Appendices

A Proofs

A.1 Proof of Proposition 1

Proof. We first define $S_i \in \{0, 1\}$ as a random variable, indicating whether we sample the i-th term or not. By the property of probability, we can decompose the marginal $\mathbb{P}[S_t = 1]$ by

$$\mathbb{P}(S_i = 1) = \mathbb{P}[S_i = 1|S_{i-1} = 1] \cdot \mathbb{P}[S_{i-1} = 1] + \mathbb{P}[S_i = 1|S_{i-1} = 0] \cdot \mathbb{P}[S_{i-1} = 0].$$

Here, $\mathbb{P}[S_i = 1|S_{i-1} = 0] = 0$ in our algorithm because we terminate the iteration if $S_{i-1} = 0$. Thus,

$$\mathbb{P}[S_i = 1] = \mathbb{P}[S_i = 1|S_{i-1} = 1] \cdot \mathbb{P}[S_{i-1} = 1].$$

Moreover, as the random variable S_i is sampled from a Bernoulli distribution $Bernoulli(\alpha)$, we have $\mathbb{P}[S_i = 1|S_{i-1} = 1] = \alpha$ in our algorithm. Plugging this into the above,

$$\mathbb{P}[S_i = 1] = \alpha \mathbb{P}[S_{i-1} = 1].$$

Recursively applying this equation,

$$\mathbb{P}[S_i = 1] = \alpha^{i-t} \mathbb{P}[S_t = 1].$$

Since $\mathbb{P}[S_i = 1] = 1$ for all $i \leq t$ in our algorithm, we have

$$\mathbb{P}[S_i = 1] = \alpha^{i-t}.$$

Considering the possibility of reaching step i, the forward pass of our algorithm can be written as:

$$\hat{Z}^{(i)} = \hat{Z}^{(t)} + \sum_{k=t+1}^{i} \gamma^k \frac{\mathbb{1}\{S_k = 1\}}{\alpha^{(k-t)}} g(W)^k f(X, \mathcal{G}) S^k,$$

where $\mathbb{1}\{S_k = 1\}$ take the value 1 when $S_k = 1$ (i.e., the algorithm reaches step k), otherwise it takes the value 0.

As we continue the algorithm until the termination (i.e., $S_i = 0$), we can write the approximated equilibrium as follows:

$$\hat{Z}^* = \hat{Z}^{(t)} + \sum_{k=t+1}^{\infty} \gamma^k \frac{\mathbb{1}\{S_k = 1\}}{\alpha^{(k-t)}} g(W)^k X S^k.$$

By taking the expectation of the second term, with the definition $A_i = \gamma^i g(W)^i f(X, \mathcal{G}) S^i$ and the condition that $\sum_{k=t+1}^{\infty} A_k \frac{1}{\alpha^{k-t}}$ exists, we have the following:

$$\begin{aligned}
\mathbb{E}\left[\sum_{k=t+1}^{\infty} \gamma^k \frac{\mathbb{1}\{S_k = 1\}}{\alpha^{(k-t)}} g(W)^k X S^k\right] &= \mathbb{E}\left[\sum_{k=t+1}^{\infty} A_k \frac{\mathbb{1}\{S_k = 1\}}{\alpha^{k-t}}\right] \\
&= \sum_{k=t+1}^{\infty} A_k \mathbb{E}\left[\frac{\mathbb{1}\{S_k = 1\}}{\alpha^{k-t}}\right] \\
&= \sum_{k=t+1}^{\infty} A_k \frac{\mathbb{P}[S_k = 1]}{\alpha^{k-t}} \\
&= \sum_{k=t+1}^{\infty} A_k = \sum_{k=t+1}^{\infty} \gamma^k g(W)^k f(X, \mathcal{G}) S^k,
\end{aligned}$$

Combining the above expectation of the second term with the deterministic term $\hat{Z}^{(t)} = \sum_{i=0}^{t} \gamma g(W)^k f(X, \mathcal{G}) S^k$, we get obtain the expectation $\mathbb{E}[\hat{Z}^*]$ as follows:

$$\begin{aligned}\mathbb{E}[\hat{Z}^*] &= \hat{Z}^{(t)} + \mathbb{E}\left[\sum_{k=t+1}^{\infty} \gamma^k \frac{\mathbb{1}\{S_k = 1\}}{\alpha^{(k-t)}} g(W)^k X S^k\right] \\ &= \sum_{k=0}^{t} \gamma^k g(W)^k f(X, \mathcal{G}) S^k + \sum_{k=t+1}^{\infty} \gamma^k g(W)^k f(X, \mathcal{G}) S^k \\ &= \sum_{k=0}^{\infty} \gamma^k g(W)^k f(X, \mathcal{G}) S^k = Z^*.\end{aligned}$$

This indicates that our proposed stochastic solver is an unbiased estimator of the equilibrium Z^*. $\square$

In the above proof, we have an assumption that $\sum_{k=t+1}^{\infty} A_k \frac{1}{\alpha^{k-t}}$ exists. In the case where $\sum_{k=t+1}^{\infty} A_k \frac{1}{\alpha^{k-t}}$ does not exist, we define f_n to be the output of a modified version of the Algorithm 1 where we replace the while-loop with the for-loop up to n step: i.e., we forceful terminate the while-loop if it takes more than n steps. Then, by using the same proof steps except that we replace the infinite sum with the finite sum upto n terms, we conclude that for any n, $\mathbb{E}[f_n] = \sum_{k=0}^{n} A_k$. This implies that our proposed stochastic solver is still an unbiased estimator of the equilibrium Z^* of the forward pass up to the error $\sum_{k=n}^{\infty} A_k$. For any desired error value $\epsilon > 0$ (including machine precision), there exists a sufficiently large n such that $\sum_{k=n}^{\infty} A_k \leq \epsilon$. Thus, the statement in Proposition 1 still holds true up to the machine precision without the condition that $\sum_{k=t+1}^{\infty} A_k \frac{1}{\alpha^{k-t}}$ exists. The output of our algorithm is equivalent to f_n with $n = \infty$. Thus, it is ensured to be unbiased.

B More on Experiments

B.1 Dataset Statistics and Descriptions

The dataset statistics are provided in Table 8. ogbn-products is the largest dataset used in our paper, which contains around 25 million nodes and 61 million edges. Reddit is the densest dataset here with the average node degrees as 50. We follow Li et al. (2023) to use six datasets in our experiments. We provide a detailed description of each dataset as follows:

- **Flickr** is a single-label multi-class classification dataset. The task is to categorize types of images based on the descriptions and common properties of online images. We are using the Flickr dataset as provided in Zeng et al. (2020). Flickr data are collected in the SNAP website [4] from different sources. Flickr contains an undirected graph and a node in the graph represents an image on Flickr. An edge is connected between two nodes if two corresponding images share some common properties (e.g., the same gallery, comments from the same user, etc.). The node features are the 500-dimensional bag-of-word representations of the images. For labels, each image belongs to one of the 7 classes.
- **Reddit** is a single-label multi-class classification dataset. The task is to predict different communities of online posts. We use the Reddit dataset from Hamilton et al. (2017) as in Li et al. (2023). The nodes are online posts and an edge is connected between two posts if the same user comments on both. Word features in posts are 300-dimensional word vectors. Node features are concatenated using 1) the average embedding of the post title, 2) the average embedding of all the post's comments, 3) the post's score, 4) the number of comments on the post.
- **Yelp** is a multi-label multi-class classification dataset. The task is to categorize types of businesses based on users and friendships. We use the Yelp dataset provided in Zeng et al. (2020). Yelp contains a single graph. The nodes are users who provide reviews. If two users are friends, an edge between them is connected. The features of each node are added and normalized using several 300-dimensional vectors representing a review word provided by the user.

[4]https://snap.stanford.edu/data/web-flickr.html

Table 8: Dataset statistics.

Dataset	Nodes	Edges	Avg. Degree	Classes	Features	Train/Val/Test
Flickr	89,250	899,756	10	7	500	0.50/0.25/0.25
Reddit	232,965	11,606,919	50	41	602	0.66/0.10/0.24
Yelp	716,847	6,977,410	10	100	300	0.75/0.10/0.15
PPI	14,755	225,270	14	121	50	0.79/0.11/0.10
ogbn-arxiv	169,343	1,166,243	7	40	128	0.54/0.18/0.29
ogbn-products	2,449,029	61,859,140	25	100	47	0.10/0.02/0.88

- **PPI** is a single-label multi-class classification dataset that contains multiple graphs. The task is to classify protein functions based on the interactions of human tissues. PPI dataset has 24 graphs in total and each graph represents a different human tissue. In a graph, nodes represent proteins and edges indicate interactions between proteins. Each node can have up to 121 labels, which are originally collected from Molecular Signatures dataset (Subramanian et al., 2005) by Hamilton et al. (2017). The dataset splits used in our paper are the same as in Hamilton et al. (2017), i.e., 20 graphs for training, 2 graphs for validation, and 2 graphs for testing.
- **ogbn-arxiv** is a single-label classification dataset that contains a directed graph. The task is to predict the 40 subject areas of arXiv CS papers, such as cs.AI, cs.LG, and cs.OS, which are manually labeled by the authors of the paper and the moderators. each node is an arXiv paper and each directed edge indicates that one paper cites another one. The node feature of each node is a 128-dimensional vector obtained by averaging the embeddings of words in the title and abstract. We download ogbn-arxiv dataset from the OGB website [5]. The detailed descriptions can be found in Hu et al. (2020).
- **ogbn-products** is a single-label multi-class classification dataset which contains an undirected and unweighted graph. Nodes represent products sold on Amazon, and edges between two products indicate that these two products are purchased together. Node features are generated by extracting bag-of-words features from the description of the product followed by a Principal Component Analysis to reduce the dimension to 100. We download ogbn-products dataset from the OGB website [5]. The detailed descriptions can be found in Hu et al. (2020).

B.2 Experimental Setting

For experimental setup, we mainly follow Li et al. (2023). As we use the same experimental setting on some datasets, we reuse the results of some baselines from Li et al. (2023) and Zeng et al. (2020). We compare SEIGNN with 3 implicit GNNs (i.e., USP (Li et al., 2023), MGNNI (Liu et al., 2022), and IGNN (Gu et al., 2020)) and 6 explicit/traditional GNNs (GCN (Kipf and Welling, 2016), GraphSAGE (Hamilton et al., 2017), FastGCN (Chen et al., 2018), ASGCN (Huang et al., 2018), ClusterGCN (Chiang et al., 2019), and GraphSAINT (Zeng et al., 2020)). The experiments are run with 5 different trials. The averaged accuracy and standard deviation are reported. We mainly run the experiments on an RTX-A5000 GPU with 24GB GPU memory.

Model Architecture and Hyperparameters For SEIGNN, we use the same structure with a few implicit graph layers and the same number of linear layers as in MGNNI (Liu et al., 2022) and USP (Li et al., 2023). We select the number of implicit graph layers from {2, 3, 4}. We also conduct a hyperparameter search on learning rate {0.01, 0.005, 0.001} and dropout rate {0.0, 0.2, 0.5}. The number of deterministic steps t in our stochastic solver is chosen from {3, 5} and the continuation probability α is set to 0.5. The hyperparameter γ used in an implicit graph layer is set to 0.8. The Adam optimizer (Kingma and Ba, 2015) is used for optimization. The number of partitions for adding coarse nodes in our mini-batch training method is selected from {50, 100, 200}. The number of target nodes in a mini-batch is configured as follows: 8192 for Flickr and PPI, 10240 for ogbn-arxiv, Yelp, and Reddit, and 16384 for ogbn-products.

[5]https://ogb.stanford.edu/docs/nodeprop

Table 9: GPU Memory Usage (GB) of different models. OOM indicates Out-of-Memory.

	2-layer			3-layer		
	SEIGNN	MGNNI	IGNN	SEIGNN	MGNNI	IGNN
Reddit	7.23	23.77	18.67	8.41	OOM	22.21
Yelp	5.82	15.96	13.65	7.27	20.72	17.34
ogbn-products	9.21	OOM	OOM	9.78	OOM	OOM

B.3 Additional Experimental Results

Besides the overall comparison of accuracy and efficiency, we also investigate the memory usage of different implicit GNNs. Table 9 shows that SEIGNN requires significantly less GPU memory compared with IGNN and MGNNI. In particular, SEIGNN only uses 37% of the memory as IGNN with 3 implicit layers on Reddit. Moreover, we can see that SEIGNN cost less than 10GB GPU memory on ogbn-products while MGNNI and IGNN face the out-of-memory issue using a GPU with 24GB memory.