin{aligned}
&\|\nabla f_{\mathbf{S}}(\mathbf{S}_1) - \nabla f_{\mathbf{S}}(\mathbf{S}_2)\|_{\mathrm{F}} \\
\leq&\left(2\gamma + 2\mu_2 + \frac{\lambda}{c^2}N^2B^2\right)\|\mathbf{S}_1 - \mathbf{S}_2\|_{\mathrm{F}} + \lambda\sqrt{N}\left\|\mathbf{D}_1^{-1}\mathbf{S}_1\mathbf{H}\mathbf{H}^\top\mathbf{D}_1^{-1} - \mathbf{D}_2^{-1}\mathbf{S}_2\mathbf{H}\mathbf{H}^\top\mathbf{D}_2^{-1}\right\|_{\mathrm{F}}
\end{aligned}$$

$$
\begin{aligned}
=&\left(2\gamma+2\mu_2+\frac{\lambda}{c^2}N^2B^2\right)\|\mathbf{S}_1-\mathbf{S}_2\|_{\mathrm{F}}+\lambda\sqrt{N}\left\|\mathbf{D}_1^{-1}\mathbf{S}_1\mathbf{H}\mathbf{H}^\top\mathbf{D}_1^{-1}-\mathbf{D}_1^{-1}\mathbf{S}_1\mathbf{H}\mathbf{H}^\top\mathbf{D}_2^{-1}\right\|_{\mathrm{F}}\\
&\qquad+\lambda\sqrt{N}\left\|\mathbf{D}_1^{-1}\mathbf{S}_1\mathbf{H}\mathbf{H}^\top\mathbf{D}_2^{-1}-\mathbf{D}_2^{-1}\mathbf{S}_2\mathbf{H}\mathbf{H}^\top\mathbf{D}_2^{-1}\right\|_{\mathrm{F}}\\
\leq&\left(2\gamma+2\mu_2+\frac{\lambda}{c^2}N^2B^2\right)\|\mathbf{S}_1-\mathbf{S}_2\|_{\mathrm{F}}+\lambda\sqrt{N}\left\|\mathbf{D}_1^{-1}\mathbf{S}_1\mathbf{H}\mathbf{H}^\top\right\|_{\mathrm{F}}\left\|\mathbf{D}_1^{-1}-\mathbf{D}_2^{-1}\right\|_2\\
&\qquad+\lambda\sqrt{N}\left\|\mathbf{D}_1^{-1}\mathbf{S}_1\mathbf{H}\mathbf{H}^\top-\mathbf{D}_2^{-1}\mathbf{S}_2\mathbf{H}\mathbf{H}^\top\right\|_{\mathrm{F}}\left\|\mathbf{D}_2^{-1}\right\|_2.
\end{aligned}
$$

Observe that

$$
\left\|\mathbf{D}^{-1}\right\|_2=\left\|\mathrm{Diag}\big((\mathbf{S}\mathbf{1})^{-1}\big)\right\|_2=\max_i\frac{1}{\mathbf{S}_{i,:}\mathbf{1}}\leq\frac{1}{c}
$$

and

$$
\left\|\mathbf{D}^{-1}\mathbf{S}\right\|_2\leq\left\|\mathbf{D}^{-1}\mathbf{S}\right\|_{\mathrm{F}}=\sqrt{\sum_{i,j}\left(\frac{\mathbf{S}_{ij}}{\mathbf{S}_{i,:}\mathbf{1}}\right)^2}\leq\frac{N}{c}.
$$

We further obtain

$$
\begin{aligned}
&\|\nabla f_{\mathbf{S}}(\mathbf{S}_1)-\nabla f_{\mathbf{S}}(\mathbf{S}_2)\|_{\mathrm{F}}\\
\leq&\Big(2\gamma+2\mu_2+\frac{\lambda}{c^2}N^2B^2+\frac{\lambda}{c^2}N\sqrt{N}\left\|\mathbf{D}_1^{-1}\mathbf{S}_1\mathbf{H}\mathbf{H}^\top\right\|_{\mathrm{F}}\Big)\|\mathbf{S}_1-\mathbf{S}_2\|_{\mathrm{F}}\\
&\qquad+\frac{\lambda}{c}\sqrt{N}\left\|\left(\mathbf{D}_1^{-1}\mathbf{S}_1-\mathbf{D}_2^{-1}\mathbf{S}_2\right)\mathbf{H}\mathbf{H}^\top\right\|_{\mathrm{F}}\\
\leq&\Big(2\gamma+2\mu_2+\frac{\lambda}{c^2}N^2B^2+\frac{\lambda}{c^2}N\sqrt{N}\left\|\mathbf{D}_1^{-1}\mathbf{S}_1\right\|_2\left\|\mathbf{H}\mathbf{H}^\top\right\|_{\mathrm{F}}\Big)\|\mathbf{S}_1-\mathbf{S}_2\|_{\mathrm{F}}\\
&\qquad+\frac{\lambda}{c}\sqrt{N}\left\|\mathbf{D}_1^{-1}\mathbf{S}_1-\mathbf{D}_2^{-1}\mathbf{S}_2\right\|_2\left\|\mathbf{H}\mathbf{H}^\top\right\|_{\mathrm{F}}\\
\leq&\Big(2\gamma+2\mu_2+\frac{\lambda}{c^2}N^2B^2+\frac{\lambda}{c^3}N^3\sqrt{N}B^2\Big)\|\mathbf{S}_1-\mathbf{S}_2\|_{\mathrm{F}}+\frac{\lambda}{c}N\sqrt{N}B^2\left\|\mathbf{D}_1^{-1}\mathbf{S}_1-\mathbf{D}_2^{-1}\mathbf{S}_2\right\|_2.
\end{aligned}
\tag{12}
$$

Also, note that

$$
\begin{aligned}
\left\|\mathbf{D}_1^{-1}\mathbf{S}_1-\mathbf{D}_2^{-1}\mathbf{S}_2\right\|_2&\leq\left\|\mathbf{D}_1^{-1}\mathbf{S}_1-\mathbf{D}_2^{-1}\mathbf{S}_2\right\|_{\mathrm{F}}\\
&\leq\left\|\mathbf{D}_1^{-1}\mathbf{S}_1-\mathbf{D}_1^{-1}\mathbf{S}_2\right\|_{\mathrm{F}}+\left\|\mathbf{D}_1^{-1}\mathbf{S}_2-\mathbf{D}_2^{-1}\mathbf{S}_2\right\|_{\mathrm{F}}\\
&\leq\left\|\mathbf{D}_1^{-1}\right\|_2\|\mathbf{S}_1-\mathbf{S}_2\|_{\mathrm{F}}+\|\mathbf{S}_2\|_2\left\|\mathbf{D}_1^{-1}-\mathbf{D}_2^{-1}\right\|_{\mathrm{F}}\\
&\leq\left\|\mathbf{D}_1^{-1}\right\|_2\|\mathbf{S}_1-\mathbf{S}_2\|_{\mathrm{F}}+N\left\|\mathbf{D}_1^{-1}-\mathbf{D}_2^{-1}\right\|_{\mathrm{F}}\\
&\leq\left(\frac{1}{c}+\frac{1}{c^2}N^2\right)\|\mathbf{S}_1-\mathbf{S}_2\|_{\mathrm{F}}.
\end{aligned}
\tag{13}
$$

Substituting (13) into (12) gives

$$
\begin{aligned}
\|\nabla f_{\mathbf{S}}(\mathbf{S}_1)-\nabla f_{\mathbf{S}}(\mathbf{S}_2)\|_{\mathrm{F}}&\leq\left(2\gamma+2\mu_2+\frac{\lambda}{c^2}N^2B^2+\frac{2\lambda}{c^3}N^3B^2\sqrt{N}+\frac{\lambda}{c^2}N\sqrt{N}B^2\right)\|\mathbf{S}_1-\mathbf{S}_2\|_{\mathrm{F}}\\
&\leq\left(2\gamma+2\mu_2+\frac{2\lambda}{c^2}N^2B^2+\frac{2\lambda}{c^3}N^3B^2\sqrt{N}\right)\|\mathbf{S}_1-\mathbf{S}_2\|_{\mathrm{F}}.
\end{aligned}
$$

Therefore, function $\nabla f_{\mathbf{S}}(\mathbf{S})$ is L-smooth with Lipschitz constant $L_S=2\gamma+2\mu_2+\frac{2\lambda}{c^2}N^2B^2+\frac{2\lambda}{c^3}N^3B^2\sqrt{N}$.

Based on the above results, we can conclude that $\nabla f_{\mathbf{H}}(\mathbf{H})$ and $\nabla f_{\mathbf{S}}(\mathbf{S})$ are L-smooth and the convergence of ASMP is guaranteed with $0<\eta_1<\frac{1}{L_H}$ and $0<\eta_2<\frac{2}{L_S}$.

D Discussion on the Joint Optimization Approach

We define the smooth part of the objective in (4) as follows:

$$
f\left(\mathbf{H},\mathbf{S}\right)=\|\mathbf{H}-\mathbf{X}\|_{\mathrm{F}}^2+\gamma\|\mathbf{S}-\mathbf{A}\|_{\mathrm{F}}^2+\lambda\mathrm{Tr}\left(\mathbf{H}^\top\mathbf{L}_{\mathrm{rw}}\mathbf{H}\right)+\mu_2\left\|\mathbf{S}\right\|_{\mathrm{F}}^2.
$$

In the following, we first derive the Lipschitz constant of $f(\mathbf{H}, \mathbf{S})$. Observe that

$$
\begin{aligned}
&\left\| \begin{bmatrix} \nabla_{\mathbf{H}} f(\mathbf{H}_1, \mathbf{S}_1) \\ \nabla_{\mathbf{S}} f(\mathbf{H}_1, \mathbf{S}_1) \end{bmatrix} - \begin{bmatrix} \nabla_{\mathbf{H}} f(\mathbf{H}_2, \mathbf{S}_2) \\ \nabla_{\mathbf{S}} f(\mathbf{H}_2, \mathbf{S}_2) \end{bmatrix} \right\|_{\mathrm{F}}^2 \\
=& \left\| \nabla_{\mathbf{H}} f(\mathbf{H}_1, \mathbf{S}_1) - \nabla_{\mathbf{H}} f(\mathbf{H}_2, \mathbf{S}_2) \right\|_{\mathrm{F}}^2 + \left\| \nabla_{\mathbf{S}} f(\mathbf{H}_1, \mathbf{S}_1) - \nabla_{\mathbf{S}} f(\mathbf{H}_2, \mathbf{S}_2) \right\|_{\mathrm{F}}^2 \\
=& \left\| \nabla_{\mathbf{H}} f(\mathbf{H}_1, \mathbf{S}_1) - \nabla_{\mathbf{H}} f(\mathbf{H}_1, \mathbf{S}_2) + \nabla_{\mathbf{H}} f(\mathbf{H}_1, \mathbf{S}_2) - \nabla_{\mathbf{H}} f(\mathbf{H}_2, \mathbf{S}_2) \right\|_{\mathrm{F}}^2 \\
&\qquad + \left\| \nabla_{\mathbf{S}} f(\mathbf{H}_1, \mathbf{S}_1) - \nabla_{\mathbf{S}} f(\mathbf{H}_2, \mathbf{S}_1) + \nabla_{\mathbf{S}} f(\mathbf{H}_2, \mathbf{S}_1) - \nabla_{\mathbf{S}} f(\mathbf{H}_2, \mathbf{S}_2) \right\|_{\mathrm{F}}^2 \\
\leq& L_H^2 \left\| \mathbf{H}_1 - \mathbf{H}_2 \right\|_{\mathrm{F}}^2 + L_S^2 \left\| \mathbf{S}_1 - \mathbf{S}_2 \right\|_{\mathrm{F}}^2 \\
&\qquad + \left\| \nabla_{\mathbf{H}} f(\mathbf{H}_1, \mathbf{S}_1) - \nabla_{\mathbf{H}} f(\mathbf{H}_1, \mathbf{S}_2) \right\|_{\mathrm{F}}^2 + \left\| \nabla_{\mathbf{S}} f(\mathbf{H}_1, \mathbf{S}_1) - \nabla_{\mathbf{S}} f(\mathbf{H}_2, \mathbf{S}_1) \right\|_{\mathrm{F}}^2 .
\end{aligned}
$$

Since $\left\| \mathbf{H}_{i,:} \right\|_2 \leq B$, we have $\left\| \mathbf{H} \right\|_{\mathrm{F}} \leq \sqrt{N} B$. Then we can obtain

$$
\begin{aligned}
\left\| \nabla_{\mathbf{H}} f(\mathbf{H}_1, \mathbf{S}_1) - \nabla_{\mathbf{H}} f(\mathbf{H}_1, \mathbf{S}_2) \right\|_{\mathrm{F}} &= \left\| 2\lambda \left(\mathbf{I} - \mathbf{D}_1^{-1} \mathbf{S}_1 \right) \mathbf{H}_1 - 2\lambda \left(\mathbf{I} - \mathbf{D}_2^{-1} \mathbf{S}_2 \right) \mathbf{H}_1 \right\|_{\mathrm{F}} \\
&\leq 2\lambda \left\| \mathbf{D}_1^{-1} \mathbf{S}_1 - \mathbf{D}_2^{-1} \mathbf{S}_2 \right\|_2 \left\| \mathbf{H}_1 \right\|_{\mathrm{F}} \\
&\leq \left(\frac{2\lambda}{c} \sqrt{N} B + \frac{2\lambda}{c^2} N^2 \sqrt{N} B \right) \left\| \mathbf{S}_1 - \mathbf{S}_2 \right\|_{\mathrm{F}} .
\end{aligned}
$$

Besides, we can upper bound $\left\| \nabla_{\mathbf{S}} f(\mathbf{H}_1, \mathbf{S}_1) - \nabla_{\mathbf{S}} f(\mathbf{H}_2, \mathbf{S}_1) \right\|_{\mathrm{F}}$ as follows:

$$
\begin{aligned}
&\left\| \nabla_{\mathbf{S}} f(\mathbf{H}_1, \mathbf{S}_1) - \nabla_{\mathbf{S}} f(\mathbf{H}_2, \mathbf{S}_1) \right\|_{\mathrm{F}} \\
\leq& \left\| \lambda \mathbf{D}_1^{-1} \left(\mathbf{H}_1 \mathbf{H}_1^\top - \mathbf{H}_2 \mathbf{H}_2^\top \right) - \lambda \mathrm{Diag} \left(\mathbf{D}_1^{-1} \mathbf{S}_1 \mathbf{H}_1 \mathbf{H}_1^\top \mathbf{D}_1^{-1} - \mathbf{D}_1^{-1} \mathbf{S}_1 \mathbf{H}_2 \mathbf{H}_2^\top \mathbf{D}_1^{-1} \right) \mathbf{1}^\top \right\|_{\mathrm{F}} \\
\leq& \lambda \left\| \mathbf{D}_1^{-1} \right\|_2 \left\| \mathbf{H}_1 \mathbf{H}_1^\top - \mathbf{H}_2 \mathbf{H}_2^\top \right\|_{\mathrm{F}} + \lambda \sqrt{N} \left\| \mathbf{D}_1^{-1} \mathbf{S}_1 \right\|_2 \left\| \mathbf{H}_1 \mathbf{H}_1^\top - \mathbf{H}_2 \mathbf{H}_2^\top \right\|_{\mathrm{F}} \left\| \mathbf{D}_1^{-1} \right\|_2 \\
\leq& \left(\frac{\lambda}{c} + \frac{\lambda}{c^2} N \sqrt{N} \right) \left\| \mathbf{H}_1 \mathbf{H}_1^\top - \mathbf{H}_2 \mathbf{H}_2^\top \right\|_{\mathrm{F}} .
\end{aligned}
$$

Note that

$$
\begin{aligned}
\left\| \mathbf{H}_1 \mathbf{H}_1^\top - \mathbf{H}_2 \mathbf{H}_2^\top \right\|_{\mathrm{F}} &= \left\| \mathbf{H}_1 \mathbf{H}_1^\top - \mathbf{H}_1 \mathbf{H}_2^\top + \mathbf{H}_1 \mathbf{H}_2^\top - \mathbf{H}_2 \mathbf{H}_2^\top \right\|_{\mathrm{F}} \\
&= \left\| \mathbf{H}_1 \left(\mathbf{H}_1^\top - \mathbf{H}_2^\top \right) \right\|_{\mathrm{F}} + \left\| \left(\mathbf{H}_1 - \mathbf{H}_2 \right) \mathbf{H}_2^\top \right\|_{\mathrm{F}} \\
&\leq \left(\left\| \mathbf{H}_1 \right\|_2 + \left\| \mathbf{H}_2 \right\|_2 \right) \left\| \mathbf{H}_1 - \mathbf{H}_2 \right\|_{\mathrm{F}} \\
&\leq 2\sqrt{N} B \left\| \mathbf{H}_1 - \mathbf{H}_2 \right\|_{\mathrm{F}} ,
\end{aligned}
$$

then we have

$$
\left\| \nabla_{\mathbf{S}} f(\mathbf{H}_1, \mathbf{S}_1) - \nabla_{\mathbf{S}} f(\mathbf{H}_2, \mathbf{S}_1) \right\|_{\mathrm{F}} \leq \left(\frac{2\lambda}{c} \sqrt{N} B + \frac{2\lambda}{c^2} N^2 B \right) \left\| \mathbf{H}_1 - \mathbf{H}_2 \right\|_{\mathrm{F}} .
$$

Therefore, we can conclude that

$$
\begin{aligned}
\left\| \begin{bmatrix} \nabla_{\mathbf{H}} f(\mathbf{H}_1, \mathbf{S}_1) \\ \nabla_{\mathbf{S}} f(\mathbf{H}_1, \mathbf{S}_1) \end{bmatrix} - \begin{bmatrix} \nabla_{\mathbf{H}} f(\mathbf{H}_2, \mathbf{S}_2) \\ \nabla_{\mathbf{S}} f(\mathbf{H}_2, \mathbf{S}_2) \end{bmatrix} \right\|_{\mathrm{F}}^2 \leq& \left(L_H^2 + \left(\frac{2\lambda}{c} \sqrt{N} B + \frac{2\lambda}{c^2} N^2 B \right)^2 \right) \left\| \mathbf{H}_1 - \mathbf{H}_2 \right\|_{\mathrm{F}}^2 \\
&+ \left(L_S^2 + \left(\frac{2\lambda}{c} \sqrt{N} B + \frac{2\lambda}{c^2} N^2 \sqrt{N} B \right)^2 \right) \left\| \mathbf{S}_1 - \mathbf{S}_2 \right\|_{\mathrm{F}}^2 .
\end{aligned}
$$

Thus, function $\nabla f(\mathbf{H}, \mathbf{S})$ is L-smooth with Lipschitz constant

$$
L = \max \left\{ \sqrt{L_H^2 + \left(\frac{2\lambda}{c} \sqrt{N} B + \frac{2\lambda}{c^2} N^2 B \right)^2}, \sqrt{L_S^2 + \left(\frac{2\lambda}{c} \sqrt{N} B + \frac{2\lambda}{c^2} N^2 \sqrt{N} B \right)^2} \right\} . \tag{14}
$$

This result indicates that the Lipschitz constant L is larger then the L_H and L_S. Moreover, recall that $L_H = 2 + 4\lambda$. Since in practice, N is generally much larger than other constants, i.e., c, λ, and B, if we use the joint optimization approach instead of the alternating optimization approach, the Lipschitz constant with respect to variable $\mathbf{H}$ will increase by a large margin.

After deriving the Lipshitz constants for both the joint optimization approach and the alternating optimziation approach, we further compare there convergence rates below. Denote $\left\{ \mathbf{H}^{(k)}, \mathbf{S}^{(k)} \right\}_{k=0}^{K}$

as the sequence generated either by the above joint optimization approach or by the alternating optimization approach used in ASMP. Following the convergence results in Bolte et al. (2014); Nikolova and Tan (2017), we can conclude that for any $K \in \mathbb{N}$, the following inequality holds:

$$\inf_{k \geq K} \left\{ \|\mathbf{H}^{(k+1)} - \mathbf{H}^{(k)}\|^2 + \left\|\mathbf{S}^{(k+1)} - \mathbf{S}^{(k)}\right\|^2 \right\} \leq \frac{1}{\rho K} \left(p\left(\mathbf{H}^{(0)}, \mathbf{S}^{(0)}\right) - p^* \right),$$

where ρ is a constant depending on step sizes and Lipschitz constant, $p\left(\mathbf{H}, \mathbf{S}\right)$ represents the objective function in (4), and p^* denotes the minimum of $p\left(\mathbf{H}, \mathbf{S}\right)$. Theoretically, for the joint optimization approach, the maximum step size we can choose to guarantee the sufficient descent of the objective at each step is $\rho = \frac{1}{\eta} - \frac{L}{2}$, while for the alternating optimization approach, the requirement of ρ is relaxed to $\min\left\{\frac{1}{\eta_1} - \frac{L_H}{2}, \frac{1}{\eta_2} - \frac{L_S}{2}\right\}$. Due to the fact that $L > \max\left\{L_H, L_S\right\}$ as shown in (14), the alternating minimization method is allowed to adopt a larger step size at each block than the joint optimization approach, resulting in a faster convergence behavior of the sequence. Motivated by this fact, we develop ASMP based on the alternating procedure rather than the joint one so that the resulting message passing structure contains fewer layers to achieve the similar or even better numerical performance compared to the joint one.

E Extension of ASGNN

In this paper, we focus on problems with a given graph structure, while ASGNN is also applicable when the initial structure is not available. In such a case, we can first create a k-nearest neighbor graph or use some optimization methods (Dong et al., 2016; Kalofolias, 2016; Kumar et al., 2020) to learn a graph structure based on the node features. Alternatively, we can use a neural network (Shrivastava et al., 2020; Pu et al., 2021) to generate an initial graph that is jointly learned with ASGNN. The adaptive structure in ASGNN can also help refine the generated graph and such use of ASGNN is a promising future research direction.

F Convergence Property of ASMP in Practice

To evaluate the convergence of ASMP with learned step sizes, we conduct experiments on Cora, Citeseer, and Cora-ML datasets at a 25% perturbation rate under meta-attack. Specially, we train a 4-layer ASGNN model and evaluate the objective function values in different layers. Since we use a recurrent structure in ASGNN, i.e., the step sizes used in different layers are the same, we are able to extend the trained 4-layer ASGNN model to a deeper one. The values of the objective function (4) are showcased in Figure 2, in which we normalize the objective values by dividing the objective value in the first iteration. From Figure 2, we conclude that the ASMP with learned step sizes can monotonically decrease the objective function value during the message passing process in the first 16 layers. Note that although the monotonic decreasing property does not hold in 16-18 layers in the Cora-Ml dataset, it is mainly because the step sizes are learned from a 4-layer model. The results indicate that although the learned step sizes do not satisfy the results in Theorem 1, they still ensure the monotonic decrease of the objective function value.

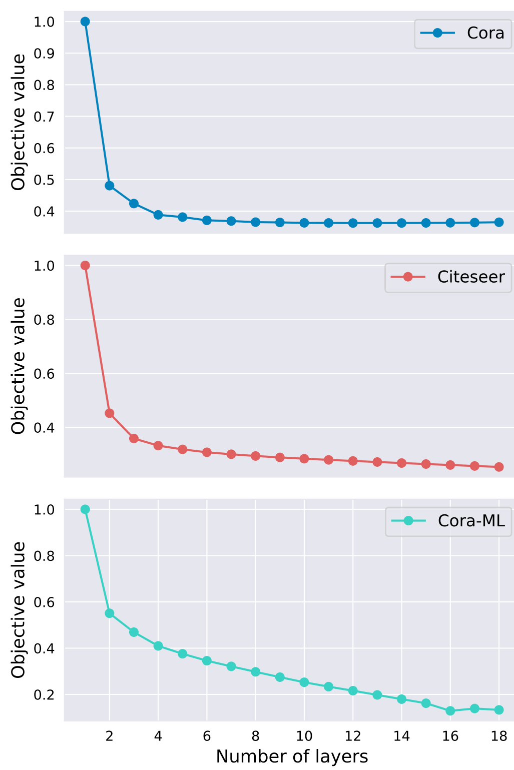

Figure 2: The value of the objective in (4) during ASMP.