UNBIASED STOCHASTIC PROXIMAL SOLVER FOR GRAPH NEURAL NETWORKS WITH EQUILIBRIUM STATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Neural Networks (GNNs) are widely used deep learning models that can extract meaningful representations from graph datasets and achieve great success in many machine learning tasks. Among them, graph neural networks with iterative iterations like unfolded GNNs and implicit GNNs can effectively capture long-range dependencies in graphs and demonstrate superior performance on large graphs since they can mathematically ensure its convergence to some nontrivial solution after lots of aggregations. However, the aggregation time for such models costs a lot as they need to aggregate the full graph in each update. Such weakness limits the scalability of the implicit graph models. To tackle such limitations, we propose two unbiased stochastic proximal solvers inspired by the stochastic proximal gradient descent method and its variance reduction variant called USP and USP-VR solvers. From the point of stochastic optimization, we theoretically prove that our solvers are unbiased, which can converge to the same solution as the original solvers for unfolded GNNs and implicit GNNs. Furthermore, the computation complexities for unfolded GNNs and implicit GNNs with our proposed solvers are significantly less than their vanilla versions. Experiments on various large graph datasets show that our proposed solvers are more efficient and can achieve state-of-the-art performance.

1 INTRODUCTION

Graph Neural Networks (GNNs) (Zhou et al., 2020; Wu et al., 2020) can effectively aggregate information from its neighbors and then encode graph information into meaningful representations and have been widely used to extract meaningful representations of nodes in graph-structured data recently. Furthermore, Graph Convolution Networks (GCNs)(Kipf & Welling, 2016) involve the convolution structure in the GNNs and drastically improve the performance on a wide range of tasks like computer vision (Xu et al., 2020b), recommendation systems (He et al., 2020; Zhang et al., 2020b) and biochemical researches (Mincheva & Roussel, 2007; Wan et al., 2019). Due to these results, GCN models have attracted a lot attention and various techniques have be proposed recently, including graph attention (Veličković et al., 2017), layer normalization (Zhao & Akoglu, 2019), linearization (Wu et al., 2019) and others (Klicpera et al., 2018; Rong et al., 2020; Chen et al., 2021).

Current GNN models usually capture topological information of T-hops by performing T iterations graph aggregation. However, T cannot be large. Otherwise, their outputs may degenerate to some trivial points and such a phenomenon is called over-smoothing (Yang et al., 2020; Li et al., 2019). Due to this reason, traditional GNNs cannot discover the dependency with longer ranges. In order to tackle these problems, researchers have proposed some graph neural networks with iterative update algorithms which can finally converge to some non-trivial points (Yang et al., 2021a;b). The implicit graph neural networks (IGNNs) (Gu et al., 2020) and Convergent Graph Solvers (CGS) (Park et al., 2021) can also be regarded as a type of such models. Since these models will finally converge to a equilibrium state (stationary points or fixed points), we call these models **Graph Equilibrium Models** for convenience in the following paper.

Above graph equilibrium models enjoy superior advantages in capturing the long-range information because they implicitly finish the "huge hop" aggregation via its forward procedure. However, graph equilibrium models have to recursively aggregate the neighborhoods of graph nodes since solving their equilibrium state needs iteratively aggregating the full graph. Therefore, it needs expensive computation costs to deal with large graphs especially when they are dense. Although many works (Chen et al., 2018; Hamilton et al., 2017) propose different aggregation methods through the sampling nodes in traditional graph models, there are no guarantees for the convergence and unbiased approximation when applying them to the graph equilibrium models.

For the above reasons, how to efficiently obtain the outputs for these graph equilibrium models is an interesting problem worth exploring. Since Yang et al. (2021b); Zhu et al. (2021); Zhang et al. (2020a) have demonstrated that the implicit and the unfolded graph neural networks can be regarded as solving the graph denoising problem, we are inspired to study the efficiency of these models from the optimization view. Then we propose two stochastic solvers for these graph equilibrium models with convergence guarantees from the perspective of solving the problem via stochastic proximal gradient descent methods. Since we only need to aggregate subgraphs to obtain final outputs, our proposed solvers are much more efficient than vanilla deterministic solvers by gradient descent or fixed-point iterations. Furthermore, we can theoretically prove that our solvers can obtain the unbiased output as the vanilla deterministic solvers do.

Our Contributions. We summarize the contributions of our methods as follows:

- By splitting the graph denoising optimization for the equilibrium states as several sub-optimization problems, we can treat the forward procedure for the graph equilibrium models as solving the proper finite-sum optimization problem. Then we propose two stochastic solvers for the graph equilibrium models: **Unbiased Stochastic Proximal Solver** (**USP solver**) and its variant with variance reduction **USP-VR solver**.
- Compared with the vanilla deterministic solvers which aggregate the full graph for graph equilibrium models' forward procedure, our USP solver and its variant only need to aggregate subgraphs to reach the equilibrium. Therefore, graph equilibrium models can be more efficient than before with our stochastic solvers.
- We theoretically prove that our USP solvers can converge to the same output as the graph equilibrium models' with their original deterministic forward procedure in expectation. Furthermore, we also empirically demonstrate the advantages of our proposed method with various experiments.

2 Related Works

2.1 Graph Neural Networks

Due to GNNs' impressive performance, GNNs have gained influence for graph-related works. Most Graph Neural Networks (Kipf & Welling, 2016; Veličković et al., 2017; Wu et al., 2019; Xu et al., 2018) aggregate the graph information for graph tasks. However, these models only involve graph aggregation of finite times due to the over-smoothing problem. Thereby, they can hardly capture very long-range dependency. Contrary to these models, implicit graph models (Liu et al., 2021a; Gu et al., 2020; Park et al., 2021) aggregate the graph information for a lot of iterations or "infinite" iterations with theoretically non-trivial equilibrium outputs.

Moreover, recent works have been done to explore the connections between the graph neural models and the graph denoising optimization problem. Some works (Zhu et al., 2021; Zhang et al., 2020a) recover different graph models to various graph denoising problems. Furthermore, Zhu et al. (2021) also proposed two types of models by reformulating the graph denoising problem from the spectral filter perspective. Some researchers focus on interpreting existing graph attention and proposing new attention modules from the view of redesigning the correlated graph denoising models (Yang et al., 2021a; Ma et al., 2021; Yang et al., 2021b).

In addition to these works, researchers also pay attention to GNNs from a different perspective. For example, robust graph neural networks (Jin et al., 2020; Luo et al., 2021), pretraining graph neural networks (Hu et al., 2019b; Qiu et al., 2020), explanations for graph neural networks (Ying et al., 2019; Yuan et al., 2020) and connections to differential systems (Xu et al., 2020a; Chamberlain et al., 2021; Wang et al., 2021).

2.2 Graph Equilibrium Models and implicit models

Since the implicit graph models (Gu et al., 2020) and unfolded graph Yang et al. (2021a); Liu et al. (2021c); Klicpera et al. (2018)'s forward procedure will converge to the equilibrium state of the corresponding optimization problem and Yang et al. (2021b) has already shown their relationships, we call these models the graph equilibrium models in the following. Yang et al. (2021a) briefly discusses the deterministic proximal gradient descent method's implementation for explicit graph models. Different regularizers' impacts on graph models' robustness have also been explored in Liu et al. (2021b;c) by adding them and reformulating their correlated optimization problems. Implicit graph models(El Ghaoui et al., 2021) are new types of graph neural architectures inspired by the deep equilibrium models (Bai et al., 2019; Li et al., 2021) in deep learning whose outputs are determined implicitly by the designation of fixed-point equations with graph aggregation. Since these models' forward procedures are all trying to reach the equilibrium points by iteratively aggregating on graphs, these models can capture the long-range information of graphs. Therefore, the graph equilibrium models can perform well on various large-scale graphs. However, as these models need aggregating whole graphs lots of times to reach the outputs, the computation complexity is large when dealing with large and dense graphs if we use the original solvers for the graph equilibrium models.

2.3 Graph Neural Networks for Large Graphs

Aggregating through large or dense graphs needs huge computation complexity. Different training or inference strategy have been proposed to reduce the computation cost for training and testing procedures. GraphSAINT (Zeng et al., 2019) and Cluster-GCN (Chiang et al., 2019) propose subgraph sampling and restrict the back-propagation within the subgraphs to reduce the complexity of training. VR-GCN (Chen et al., 2017) uses the variance elimination method for training to improve the training performance. Narayanan et al. (2021) uses lazy update for weight parameters' training. GraphSAGE (Hamilton et al., 2017) update a sub-sampled neighborhood of a node during training and inferencing to speed up GNNs. FastGCN (Chen et al., 2018) and others (Huang et al., 2018; Zou et al., 2019) use sampling nodes instead for each GCN layer for fast graph aggregation during mini-batch training and full graph testing. However, these methods are not designed or analyzed for graph equilibrium models. Thereby, we propose two unbiased stochastic solver for graph equilibrium models inspired by the stochastic optimization algorithm and give the analysis for the convergence of their output.

3 Graph Equilibrium Models Using the Unbiased Stochastic Solver

3.1 Preliminaries and Settings

According to former works, implicit graph neural networks and other unfolded graph neural networks' forward procedure to get the output features $\mathbf{Z}$ before prediction for given input $\mathbf{X}$ can be formulated as the following equation:

$$\mathbf{Z}^{(n+1)} = \sigma\left(\mathbf{Z}^{(n)} - \gamma\mathbf{Z}^{(n)} + \gamma\mathbf{B} - \gamma\tilde{\mathbf{A}}\mathbf{Z}\mathbf{W}\mathbf{W}^\top\right), \tag{1}$$

with $\tilde{\mathbf{A}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ denotes the Laplacian matrix, $\mathbf{A}$ is the adjacent matrix, input injection is $\mathbf{B} = \tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_B$ as former works (Gu et al., 2020; Park et al., 2021) and σ is the activation function. It can be viewed as the implicit graph neural networks when $\gamma = 1$ and otherwise it can be regarded as other unfolded graph neural networks. The above equation can also be regarded as solving the following graph denoising problems by deterministic gradient descent with step size γ,

$$\min_{\mathbf{Z}} G(\mathbf{Z}) = g(\mathbf{Z}) + f(\mathbf{Z}) = \frac{1}{2}\|\mathbf{Z}-\mathbf{B}\|_F^2 + \frac{1}{2}tr\left((\mathbf{Z}\mathbf{W})^\top \tilde{\mathbf{A}}\,(\mathbf{Z}\mathbf{W})\right) + f(\mathbf{Z}), \tag{2}$$

where f is the proximal function that tries to constrain the output features and can induce the widely used activation functions by different choices. For example, if we choose to constrain the output features to be non-negative, then the corresponding graph equilibrium models will use ReLU activation functions as their activation. Furthermore, the above formulation is the general formulation for different models. IRLS (Yang et al., 2021a) is setting $\mathbf{W} = \mathbf{I}$ and APPNP (Klicpera et al., 2018) needs weight $\mathbf{W}$ and activation all be identity. Although our formulation will induce the symmetric weight structure $\mathbf{W}\mathbf{W}^\top$, such relaxation won't influence the performance much as illustrated in Hu et al. (2019a). In order to solve the above problem to get the equilibrium state, most works iteratively using the deterministic gradient descent method like Eqn (1). However, such a solver is time-consuming especially when we need to aggregate large graphs in each update. Therefore, we are going to propose new solvers for the above graph equilibrium models to make them more efficient. Before that, we are going to make some assumptions for the convenience of our analysis.

Firstly, we assume $\|\mathbf{W}\mathbf{W}^\top\|_2 \leq 1/\tilde{\mathbf{A}}$ which can easily achieved by weight normalization (Salimans & Kingma, 2016) and such technique will not influence the performance much as other works (Bai et al., 2020; Li et al., 2021) show. Furthermore, we assume the derivative of $g(\mathbf{Z})$ are Lipschitz continuous with Lipschitz constant L satisfying $1 < L < 2$, which can also be achieved by weight normalization. $G(\mathbf{Z})$ is strongly convex with $\mu = 1$, which can easily be achieved for many activation functions including ReLU. And in our work, we mainly focus on the graph equilibrium models without attention. Therefore, we take two typical graph equilibrium models as our baselines: IRLS-Base with ReLU activation and IGNN.

3.2 Solve Graph Equilibrium models via Unbiased Stochastic Solver

In order to obtain the solutions in a stochastic way, we first reformulate the graph optimization problem (2) from the view of edges as follows:

$$\min_{\mathbf{Z}} G(\mathbf{Z}) = g(\mathbf{Z}) + f(\mathbf{Z}) = \left(\frac{1}{2}\|\mathbf{Z} - \mathbf{B}\|_F^2 + \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} \tilde{\mathbf{A}}_{ij}\|\mathbf{z}_i\mathbf{W} - \mathbf{z}_j\mathbf{W}\|_2^2 \right) + f(\mathbf{Z}), \quad (3)$$

where $\mathcal{E}$ is the full edges set of the graph, $(i,j) \in \mathcal{E}$ means the edge between node i and j exist in the edge set $\mathcal{E}$, $\mathbf{z}_i$ is output feature vector for the i-th node and $\tilde{\mathbf{A}}_{ij}$ denotes (i,j)-th element of original Laplacian $\tilde{\mathbf{A}}$. Viewing graph edges as samples of the dataset, the original algorithm can be viewed as using the global proximal gradient descent method for the problem (3). However, acquiring the global gradient in each iteration is expansive as we demonstrate above. In order to make the whole structure more efficient, we straightforwardly convert the deterministic solvers for graph equilibrium models to the stochastic scheme.

First, we separate the global graph denoising objective g into m sub-objectives:

$$g_{\hat{\mathcal{E}}_k}(\mathbf{Z}) = \frac{1}{2}\|\mathbf{Z} - \mathbf{B}\|_F^2 + \frac{m}{2}\sum_{(i,j)\in\hat{\mathcal{E}}_k} \tilde{\mathbf{A}}_{ij}\|\mathbf{z}_i\mathbf{W} - \mathbf{z}_j\mathbf{W}\|_2^2, \quad (4)$$

where $\hat{\mathcal{E}}_k \subset \mathcal{E}$ is the k-th subset of edges which are randomly split into m sets from the full graph set and each contains around $\lfloor \frac{|\mathcal{E}|}{m} \rfloor$ edges. Then the original optimization problem can be regarded as a finite sum optimization problem: $G(\mathbf{Z}) = \frac{1}{m}\sum_{k=1}^m g_{\hat{\mathcal{E}}_k}(\mathbf{Z}) + f(\mathbf{Z})$.

Then we can use the stochastic proximal gradient method by randomly choosing $k \in \{1, \cdots, m\}$ with the same probability and calculate the stochastic gradient for updating as follows:

$$\mathbf{V}^{(t)} = \mathbf{Z}^{(t)} + m\hat{\mathbf{A}}_{\hat{\mathcal{E}}_k}\mathbf{Z}^{(t)}\mathbf{W}\mathbf{W}^\top - \mathbf{B}, \quad (5)$$

$$\mathbf{Z}^{(t+1)} = \sigma\left(\mathbf{Z}^{(t)} - \eta_t\mathbf{V}^{(t)}\right), \quad (6)$$

where $\eta_t = \frac{1}{t}$ is the step size and $\hat{\mathbf{A}}_{\hat{\mathcal{E}}_k}$ is the laplacian matrix for $g_{\hat{\mathcal{E}}_k}$ with $m\mathbb{E}_k\hat{\mathbf{A}}_{\hat{\mathcal{E}}_k} = \tilde{\mathbf{A}}$ and its (k_1, k_2)-th element defined as:

$$\hat{\mathbf{A}}_{\hat{\mathcal{E}}_k}(k_1, k_2) = \begin{cases} \sum_{(k1,j)\in\hat{\mathcal{E}}_k} \tilde{\mathbf{A}}_{k_1 j}, & if\ k_1 = k_2, \\ \tilde{\mathbf{A}}_{k_1 k_2}, & if\ (k_1, k_2) \in \hat{\mathcal{E}}_k, \\ 0, & otherwise. \end{cases}$$

We call the graph equilibrium models "A" using our solvers are Eqn (6) as the "A+USP" in the following. "A+USP" can converge to the equilibrium state $\mathbf{Z}^*$ as the following proposition states:

Proposition 1. *With the assumptions in Section 3.1, the expectations of our uniformly sampled stochastic gradient defined in Eqn (5) is the same as the $g(\mathbf{Z})$'s gradient, i.e.* $\mathbb{E}_{\hat{\mathcal{E}}} \nabla_{\mathbf{Z}} \left[g_{\hat{\mathcal{E}}_k}(\mathbf{Z}) \right] = \nabla_{\mathbf{Z}} g(\mathbf{Z})$. *Furthermore, we can conclude that with our USP solver Eqn (6) and* $\eta_t = \frac{1}{t}$*, the result will converge to the solution of the global objective Eqn (3) in expectation with the sub-linear convergent rate, i.e.,*

$$\mathbb{E} \left\| \mathbf{Z}^{(t)} - \mathbf{Z}^* \right\|_F^2 = O\left(\frac{1}{t}\right). \tag{7}$$

Proofs can be found in the Appendix. The above proposition indicates that iteratively forwarding the stochastic proximal gradient descent step Eqn (6) can finally get the solution for the problem Eqn (3), which means that our proposed stochastic solver is unbiased. Since the inner iterations Eqn (5) and (6) only need subgraphs instead of the full graph, our stochastic proximal solver can be much more efficient than the original solvers for graph equilibrium models in practice. However, the USP solver's forward procedure can only provide sub-linear convergence. Thereby, the outputs of our proposed solver will be less accurate than the vanilla deterministic models when their propagation times are limited during the forward procedure in practice and may influence performance. To solve such a problem, we propose another stochastic solver with the linear convergent rate in the following section.

3.3 Solve Graph Equilibrium Models via Unbiased Stochastic Solver with Variance Reduction

Although using the stochastic proximal gradient defined in Eqn. (5) is unbiased, the large variance of the stochastic gradient still hinders our stochastic solver's convergence speed. A straightforward way to reduce the variance is utilizing the full gradient to correct the gradient every m_2 iteration. For example, if next m_2 iterations of graph equilibrium models' forwarding are initialized with $\mathbf{Z}^{(0)} = \tilde{\mathbf{Z}}$, then for the k-th iteration with $k \geq 1$, we can use the modified gradient $\hat{\mathbf{V}}^{(k)}$ to replace the original $\mathbf{V}^{(k)}$ in Eqn (5):

$$\hat{\mathbf{V}}^{(k)} = \nabla g_{\hat{\mathcal{E}}_{i_k}}(\mathbf{Z}^{(k-1)}) - \nabla g_{\hat{\mathcal{E}}_{i_k}}(\tilde{\mathbf{Z}}) + \nabla g(\tilde{\mathbf{Z}}), \tag{8}$$

with i_k sampled randomly from 1 to m with the same probability. Moreover, we can easily conclude that the modified direction $\hat{\mathbf{V}}_k$ is unbiased and have a smaller variance than $\mathbf{V}^{(k)}$ in Eqn (5):

$$\mathbb{E}\left[\hat{\mathbf{V}}^{(k)}\right] = \nabla g(\mathbf{Z}^{(k-1)}),$$
$$\mathbb{E}\left\|\hat{\mathbf{V}}^{(k)} - \nabla g(\mathbf{Z}^{(k-1)})\right\|^2 \leq \mathbb{E}\left\|\mathbf{V}^{(k)} - \nabla g(\mathbf{Z}^{(k-1)})\right\|^2.$$

Replacing the original stochastic gradient with this modified version in the USP solver, we obtain our Unbiased Stochastic Proximal Solver with Variance Reduction (USP-VR) which is inspired by the variance reduction algorithm proposed in Xiao & Zhang (2014). The procedure for our USP-VR solver is listed in Alogrithm 1.

Since the global optimization problem $G(\mathbf{Z})$ is strongly convex, the USP-VR solver's multi-stage procedure can progressively reduce the variance of the stochastic gradient $\mathbf{V}_k$ and both $\tilde{\mathbf{Z}}, \mathbf{Z}^{(k)}$ will finally converge to $\mathbf{Z}^*$ with proper step size η.

Proposition 2. *With the same settings and assumptions in Prposition 1, we can conclude that our proposed USP-VR solver can converge to the equilibrium state with step size* $\eta < \frac{1}{8L_{\max}}$ *and sufficiently large* m_2 *so that*

$$\rho = \frac{8L_{\max}\eta^2(m_2+1)+1}{(\eta - 8L_{\max}\eta^2)m_2} < 1.$$

where $L_{\max} = \max_k L_{g_{\hat{\mathcal{E}}_k}}$ *where* L_i *are the Lipschitz constant for* $g_{\hat{\mathcal{E}}_i}$*'s derivatives. The output* $\tilde{\mathbf{Z}}^{(i)}$ *will converge to* $\mathbf{Z}^*$ *linearly in expectation as follows:*

$$\mathbb{E}\left\|\tilde{\mathbf{Z}}^{(i)} - \mathbf{Z}^*\right\|_F^2 \leq \rho^i \left\|\tilde{\mathbf{Z}}^{(0)} - \mathbf{Z}^*\right\|_F^2.$$

As the above proposition shows, our USP-VR solver enjoys a linear convergent rate like the vanilla deterministic solvers. Therefore, our USP-VR solver can achieve a more accurate solution compared

Algorithm 1 USP-VR solver's procedure for the equilibrium state.

```
Input: Input graph Laplacian $\tilde{\mathbf{A}}$, input injection $\mathbf{B}$, initial state $\tilde{\mathbf{z}}_0$, step size $\eta$, maximum iteration
    number $m_1$, $i = 1$, maximum inner iteration number $m_2$, $\mathbf{W}$,$\sigma$ depend on the choices of different
    graph equilibrium models.
Output: output $\tilde{\mathbf{Z}}$.
 1: Randomly split the whole edge set $\mathcal{E}$ into $m$ subsets $\{\hat{\mathcal{E}}_1, \cdots, \hat{\mathcal{E}}_m\}$ and generate the subgraph's
    laplacian matrix. $\{\hat{\mathbf{A}}_{\hat{\mathcal{E}}_1}, \cdots, \hat{\mathbf{A}}_{\hat{\mathcal{E}}_m}\}$.
 2: while $i \leq m_1$ and $\tilde{\mathbf{Z}}^{(i)}$ not satisfies stop condition do
 3:     $\tilde{\mathbf{Z}} = \tilde{\mathbf{Z}}^{(i)}$.
 4:     $\tilde{\mathbf{V}} = \tilde{\mathbf{Z}} + \tilde{\mathbf{A}}\tilde{\mathbf{Z}}\mathbf{W}\mathbf{W}^\top - \mathbf{B}$.
 5:     $\mathbf{Z}^{(0)} = \tilde{\mathbf{Z}}$.
 6:     for $k = 1$ to $m_2$ do
 7:         Randomly pick $s_k \in \{1, \cdots, m\}$ with the same probability.
 8:         $\hat{\mathbf{V}}^{(k)} = (\mathbf{Z}^{(k-1)} - \tilde{\mathbf{Z}}) + m\hat{\mathbf{A}}_{\hat{\mathcal{E}}_{s_k}}(\mathbf{Z}^{(k-1)} - \tilde{\mathbf{Z}})\mathbf{W}\mathbf{W}^\top + \tilde{\mathbf{V}}$.
 9:         $\mathbf{Z}^{(k)} = \sigma\left(\mathbf{Z}^{(k-1)} - \eta\hat{\mathbf{V}}^{(k)}\mathbf{W}\right)$.
10:     end for
11:     $\tilde{\mathbf{Z}}^{(i+1)} = \frac{1}{m_2}\sum_{k=1}^{m_2}\mathbf{Z}^{(k)}$.
12:     $i = i + 1$
13: end while
```

with the USP-VR solver with limited propagation times in the forward procedure. Moreover, our USP-VR solver is still much faster than the original deterministic solvers since we only use the full graph for every m_2 iteration.

3.4 Backpropagation when using our solvers

Assuming all the graph equilibrium models have achieved the equilibrium $\mathbf{Z}^{\mathrm{f}}$ during its iterative forward pass, we use the one-step gradient by feeding $\mathbf{Z}^{\mathrm{f}}$ to one deterministic iteration update (Eqn (1)). Then we can obtain the output $\mathbf{Z}^{\mathrm{o}}$ to calculate loss inspired from other models (Geng et al., 2021; Guo, 2013; Fung et al., 2022). Since we can regard the whole structure as only propagating once from a near-optimal initialization point, we can only backward such iterations' gradient. The gradient for our methods can be written as follows:

$$\begin{aligned}\frac{\partial\mathcal{L}(\mathbf{Z}^{\mathrm{o}})}{\partial\mathbf{W}} &= -(\gamma\tilde{\mathbf{A}}\mathbf{Z}^{\mathrm{f}})^\top\left[\frac{\partial\mathcal{L}(\mathbf{Z}^{\mathrm{o}})}{\partial\mathbf{Z}^{\mathrm{o}}}\odot\sigma'\left(-\gamma\tilde{\mathbf{A}}\mathbf{Z}^{\mathrm{f}}\mathbf{W}\mathbf{W}^\top + \gamma\mathbf{B} + (\mathbf{1}-\gamma)\mathbf{Z}^{\mathrm{f}}\right)\right]\mathbf{W} \\ &\quad - (\gamma\tilde{\mathbf{A}}\mathbf{Z}^{\mathrm{f}}\mathbf{W})^\top\left[\frac{\partial\mathcal{L}(\mathbf{Z}^{\mathrm{o}})}{\partial\mathbf{Z}^{\mathrm{o}}}\odot\sigma'\left(-\tilde{\mathbf{A}}\mathbf{Z}^{\mathrm{f}}\mathbf{W}\mathbf{W}^\top + \gamma\mathbf{B} + (\mathbf{1}-\gamma)\mathbf{Z}^{\mathrm{f}}\right)\right],\end{aligned} \tag{9}$$

where γ depend on the setting of different models. Then we can use such gradient to train the graph equilibrium models and we adopt it in all our experiments.

4 Experiments

In this section, we demonstrate the efficiency of our proposed USP and USP-VR solver compared with the traditional implicit models on large graph datasets with better performance on both node classification and graph classification tasks. Specifically, we test IGNN using our solvers against the vanilla IGNNs and other explicit graph models or methods on 4 popular node classification datasets (Flickr, Reddit, Yelp, PPI (Zitnik & Leskovec, 2017)) and 2 graph classification datasets. Apart from the IGNN, we also compare IRLS-Base with its original solver and ours on OGBN (Hu et al., 2020) datasets. These datasets are challenging for not only graph equilibrium models but also explicit models due to their sizes. We conduct the experiments on PyTorch (Paszke et al., 2019) with Torch-Geometric (Fey & Lenssen, 2019) and DGL (Wang et al., 2019). All the experiments are finished on an RTX-3090 GPU. The hyper-parameter settings and other details for the experiments are listed in the Appendix.

Table 1: Data Statistic ("m" stands for multi-label classification, and "s" for single-label).

Dataset	Nodes	Edges	Degree	Classes	Training/Validation/Test
Flickr	$89,250$	$899,756$	10	$7(s)$	$0.50/0.25/0.25$
Reddit	$232,965$	$114,615,892$	50	$41(s)$	$0.66/0.10/0.24$
Yelp	$716,847$	$13,954,819$	10	$100(m)$	$0.75/0.10/0.15$
PPI (Multi Graph)	$14,755$	$225,270$	14	$121(m)$	$0.79/0.11/0.10$
OGBN-Arxiv	$169,343$	$1,166,243$	7	$40(s)$	$0.54/0.18/0.18$
OGBN-Products	$2,449,029$	$61,859,140$	25	$47(s)$	$0.90/0.02/0.08$

4.1 Numerical Evaluation

We first conduct an experiment to explore whether our proposed method can converge to equilibrium as our analysis shows. As we cannot obtain the closed-form solution for Problem 2, we use the gradient of $g(\mathbf{Z})$ in Eqn (3) as the evaluation of the convergence. This is because $\nabla_{\mathbf{Z}} g(\mathbf{Z}) = 0$ also demonstrates $\mathbf{Z}$ satisfies the first-order condition of Problem 2 if $\mathbf{Z}$ is obtained by our methods.

In this part, we use Reddit to draw the convergence curve for one layer IGNN, IGNN+USP, and IGNN+USP-VR layer with 128 output feature size for each node. We set $m = 20$ for our proposed solvers and $m_2 = 10$ for IGNN+USP-VR. Then we draw the convergence curve of the relative gradient concerning the equivalent subgraph aggregation times (Each IGNN's iteration through the full graph is equivalent to m times subgraph aggregation) as Figure 1 shows.

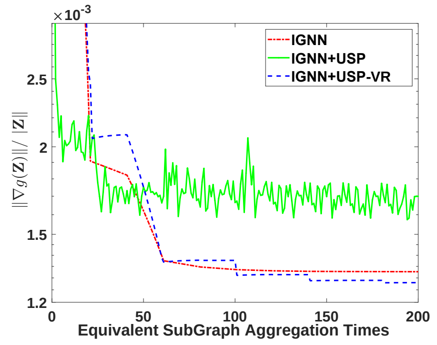


Figure 1: The convergence curve of each models' equilibrium state on Reddit Dataset.

The convergence curve is consistent with our analysis: IGNN+USP can quickly converge compared with IGNN and IGNN+USP-VR in the beginning while IGNN+USP-VR will finally converge to a better solution compared with the other two models.

4.2 Node Classification

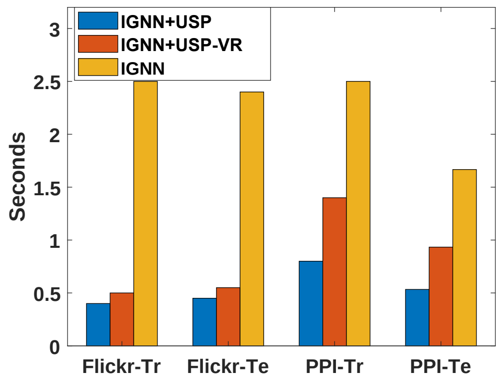
Comparison on popular large-scale graphs. In this section, we first evaluate the performance of our methods on popular large-scale benchmark node classification graph datasets with the base model IGNN. The statistics of the four datasets are listed in Table 1. We compared the performance with IGNNs and other explicit graph methods. The explicit methods to accelerate the graph training can be categorized into two classes: Sub-Graph Training (like ClusterGCN and GraphSAINT with GraphSAGE) and Full-Graph Training (includes FastGCN and others). The Sub-Graph Training methods feed the sampled subgraphs to their graph models in each training iteration while the Full-Graph Training methods feed the full graph into their models but only use the sampled nodes for each aggregation layer, which is similar to our proposed solvers. As for IGNN with its vanilla deterministic solver or our stochastic solvers, we use the full graph for the training. All models on all datasets have uniform hidden dimensions of 256, $m = 20$ for our stochastic solvers, and other training details can be found in the Appendix. All the results are listed in Table 2.

From the table, one can see that our methods can outperform other explicit models with notable advantages on most datasets, which means that our USP solvers can also capture long-range information as the deterministic implicit graph model IGNN. Furthermore, one can see that IGNN with our

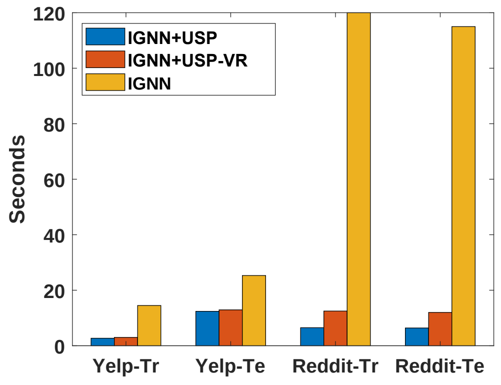
Table 2: Comparison of test set Test Accuracy/F1-micro score with different methods. "OOT" here denotes that the time cost is more than $10\times$ longer than IGNN+USP.

	Model	Flickr	Reddit	Yelp	PPI
Explicit	GCN (Kipf & Welling, 2016)	$49.2 \pm 0.3\%$	$93.3 \pm 0.1\%$	$37.8 \pm 0.1\%$	$51.2 \pm 0.3\%$
	GraphSAGE (Hamilton et al., 2017)	$50.1 \pm 1.3\%$	$95.3 \pm 0.1\%$	$63.4 \pm 0.6\%$	$63.4 \pm 0.4\%$
	FastGCN (Chen et al., 2018)	$50.1 \pm 1.3\%$	$95.3 \pm 0.1\%$	$63.4 \pm 0.6\%$	$51.3 \pm 3.2\%$
	ASGCN (Huang et al., 2018)	$50.1 \pm 1.3\%$	$95.3 \pm 0.1\%$	$63.4 \pm 0.6\%$	$68.7 \pm 1.2\%$
	ClusterGCN (Chiang et al., 2019)	$48.1 \pm 0.5\%$	$95.4 \pm 0.1\%$	$60.9 \pm 0.5\%$	$87.3 \pm 0.4\%$
	GraphSAINT (Zeng et al., 2019)	$51.5 \pm 0.1\%$	$96.7 \pm 0.1\%$	$64.5 \pm 0.3\%$	$98.0 \pm 0.2\%$
Implicit	IGNN (Gu et al., 2020)	$53.0 \pm 0.2\%$	OOT	$65.8 \pm 0.2\%$	$97.8 \pm 0.1\%$
	IGNN+USP	$54.1 \pm 0.2\%$	$96.7 \pm 0.3\%$	$\mathbf{66.2 \pm 0.2\%}$	$98.3 \pm 0.2\%$
	IGNN+USP-VR	$\mathbf{54.3 \pm 0.1\%}$	$\mathbf{96.8 \pm 0.2\%}$	$66.1 \pm 0.2\%$	$\mathbf{98.5 \pm 0.2\%}$

proposed stochastic solvers also shows advantages against the vanilla IGNN on node classification tasks. Such a phenomenon may be caused by the randomness induced by our stochastic produce improves the generalization ability of the implicit graph models. Moreover, the USP-VR solver is slightly better than USP solver because the variance reduction version can obtain more accurate equilibrium states.



(a) Flickr and PPI.



(b) Yelp and Reddit.

Figure 2: Comparison of the training time per epoch and test time through the full graph of IGNN and IGNN+USP for different Node Classification Dataset. x-Tr means the training time for model x and x-Te means the test time for model x through the full graph.

In addition to evaluating the prediction performance, we also draw the testing and training time for different models in Figure 2 to demonstrate the efficiency of our proposed solvers. From the figures, one can see that IGNN+USPs are significantly faster than IGNNs especially when the graph is huge, like our IGNN+USP are more than $10\times$ faster on Reddit (contains over 100M edges) during training time and test time. And our IGNN+USP are over $3\times$ faster on PPI and more than $5\times$ faster on Flickr. As for Yelp, our IGNN+USP are more than $6\times$ faster during training and $2\times$ during testing. Our USP-VR solver is slightly slower than our USP solver since they need additional full graph aggregation in the forward procedure.

Comparison on OGBN dataset. Apart from the above datasets, we also conduct experiments for IRLS with their vanilla deterministic solver and our proposed USP solvers listed as listed in Table 3. From the table, one can see that our USP solvers can still return better results on OGBN datasets. Furthermore, our USP solvers can also make the IRLS and IGNN more efficient as listed in the Tables. Due to the above results, our proposed USP solvers are much more efficient than the original deterministic solvers for graph equilibrium models. From the above results, we can conclude that our proposed solvers are much more efficient than IGNN's and IRLS's original solvers especially on large graphs with state-of-the-art performance on the node classification tasks.

4.3 Graph Classification

Aside from the node classification, we also conduct experiments on graph classification datasets to see whether our methods can show consistent advantages on graph classification tasks. We use D&D (Dobson & Doig, 2003) (Bioinformatics Graphs) and COLLAB (Yanardag & Vishwanathan,

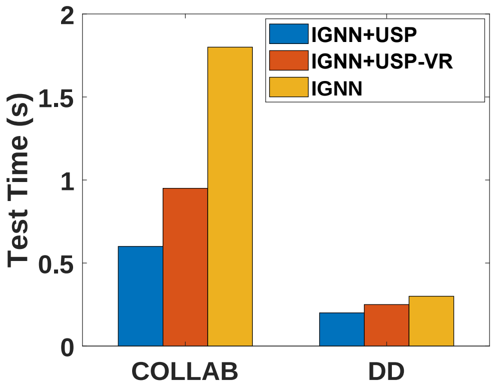
	OGBN-Arxiv		OGBN-Products	
	Accuracy	Speed Up	Accuracy	Speed Up
IRLS	71.1%	1.0×	68.8%	1.0×
IRLS+USP	71.6%	**2.0**×	71.4%	**3.0**×
IRLS+USP-VR	**71.8**%	1.5×	**71.5**%	2.2×
IGNN	70.4%	1×	69.7%	1.0×
IGNN+USP	72.7%	**3.5**×	73.4%	**5**×
IGNN+USP-VR	**72.9**%	2.2×	**73.5**%	3×

Table 3: The empirical results for IRLS and IGNN with their vanilla solver and our proposed USP solvers on OGBN datasets.

2015) (Social Graphs) for our experiments and use CGS and IGNNs as baselines. Training details can be found in the Appendix. The classification accuracy is listed in Table 3a and the test for IGNN and our methods are listed in Figure 3b. From the results, our proposed solvers can also perform well on the graph classification tasks.

dataset	D&D	COLLAB
# Graphs	1178	5000
Avg # Edges	716	2458
CGS-1	$76.75 \pm 2.9\%$	$77.4 \pm 3.5\%$
CGS-4	OOM	$77.5 \pm 3.1\%$
IGNN	$76.5 \pm 8.1\%$	$\mathbf{78.5 \pm 1.6\%}$
IGNN+USP	$78.1 \pm 6.2\%$	$78.4 \pm 1.2\%$
IGNN+USP-VR	$\mathbf{81.8 \pm 2.1\%}$	$78.1 \pm 1.5\%$

(a) Graph classification accuracy.



(b) Test time for IGNN and IGNN with our proposed solvers.

Figure 3: Graph classification accuracy and test time. Accuracies are averaged (and std are computed) on the outer 10 folds. CGS-k denotes the CGS model with k heads. "OOM" denotes out-of-memory.

IGNN with our proposed solvers is also faster than its origin in the experiment, $3\times$ on COLLAB and $1.5\times$ on D&D as the figure shows. We notice that our advantages of efficiency are less significant than using our methods in the node classification tasks. The reason is that the graphs are much smaller in the graph classification task and the time cost of graph aggregations is not too large compared with others like linear layers' propagation or data processing. Furthermore, we can conclude that our stochastic solvers can perform more efficiently as the graph scale goes larger as our experiments show. Because the time cost of graph aggregation can gradually dominate the total cost of the implicit models as the graph gets larger.

5 Conclusions

In our work, we first propose two stochastic solvers for the graph equilibrium models from the view of their relationship to the graph denoising optimization problem. Our proposed solvers can be applied to different graph equilibrium models such as IGNN and IRLS. Since our proposed solvers only need to propagate sub-graphs in each aggregation, our methods are much faster than the original fixed point iterative or gradient descent solvers used for implicit graph models and unfolded graph models, especially on large and dense graphs. Furthermore, we also theoretically prove that our solvers are unbiased and can finally output the same equilibrium state as the vanilla equilibrium models. The empirical results also demonstrate the advantages of different models with our proposed solvers for large-scale graphs.

REFERENCES

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Multiscale deep equilibrium models. *Advances in Neural Information Processing Systems*, 33:5238–5250, 2020.

Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International Conference on Machine Learning*, pp. 1407–1418. PMLR, 2021.

Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction. *arXiv preprint arXiv:1710.10568*, 2017.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

Tianlong Chen, Kaixiong Zhou, Keyu Duan, Wenqing Zheng, Peihao Wang, Xia Hu, and Zhangyang Wang. Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study. *CoRR*, abs/2108.10521, 2021. URL `https://arxiv.org/abs/2108.10521`.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery &amp Data Mining*. ACM, jul 2019. doi: 10.1145/3292500.3330925. URL `https://doi.org/10.1145%2F3292500.3330925`.

Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

Laurent El Ghaoui, Fangda Gu, Bertrand Travacca, Armin Askari, and Alicia Tsai. Implicit deep learning. *SIAM Journal on Mathematics of Data Science*, 3(3):930–958, 2021.

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

Samy Wu Fung, Howard Heaton, Qiuwei Li, Daniel McKenzie, Stanley Osher, and Wotao Yin. Jfb: Jacobian-free backpropagation for implicit networks. 2022.

Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 24247–24260, 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/cb8da6767461f2812ae4290eac7cbc42-Abstract.html`.

Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. Implicit graph neural networks. *Advances in Neural Information Processing Systems*, 33:11984–11995, 2020.

Jiang Guo. Backpropagation through time. *Unpubl. ms., Harbin Institute of Technology*, 40:1–6, 2013.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017. URL `http://arxiv.org/abs/1706.02216`.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pp. 639–648, 2020.

Shell Xu Hu, Sergey Zagoruyko, and Nikos Komodakis. Exploring weight symmetry in deep neural networks. *Computer Vision and Image Understanding*, 187:102786, 2019a.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019b.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph representation learning. *Advances in neural information processing systems*, 31, 2018.

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*, pp. 66–74, 2020.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Mingjie Li, Yisen Wang, Xingyu Xie, and Zhouchen Lin. Optimization inspired multi-branch equilibrium models. In *International Conference on Learning Representations*, 2021.

Juncheng Liu, Kenji Kawaguchi, Bryan Hooi, Yiwei Wang, and Xiaokui Xiao. Eignn: Efficient infinite-depth graph neural networks. *Advances in Neural Information Processing Systems*, 34, 2021a.

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

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

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*, pp. 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*, pp. 1202–1211, 2021.

Maya Mincheva and Marc R Roussel. Graph-theoretic methods for the analysis of chemical and biochemical networks. ii. oscillations in networks with delays. *Journal of mathematical biology*, 55(1):87–104, 2007.

S Deepak Narayanan, Aditya Sinha, Prateek Jain, Purushottam Kar, and Sundararajan Sellamanickam. Iglu: Efficient gcn training via lazy updates. *arXiv preprint arXiv:2109.13995*, 2021.

Junyoung Park, Jinhyun Choo, and Jinkyoo Park. Convergent graph solvers. *arXiv preprint arXiv:2106.01680*, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1150–1160, 2020.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Hkx1qkrKPr`.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Fangping Wan, Lixiang Hong, An Xiao, Tao Jiang, and Jianyang Zeng. Neodti: neural integration of neighbor information from a heterogeneous network for discovering new drug–target interactions. *Bioinformatics*, 35(1):104–111, 2019.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. Dissecting the diffusion process in linear graph convolutional networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

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.

Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction, 2014. URL `https://arxiv.org/abs/1403.4699`.

Bingbing Xu, Huawei Shen, Qi Cao, Keting Cen, and Xueqi Cheng. Graph convolutional networks using heat kernel for semi-supervised learning. *arXiv preprint arXiv:2007.16002*, 2020a.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pp. 5453–5462. PMLR, 2018.

Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. Grid-gcn for fast and scalable point cloud learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5661–5670, 2020b.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374, 2015.

Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek F. Abdelzaher. Revisiting ”over-smoothing” in deep gcns. *CoRR*, abs/2003.13663, 2020. URL `https://arxiv.org/abs/2003.13663`.

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 *International Conference on Machine Learning*, pp. 11773–11783. PMLR, 2021a.

Yongyi Yang, Yangkun Wang, Zengfeng Huang, and David Wipf. Implicit vs unfolded graph neural networks. *CoRR*, abs/2111.06592, 2021b. URL `https://arxiv.org/abs/2111.06592`.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 430–438, 2020.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graph-saint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

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*, 2020a.

Shijie Zhang, Hongzhi Yin, Tong Chen, Quoc Viet Nguyen Hung, Zi Huang, and Lizhen Cui. Gcn-based user representation learning for unifying robust recommendation and fraudster detection. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pp. 689–698, 2020b.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. *CoRR*, abs/1909.12223, 2019. URL `http://arxiv.org/abs/1909.12223`.

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.

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*, pp. 1215–1226, 2021.

Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.

Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. Layer-dependent importance sampling for training deep and large graph convolutional networks. *Advances in neural information processing systems*, 32, 2019.

A Proofs for Proposition 1

Before the USP solver's proof, we first provide a lemma for the following problem.

Lemma 1. *(Xiao & Zhang, 2014) Let $G(\mathbf{X}) = g(\mathbf{X}) + f(\mathbf{X})$, where $\nabla g(\mathbf{X})$ is Lipschitz continuous with parameter L and $g(\mathbf{X})$ has strong convexity parameters μ. For any $\mathbf{X} \in \operatorname{dom}(f)$ and arbitary $\mathbf{V}$ in the real space, define:*

$$\begin{aligned}\mathbf{X}^{+} &= \operatorname{prox}_{\eta f}(\mathbf{X} - \eta\mathbf{V})\\ \mathbf{G} &= \frac{1}{\eta}(\mathbf{X} - \mathbf{X}^{+})\\ \Delta &= \mathbf{V} - \nabla g(\mathbf{X}),\end{aligned}$$

where η is a step size satisfying $0 < \eta \leq 1/L$. Then we have for any $\mathbf{Y}$ in the real space,

$$G(\mathbf{Y}) \geq G(\mathbf{X}^{+}) + \operatorname{Tr}(G^{\top}(\mathbf{Y} - \mathbf{X})) + \frac{\eta}{2}\|\mathbf{G}\|_F^2 + \frac{\mu}{2}\|\mathbf{Y} - \mathbf{X}\|_F^2 + \operatorname{Tr}(\Delta^{\top}(\mathbf{X}^{+} - \mathbf{Y}))$$

Proof. It is easy to prove the gradients are unbiased, since $g_{\hat{\mathcal{E}}_k}(\mathbf{Z}) = \mathbf{Z} - \mathbf{B} + \hat{\mathbf{A}}_{\hat{\mathcal{E}}_k}\mathbf{Z}$ and $\mathbb{E}\hat{\mathbf{A}}_{\hat{\mathcal{E}}_k} = \tilde{\mathbf{A}}$. Thereby, the gradients are unbiased:

$$\mathbb{E}\nabla g_{\hat{\mathcal{E}}_k}(\mathbf{Z}) = \mathbf{Z} - \mathbf{B} + \mathbb{E}\hat{\mathbf{A}}_{\hat{\mathcal{E}}_k}\mathbf{Z} = \mathbf{Z} - \mathbf{B} + \tilde{\mathbf{A}}\mathbf{Z} = \nabla g(\mathbf{Z})$$

In order to prove the convergence, we first define the gradient mapping $\mathbf{G}^{(t)} = (\mathbf{Z}^{(t)} - \mathbf{Z}^{(t-1)})/\eta_t$, then we can get:

$$\begin{aligned}\left\|\mathbf{Z}^{(t)} - \mathbf{Z}^{*}\right\|_F^2 &= \|\mathbf{Z}^{(t-1)} - \eta_t\mathbf{G}^{(t)} - \mathbf{Z}^{*}\|_F^2\\ &= \left\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^{*}\right\|_F^2 - 2\eta_t\operatorname{Tr}\left(\mathbf{G}^{(t)\top}(\mathbf{Z}^{(t-1)} - \mathbf{Z}^{*})\right) + \eta_t^2\|\mathbf{G}^{(t)}\|_F^2.\end{aligned}$$

Then we can apply Lemma 1 and we can get:

$$\begin{aligned}-\operatorname{Tr}\left(\mathbf{G}^{(t)\top}(\mathbf{Z}^{(t-1)} - \mathbf{Z}^{*})\right) + \frac{\eta_t}{2}\|\mathbf{G}^{(t)}\|_F^2 \leq G(\mathbf{Z}^{*}) - G(\mathbf{Z}^{(t)}) - \frac{1}{2}\left\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^{*}\right\|_F^2\\ + \operatorname{Tr}\left(\Delta^{(t)\top}(\mathbf{Z}^{(t)} - \mathbf{Z}^{*})\right),\end{aligned}$$

where $\Delta^{(t)} = \nabla g_{\hat{\mathcal{E}}_{i_k}}(\mathbf{Z}^{(t-1)}) - \nabla g(\mathbf{Z}^{(t-1)})$. Combining the above equations and rearranging them, we have:

$$\|\mathbf{Z}^{(t)} - \mathbf{Z}^{*}\|_F^2 + 2\eta_t\left(g(\mathbf{Z}^{(t)}) - g(\mathbf{Z}^{*}) + \operatorname{Tr}\left(\Delta^{(t)\top}(\mathbf{Z}^{(t)} - \mathbf{Z}^{*})\right)\right) \leq (1 - \eta_t)\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^{*}\|_F^2.$$

Since $g(\mathbf{Z}^{(t)}) - g(\mathbf{Z}^{*})$ is non-negative with i_k is uniformly chosen from $\{1, .., m\}$ and $\eta_t = 1/t$ we have:

$$\|\mathbf{Z}^{(t)} - \mathbf{Z}^{*}\|_F^2 + \frac{2}{t}\operatorname{Tr}\left(\Delta^{(t)\top}(\mathbf{Z}^{(t)} - \mathbf{Z}^{*})\right) \leq \frac{t-1}{t}\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^{*}\|_F^2$$

To bound $\frac{2}{t}\operatorname{Tr}\left(\Delta^{(t)\top}(\mathbf{Z}^{(t-1)} - \mathbf{Z}^{*})\right)$, we set an auxiliary state $\bar{\mathbf{Z}}^{(t)}$ updated with the proximal full gradient update:

$$\bar{\mathbf{Z}}^{(t)} = \sigma(\mathbf{Z}^{(t)} - 1/t * \nabla g(\mathbf{Z}^{(t-1)})).$$

Then we can get:

$$\begin{aligned}-\frac{2}{t}\operatorname{Tr}\left(\Delta^{(t)\top}(\mathbf{Z}^{(t)} - \mathbf{Z}^{*})\right) &= -\frac{2}{t}\operatorname{Tr}\left(\Delta^{(t)\top}(\mathbf{Z}^{(t)} - \bar{\mathbf{Z}}^{(t)})\right) - \frac{2}{t}\operatorname{Tr}\left(\Delta^{(t)\top}(\bar{\mathbf{Z}}^{(t)} - \mathbf{Z}^{*})\right)\\ &\leq \frac{2}{t^2}\left\|\Delta^{(t)}\right\|_2^2 - \frac{2}{t}\operatorname{Tr}\left(\Delta^{(t)\top}(\bar{\mathbf{Z}}^{(t)} - \mathbf{Z}^{*})\right)\end{aligned}$$

Thereby,

$$\|\mathbf{Z}^{(t)} - \mathbf{Z}^{*}\|_F^2 \leq \frac{t-1}{t}\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^{*}\|_F^2 + \frac{2}{t^2}\|\Delta^{(t)}\|_F^2 - \frac{2}{t}\operatorname{Tr}\left(\Delta^{(t)\top}(\bar{\mathbf{Z}}^{(t)} - \mathbf{Z}^{*})\right).$$

Take expectations on i_k, we can get that $\mathbb{E}\mathrm{Tr}\left(\Delta^{(t)\top}(\bar{\mathbf{Z}}^{(t)} - \mathbf{Z}^*)\right) = 0$ since our gradient is unbiased. Set η_t as a constant η, the above formulation becomes:

$$\mathbb{E}\|\mathbf{Z}^{(t)} - \mathbf{Z}^*\|_F^2 \leq \frac{t-1}{t}\mathbb{E}\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^*\|_F^2 + \frac{2}{t^2}\mathbb{E}\|\Delta^{(t)}\|_F^2.$$

Since $\mathbb{E}\left\|\mathbf{X} - \mathbb{E}\mathbf{X}\right\|_2^F \leq \mathbb{E}\mathbf{X}^2 - (\mathbb{E}\mathbf{X})^2$ and derivatives of $g_{\hat{\mathcal{E}}_{i_k}}$ are $L_{\max}$-Lipschitz continuous, then,

$$\mathbb{E}\|\mathbf{Z}^{(t)} - \mathbf{Z}^*\|_F^2 \leq \frac{t-1}{t}\mathbb{E}\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^*\|_F^2 + \frac{2L_{\max}^2}{t^2}\mathbb{E}\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^*\|_F^2.$$

Since the order of the second term is higher, we can neglect when considering the convergent speed. Thereby, there exist a C and T, for $t > T$:

$$\mathbb{E}\|\mathbf{Z}^{(t)} - \mathbf{Z}^*\|_F^2 \leq C\frac{t-1}{t}\mathbb{E}\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^*\|_F^2.$$

Then $\mathbb{E}\|\mathbf{Z}^{(t)} - \mathbf{Z}^*\|_F^2 = O(1/t)$. The proof is complete □

B PROOFS FOR PROPOSITION 2

Proof. Like the proofs in Proposition 1, First, we define the gradient mapping $\mathbf{G}^{(t)} = (\mathbf{Z}^{(t)} - \mathbf{Z}^{(t-1)})/\eta_t$, then we can get:

$$\begin{aligned}\left\|\mathbf{Z}^{(t)} - \mathbf{Z}^*\right\|_F^2 &= \|\mathbf{Z}^{(t-1)} - \eta_t\mathbf{G}^{(t)} - \mathbf{Z}^*\|_F^2 \\ &= \left\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^*\right\|_F^2 - 2\eta_t\mathrm{Tr}\left(\mathbf{G}^{(t)\top}(\mathbf{Z}^{(t-1)} - \mathbf{Z}^*)\right) + \eta_t^2\|\mathbf{G}^{(t)}\|_F^2.\end{aligned}$$

Then we can apply Lemma 1 and we can get:

$$\begin{aligned}-\mathrm{Tr}\left(\mathbf{G}^{(t)\top}(\mathbf{Z}^{(t-1)} - \mathbf{Z}^*)\right) + \frac{\eta_t}{2}\|\mathbf{G}^{(t)}\|_F^2 \leq G(\mathbf{Z}^*) - G(\mathbf{Z}^{(t)}) - \frac{1}{2}\left\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^*\right\|_F^2 \\ + \mathrm{Tr}\left(\Delta^{(t)\top}(\mathbf{Z}^{(t)} - \mathbf{Z}^*)\right),\end{aligned}$$

where $\Delta^{(t)} = \hat{\mathbf{V}}_k - \nabla g(\mathbf{Z}^{(t-1)}) = \nabla g_{\hat{\mathcal{E}}_k}(\mathbf{Z}^{(t-1)}) - g_{\hat{\mathcal{E}}_k}(\tilde{\mathbf{Z}}) + g(\tilde{\mathbf{Z}}) - \nabla g(\mathbf{Z}^{(t-1)})$ as Eqn (8)'s definition. Combining the above equations and rearranging them, we have:

$$\|\mathbf{Z}^{(t)} - \mathbf{Z}^*\|_F^2 + 2\eta_t\left(g(\mathbf{Z}^{(t)}) - g(\mathbf{Z}^*) + \mathrm{Tr}\left(\Delta^{(t)\top}(\mathbf{Z}^{(t)} - \mathbf{Z}^*)\right)\right) \leq (1-\eta_t)\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^*\|_F^2.$$

Rearranging the above equation, we can get:

$$\|\mathbf{Z}^{(t)} - \mathbf{Z}^*\|_F^2 \leq \|\mathbf{Z}^{(t-1)} - \mathbf{Z}^*\|_F^2 - 2\eta_t\left(g(\mathbf{Z}^{(t)}) - g(\mathbf{Z}^*) + \mathrm{Tr}\left(\Delta^{(t)\top}(\mathbf{Z}^{(t)} - \mathbf{Z}^*)\right)\right)$$

Like the former proof, we set an auxiliary state $\bar{\mathbf{Z}}^{(t)}$ updated with the proximal full gradient update:

$$\bar{\mathbf{Z}}^{(t)} = \sigma(\mathbf{Z}^{(t)} - 1/t * \nabla g(\mathbf{Z}^{(t-1)})).$$

Then we can get:

$$-2\eta_t\mathrm{Tr}\left(\Delta^{(t)\top}(\mathbf{Z}^{(t)} - \mathbf{Z}^*)\right) \leq 2\eta_t^2\left\|\Delta^{(t)}\right\|_2^2 - 2\eta_t\mathrm{Tr}\left(\Delta^{(t)\top}(\bar{\mathbf{Z}}^{(t)} - \mathbf{Z}^*)\right)$$

Thereby,

$$\|\mathbf{Z}^{(t)}-\mathbf{Z}^*\|_F^2 \leq \|\mathbf{Z}^{(t-1)}-\mathbf{Z}^*\|_F^2+2\eta_t^2\|\Delta^{(t)}\|_F^2-2\eta_t\mathrm{Tr}\left(\Delta^{(t)\top}(\bar{\mathbf{Z}}^{(t)} - \mathbf{Z}^*)\right)-2\eta_t(g(\mathbf{Z}^{(t)})-g(\mathbf{Z}^*)).$$

Take expectations on i_k, we can get that $\mathbb{E}\mathrm{Tr}\left(\Delta^{(t)\top}(\bar{\mathbf{Z}}^{(t)} - \mathbf{Z}^*)\right) = 0$ since our gradient is unbiased. Then, the above formulation becomes:

$$\mathbb{E}\|\mathbf{Z}^{(t)} - \mathbf{Z}^*\|_F^2 \leq \mathbb{E}\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^*\|_F^2 + 2\eta^2\mathbb{E}\|\Delta^{(t)}\|_F^2 - 2\eta(\mathbb{E}g(\mathbf{Z}^{(t)}) - g(\mathbf{Z}^*)).$$

Then we need to bound $\mathbb{E}\|\Delta^{(t)}\|_F^2$, we need the following proposition to do so.

Proposition 3. *For our defined $G(\mathbf{Z})$ with the derivatives of $g(\mathbf{Z})$ and $g_{\hat{\mathcal{E}}_k}(\mathbf{Z})$ are all $L_{\max}$-Lipschitz continuous as we defined in Propostion 2. Then the following equations are satisfied:*

$$\frac{1}{m}\sum_{i=1}^{m}\|\nabla g_{\hat{\mathcal{E}}_i}(\mathbf{Z}) - \nabla g(\mathbf{Z})\|_F^2 \leq 2L_{\max}\left(G(\mathbf{Z}) - G(\mathbf{Z}^*)\right)$$

$$\mathbb{E}\left\|\hat{\mathbf{V}}^{(t)} - \nabla g(\mathbf{Z}^{(t-1)})\right\|_F^2 \leq 4L_{\max}\left(G(\mathbf{Z}^{(t-1)}) - G(\mathbf{Z}^*) + G(\tilde{\mathbf{Z}}) - G(\mathbf{Z}^*)\right),$$

$\mathbf{Z}^*$ is the equilibrium state and $\hat{\mathbf{V}}^{(t)}$ and $\tilde{\mathbf{Z}}$ follows the definition in Algorithm 1.

The proofs for the above proposition is listed as follows:

Proof. For the first equation, we first consider $\phi_i(\mathbf{Z})$ for any $i \in 1, ...m$:

$$\phi_i(\mathbf{Z}) = g_{\hat{\mathcal{E}}_i}(\mathbf{Z}) - g_{\hat{\mathcal{E}}_i}(\mathbf{Z}^*) - \mathrm{Tr}(\nabla g_{\hat{\mathcal{E}}_i}(\mathbf{Z}^*)^\top(\mathbf{Z} - \mathbf{Z}^*))$$

Because $\nabla\phi_i(\mathbf{Z}^*) = 0$, hence $\min_{\mathbf{Z}}\phi_i(\mathbf{Z}) = \phi_i(\mathbf{Z}^*) = 0$. Since$\nabla\phi_i$ is Lipschitz continuous with $L_{\max}$, we have:

$$\frac{1}{2L_{\max}}\|\nabla\phi_i(\mathbf{Z})\|_F^2 \leq \phi_i(\mathbf{Z}) - \min_{\mathbf{X}}\phi_i(\mathbf{X}) = \phi_i(\mathbf{Z}).$$

Then,

$$\left\|g_{\hat{\mathcal{E}}_i}(\mathbf{Z}) - g_{\hat{\mathcal{E}}_i}(\mathbf{Z}^*)\right\|_F^2 \leq 2L_{\max}\left(g_{\hat{\mathcal{E}}_i}(\mathbf{Z}) - g_{\hat{\mathcal{E}}_i}(\mathbf{Z}^*) - \mathrm{Tr}(\nabla g_{\hat{\mathcal{E}}_i}^\top(\mathbf{Z}^*)(\mathbf{Z} - \mathbf{Z}^*))\right).$$

Summing the above equation over $i = 1, .., m$, we have:

$$\frac{1}{m}\sum_{i=1}^{m}\|\nabla g_{\hat{\mathcal{E}}_i}(\mathbf{Z}) - \nabla g(\mathbf{Z})\|_F^2 \leq 2L_{\max}\left(g(\mathbf{Z}) - g(\mathbf{Z}^*) - \mathrm{Tr}(\nabla g^\top(\mathbf{Z}^*)(\mathbf{Z} - \mathbf{Z}^*)\right)$$

Since $-\nabla g(\mathbf{Z}^*) \in \partial f(\mathbf{Z}^*)$ due to the optimality, we have:

$$\begin{aligned} g\left(\mathbf{Z}\right) - g(\mathbf{Z}^*) - \mathrm{Tr}(\nabla g^\top(\mathbf{Z}^*)(\mathbf{Z} - \mathbf{Z}^*)\big) &= g(\mathbf{Z}) - g(\mathbf{Z}^*) + \mathrm{Tr}\left(\partial f^\top(\mathbf{Z}^*)(\mathbf{Z} - \mathbf{Z}^*)\right) \\ &\leq g(\mathbf{Z}) - g(\mathbf{Z}^*) + f(\mathbf{Z}^*) - f(\mathbf{Z}) \\ &= G(\mathbf{Z}) - G(\mathbf{Z}^*) \end{aligned}$$

due to f's convexity. Then we can finally get the first equation. Then we are going to prove the second equation to bound the variance:

$$\begin{aligned} \mathbb{E}\left\|\hat{\mathbf{V}}^{(t)} - \nabla g(\mathbf{Z}^{(t-1)})\right\|_F^2 &= \mathbb{E}\left\|\nabla g_{\hat{\mathcal{E}}_i}(\mathbf{Z}^{(t-1)}) - \nabla g_{\hat{\mathcal{E}}_i}(\tilde{\mathbf{Z}}) + \nabla g(\tilde{\mathbf{Z}} - \nabla g(\mathbf{Z}^{(t-1)}))\right\|_F^2 \\ &= \mathbb{E}\left\|\nabla g_{\hat{\mathcal{E}}_i}(\mathbf{Z}^{(t-1)}) - \nabla g_{\hat{\mathcal{E}}_i}(\tilde{\mathbf{Z}})\right\|_F^2 - \left\|\nabla g(\tilde{\mathbf{Z}} - \nabla g(\mathbf{Z}^{(t-1)}))\right\|_F^2 \\ &\leq \mathbb{E}\left\|\nabla g_{\hat{\mathcal{E}}_i}(\mathbf{Z}^{(t-1)}) - \nabla g_{\hat{\mathcal{E}}_i}(\tilde{\mathbf{Z}})\right\|_F^2 \\ &\leq 2\mathbb{E}\left\|\nabla g_{\hat{\mathcal{E}}_i}(\mathbf{Z}^{(t-1)}) - \nabla g_{\hat{\mathcal{E}}_i}(\mathbf{Z}^*)\right\|_F^2 + 2\mathbb{E}\left\|\nabla g_{\hat{\mathcal{E}}_i}(\mathbf{Z}^*) - \nabla g_{\hat{\mathcal{E}}_i}(\tilde{\mathbf{Z}})\right\|_F^2 \\ &\leq 4L_{\max}(G(\mathbf{Z}^{(t-1)}) - G(\mathbf{Z}^*) + G(\tilde{\mathbf{Z}}) - G(\mathbf{Z}^*)). \end{aligned}$$

□

With the above proposition, we can continue our proof.

$$\begin{aligned} \mathbb{E}\|\mathbf{Z}^{(t)} - \mathbf{Z}^*\|_F^2 &\leq \mathbb{E}\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^*\|_F^2 + 2\eta^2\mathbb{E}\|\Delta^{(t)}\|_F^2 - 2\eta(\mathbb{E}G(\mathbf{Z}^{(t)}) - G(\mathbf{Z}^*)) \\ &\leq \mathbb{E}\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^*\|_F^2 - 2\eta(\mathbb{E}G(\mathbf{Z}^{(t)}) - G(\mathbf{Z}^*)) \\ &\quad + 8L_{\max}\eta^2(G(\mathbf{Z}^{(t-1)}) - G(\mathbf{Z}^*) + G(\tilde{\mathbf{Z}}) - G(\mathbf{Z}^*)). \end{aligned}$$

As the following equation holds for $\mathbf{Z}$ obtained using our proposed solvers:

$$\frac{1}{2}\|\mathbf{Z} - \mathbf{Z}^*\|^2 \leq G(\mathbf{Z}) - G(\mathbf{Z}^*) < \|\mathbf{Z} - \mathbf{Z}^*\|^2.$$

Then we have

$$(1+\eta)\mathbb{E}\|\mathbf{Z}^{(t)} - \mathbf{Z}^*\|_F^2 < (1+8L_{\max}\eta^2)\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^*\|_F^2 + 8L_{\max}\eta^2\|\tilde{\mathbf{Z}} - \mathbf{Z}^*\|_F^2$$

For a fixed stage s, $\mathbf{Z}^{(0)} = \tilde{\mathbf{Z}} = \tilde{\mathbf{Z}}^{(s-1)}$ and $\tilde{\mathbf{Z}}^{(s)} = \frac{1}{m_2}\sum_{t=1}^{m_2}\mathbf{Z}^{(t)}$ as defined in our Algorithm 1. Then we summing over $t = 1, ...m$, and then take expectation. Since Frobenius is convex We have:

$$(\eta - 8L_{\max}\eta^2)m_2\mathbb{E}\left\|\tilde{\mathbf{Z}}^{(s)} - \mathbf{Z}^*\right\|_F^2 < (8L_{\max}\eta^2(m_2+1)+1)\mathbb{E}\left\|\tilde{\mathbf{Z}}^{(s-1)} - \mathbf{Z}^*\right\|_F^2,$$

which can be reformulated as:

$$\mathbb{E}\left\|\tilde{\mathbf{Z}}^{(s)} - \mathbf{Z}^*\right\|_F^2 < \frac{8L_{\max}\eta^2(m_2+1)+1}{(\eta - 8L_{\max}\eta^2)m_2}\mathbb{E}\left\|\tilde{\mathbf{Z}}^{(s-1)} - \mathbf{Z}^*\right\|_F^2.$$

Since $\rho = \frac{8L_{\max}\eta^2(m+1)+1}{(\eta-8L_{\max}\eta^2)m} < 1$, then our USP-VR solver can converge with linear convergence rate. □

C Experiment Details

C.1 Node Classification

Flickr. Flickr is the single class classification dataset which categorize types of images based on the descriptions and common properties of online images. For explicit models, we use the settings in Zeng et al. (2019). As for implicit models, we use the same structure with three implicit graph layers for IGNNs with the maximum iteration number equals to 50 and hidden dimension equal to 256. And we set $m = 20, m_2 = 10, \eta = 0.125$ for our proposed solvers. We train the models with Adam and $lr = 0.01$ for 2000 epochs.

Yelp. Yelp is the multiclass classification dataset which categorize types of businesses based on customer reviewers and friendship. For explicit models, we use the settings in Zeng et al. (2019). As for implicit models, we use the same structure with single implicit graph layers for IGNN with the maximum iteration number equals to 50 and hidden dimension equal to 256. And we set $m = 20, m_2 = 10, \eta = 0.125$ for our proposed solvers. We train the models with Adam and $lr = 0.01$ for 2000 epochs.

Reddit. Reddit is the single class classification dataset which predicts communities of online posts based on user comments. For explicit models, we use the settings in Zeng et al. (2019). As for implicit models, we use the same structure with three implicit graph layers for IGNN with the maximum iteration number equals to 50 and hidden dimension equal to 256. And we set $m = 20, m_2 = 10, \eta = 0.125$ for our proposed solvers. We train the models with Adam and $lr = 0.01$ for 4000 epochs.

PPI. PPI is the multiclass classification dataset classifying protein functions based on the interactions of human tissue proteins. For explicit models, we use the settings in Zeng et al. (2019). As for implicit models, we use the same structure as Gu et al. (2020) used with the maximum iteration number equals to 50. And we set $m = 20, m_2 = 10, \eta = 0.125$ for our proposed solvers. We train the models with Adam and $lr = 0.01$ for 2000 epochs.

OGBN-Arxiv. OGBN-Arxiv is the single class classification dataset classifying protein functions based on the interactions of human tissue proteins. For explicit models, we use the settings in Zeng et al. (2019). As for implicit models, we use the same structure with single implicit graph layers for IGNN with the maximum iteration number equals to 50 and hidden dimension equal to 256. As for IRLS models, we use the structure with ReLU activation and 3 MLP before the IRLS aggregation and 3 MLP after the IRLS aggregation with 512 hidden dimension with the maximum iteration number equals to 20. And we set $m = 20, m_2 = 10, \eta = 0.125$ for our proposed solvers. We train the models with Adam and $lr = 0.001$ for 1000 epochs.

OGBN-Products. OGBN-Products is the single class classification dataset classifying protein functions based on the interactions of human tissue proteins. For explicit models, we use the settings in Zeng et al. (2019). As for implicit models, we use the same structure with two implicit graph layers for IGNN with the maximum iteration number equals to 50 and hidden dimension equal to

32. As for IRLS models, we use the structure with ReLU activation and 2 MLP before the IRLS aggregation and 2 MLP after the IRLS aggregation with 64 hidden dimension with the maximum iteration number equals to 20. And we set $m = 20, m_2 = 10, \eta = 0.125$ for our proposed solvers. We train the models with Adam and $lr = 0.001$ for 1000 epochs.

C.2 Graph Classification

COLLAB is a scientific collaboration dataset. A graph corresponds to a researcher's ego network, i.e., the researcher and its collaborators are nodes and an edge indicates collaboration between two researchers and DD is a bioinformatics dataset. CGS Park et al. (2021) structure we used is the same as their papers' setting with the channel number scales to $0.25\times$ original hidden dimension and 64 batch size otherwise the model will be out-of-memory. IGNN Gu et al. (2020) and our methods are used the same structure as IGNN's for graph classification with 128 hidden number and batch size. The maximum iteration number for different models equal to 50. And we set $m = 20, m_2 = 10, \eta = 0.125$ for our proposed solvers. We train the models with Adam and $lr = 0.01$ for 300 epochs each fold training.