# Technical Appendix of "Learning Continuous Graph Structure with Bilevel Programming for Graph Neural Networks"

**Author Name**
Affiliation
pcchair@ijcai-22.org

## A Theoretical Results and Proofs

**Theorem 1** (Matrix Convergence Condition)**.** *Suppose $\mathcal{A}$ is a positive definite matrix with n eigenvalues $0 < \lambda_1 \leq \ldots \leq \lambda_n$, where $\lambda_n$ is the maximum eigenvalue. $I - \gamma\mathcal{A}$ is a convergent matrix if $\gamma \in (0, \frac{2}{\lambda_n})$.*

*Proof.* Let $\mathcal{A}$ be a $n \times n$ positive definite matrix and have $n$ eigenvalues $0 < \lambda_1 \leq \cdots \leq \lambda_n$. Assume $v_i$ is the eigenvector corresponding to the eigenvalue $\lambda_i$, we have

$$\begin{aligned}(I - \gamma\mathcal{A})v_i &= v_i - \gamma * \mathcal{A}v_i \\ &= v_i - \gamma * \lambda_i v_i \\ &= (1 - \gamma\lambda_i)v_i.\end{aligned}$$

So, $1 - \gamma\lambda_i$ is the eigenvalue of matrix $I - \gamma\mathcal{A}$, and $v_i$ is the corresponding eigenvector. Because $\lambda_1$ and $\lambda_n$ are respectively the minimum and maximum eigenvalue of $\mathcal{A}$, the maximum eigenvalue of $I - \gamma\mathcal{A}$ will be $\lambda_1' = 1 - \gamma\lambda_1$ or $\lambda_n' = 1 - \gamma\lambda_n$, and the minimum eigenvalue will be $\lambda_n' = 1 - \gamma\lambda_n$ or $\lambda_1' = 1 - \gamma\lambda_1$. Thus the spectral radius of $I - \gamma\mathcal{A}$ is $\rho(I - \gamma\mathcal{A}) = \max(|1 - \gamma\lambda_1|, |1 - \gamma\lambda_n|)$.

Because $I - \gamma\mathcal{A}$ is a convergent matrix, if and only if $\rho(I - \gamma\mathcal{A}) = \max(|1 - \gamma\lambda_1|, |1 - \gamma\lambda_n|) < 1$, thus we have $\gamma \in (0, \frac{2}{\lambda_n})$.

□

**Theorem 2** (Approximation Error Upper Bound)**.** *Suppose $\frac{\partial^2 L}{\partial w \partial w}$ is a positive definite matrix and $I - \gamma * \frac{\partial^2 L}{\partial w \partial w}$ is a convergent matrix, if we use the first $K$ terms of Neumann series to approximate $(\gamma * \frac{\partial^2 L}{\partial w \partial w})^{-1}$, the approximation error is upper bounded by $\frac{\|I - \gamma * \frac{\partial^2 L}{\partial w \partial w}\|_2^{K+1}}{1 - \|I - \gamma * \frac{\partial^2 L}{\partial w \partial w}\|_2}$.*

*When $\gamma \in (0, \frac{2}{\lambda_1 + \lambda_n})$, with the increase of $\gamma$, the upper bound decreases.*

*Proof.* Firstly, we try to derive the approximation error, which comes from using the first $K$ terms of Neumann series to approximate the inverse Hessian $(\gamma * \frac{\partial^2 L}{\partial w \partial w})^{-1}$. Let $\mathcal{A} = I - \gamma * \frac{\partial^2 L}{\partial w \partial w}$, we have the following equation

$$(I + \mathcal{A} + \mathcal{A}^2 + ... + \mathcal{A}^K)(I - \mathcal{A}) = I - \mathcal{A}^{K+1}. \quad (1)$$

Multiplying both sides by $(I - \mathcal{A})^{-1}$, we get

$$\mathcal{A}^{K+1}(I - \mathcal{A})^{-1} = (I - \mathcal{A})^{-1} - \sum_{j=0}^{K} \mathcal{A}^j. \quad (2)$$

From $\mathcal{A}^{K+1}(I - \mathcal{A})^{-1}(I - \mathcal{A}) = \mathcal{A}^{K+1}$, we can get another equation:

$$\mathcal{A}^{K+1}(I - \mathcal{A})^{-1} = \mathcal{A}^{K+1} + \mathcal{A}^{K+1}(I - \mathcal{A})^{-1}\mathcal{A} \quad (3)$$

Due to the property of matrix norm, we have

$$\|\mathcal{A}^{K+1}(I - \mathcal{A})^{-1}\|_2 \leq \|\mathcal{A}^{K+1}\|_2 + \|\mathcal{A}^{K+1}(I - \mathcal{A})^{-1}\|_2\|\mathcal{A}\|_2.$$

Note that $\mathcal{A}$ is a convergent matrix, so its spectral radius is less than 1. It is easy to know that $A$ is a real-valued symmetric matrix, thus $\rho(\mathcal{A}) = \|\mathcal{A}\|_2 < 1$. We have

$$\|\mathcal{A}^{K+1}(I - \mathcal{A})^{-1}\|_2 \leq \frac{\|\mathcal{A}\|_2^{K+1}}{1 - \|\mathcal{A}\|_2} \quad (4)$$

Combined with Eq. (2), we have

$$\|(I - \mathcal{A})^{-1} - \sum_{j=0}^{K} \mathcal{A}^j\|_2 \leq \frac{\|\mathcal{A}\|_2^{K+1}}{1 - \|\mathcal{A}\|_2} \quad (5)$$

Finally, by replacing $A$ with $I - \gamma * \frac{\partial^2 L}{\partial w \partial w}$, we have the error upper bound as

$$\|(\gamma * \frac{\partial^2 L}{\partial w \partial w})^{-1} - \sum_{j=0}^{K} (I - \gamma * \frac{\partial^2 L}{\partial w \partial w})^j\|_2 \leq \frac{\|I - \gamma * \frac{\partial^2 L}{\partial w \partial w}\|_2^{K+1}}{1 - \|I - \gamma * \frac{\partial^2 L}{\partial w \partial w}\|_2}$$

Next, we try to analyze the relationship between $\gamma$ and the error bound. From above analysis, we have $0 < \|\mathcal{A}\|_2 < 1$. It is easy to know that, with such value range, the upper bound $\frac{\|\mathcal{A}\|_2^{K+1}}{1 - \|\mathcal{A}\|_2}$ is monotone increasing with $\|\mathcal{A}\|_2$, or equivalently with $\rho(\mathcal{A})$.

As we discussed in Theorem 1, the spectral radius $\rho(\mathcal{A}) = \rho(I - \gamma * \frac{\partial^2 L}{\partial w \partial w}) = \max(|1 - \gamma\lambda_1|, |1 - \gamma\lambda_n|)$, where $\lambda_1$ and $\lambda_n$ is the minimum and maximum eigenvalue of $\frac{\partial^2 L}{\partial w \partial w}$. Assume that $\frac{\partial^2 L}{\partial w \partial w}$ is a positive definite matrix, which means $\lambda_n \geq \lambda_1 > 0$. In this case, when $\gamma \in (0, \frac{2}{\lambda_1 + \lambda_n})$, with the increase of $\gamma$, $\max(|1 - \gamma\lambda_1|, |1 - \gamma\lambda_n|)$ and $\rho(\mathcal{A})$ decrease, so does the upper bound.

So, when $\gamma \in (0, \frac{2}{\lambda_1 + \lambda_n})$, with the increase of $\gamma$, the error upper bound decreases.

□

**Algorithm 1** LCGS

**Input**: $X, Y, A$.
**Parameter**: $\alpha, \beta, T$.

1: Initialize $\theta$ with $\hat{A}$.
2: $t \leftarrow 0$
3: **while** stopping condition is not met **do**
4: $w_{\theta,t+1} \leftarrow w_{\theta,t} - \alpha * \nabla_w L(w_{\theta,t}, \theta)$ {Optimize inner variable $w$}
5: $t \leftarrow t + 1$
6: **if** $t = 0 \pmod T$ **then**
7: $hg \leftarrow$ Improved Neumann-IFT$(L, F)$
8: $\theta \leftarrow \theta - \beta * hg$ {Optimize outer variable $\theta$}
9: **end if**
10: **end while**
11: **return** $w, \theta$ {Best found weights and graph structure}

# B Algorithms

We outline the LCGS framework in Algorithm 1. Note that we use the normal stochastic gradient descent to optimize inner variable $w$ at line 4. At line 7, we use improved Neumann-IFT algorithm to estimate the hypergradient, which will be used to optimize outer variable $\theta$ at line 8.

# C Details for Node Classification Experiments

## C.1 Datasets Statistics

The statistics of the datasets used in semi-supervised node classification experiments are shown in Table 1. For the normal graph scenario, we use the same dataset split as [Kipf and Welling, 2017]. For the noisy graph scenario, we only consider the largest connected component (LCC), following [Wu *et al.*, 2019; Jin *et al.*, 2020].

## C.2 Implementation Details

We use two layers GCN with 16 hidden neurons and ReLu activation in all experiments. As additional regularization technique, we apply dropout with $p = 0.5$ as in previous work [Kipf and Welling, 2017]. We use Adam optimizer to optimize GNN parameters with learning rate $\alpha = 0.02$ and SGD optimizer to optimize graph generator with learning rate $\beta$ from $\{0.1, 0.05, 0.005\}$ for Cora and Citeseer, $\beta = 0.001$ for Pubmed. In all experiments, we truncate the inner objective optimization each $T$ iteration with $T = 5$ or 10, and estimate the maximum eigenvalue by Power Itertion with $S = 10$. For normal graph scenario, we set $c = 0$ for Cora and Pubmed, $c = 1$ for Citeseer; $K = 1$ for Cora, $K = 10$ for Citeseer and $K = 20$ for Pubmed. For noisy graph scenario, we set $c = 0$ or 1 for Cora and $c = 2$ for Citeseer; $K = 10$ for both Cora and Citeseer in all perturbation rate.

For our LCGS, we split the validation set evenly to form the set (A) and set (B), as [Franceschi *et al.*, 2019]. We set the outer objective as unregularized cross-entropy loss on (A) and minimize it by hypergradient computation. Besides, we select the best performance accord to the loss or accuracy on set (B).

| | Nodes | Edges | Train / valid / test |
|---|---|---|---|
| Cora | 2708 | 5429 | 140 / 500 / 100 |
| Citeseer | 3327 | 4732 | 120 / 500 / 1000 |
| Pubmed | 19717 | 44338 | 60 / 500 / 1000 |
| Cora | 2485 | 5069 | 247 / 249 / 1988 |
| Citeseer | 2110 | 3668 | 210 / 211 / 1688 |

Table 1: Dataset statistics. The top three rows are datasets statistics for the normal graph scenario, and the bottom two rows are datasets statistics for the noisy graph scenario.

# D Experiments on Scene Graph Generation

Scene graph generation (SGG) is an important task in computer vision, as it aims to provide a structured representation of image content, called scene graph [Johnson *et al.*, 2015]. Scene graph is a directed graph, where nodes represent objects and edges represent relationship between object pairs in an image. In SGG tasks, the graph structure for each image is often noisy and incomplete due to the noisy and missing annotation, thus learning the underlying graph structures is important for SGG.

## D.1 Method Instantiation for Scene Graph Generation

There are some differences between SGG and node classifications. Firstly, semi-supervised node classification on citation networks is a transductive learning task, which means the training, validation and testing nodes share the same graph. In contrast, SGG is an *inductive learning* task, which requires the method generalizable to new images in test. Secondly, graphs in node classification task are usually undirected graphs, while scene graphs are *directed*, due to the directivity of relationships.

Considering these differences, we implement our LCGS framework with slight modifications for SGG. Firstly, we slightly modify the graph generator by using function $f_\theta$ to model graph structure with pair-wise node features. Without loss of generality, we implement $f_\theta$ via a two-layer neural network, which is parameterized by $\theta$. Then, due to the fact that scene graphs are directed graphs, we use GGNN [Li *et al.*, 2016] as the GNN backbone to utilize the direction information. After GGNN, we concatenate the features of two related objects as the predicate feature, then use a Softmax classifier to classify predicate categories of the given object pair.

To make the continuous graph structure more rational, we impose additional constraints: task-irrelevant connections (noise) should be removed, while object correlations in true graph should be maintained. To this end, we multiply the output of graph generator by a binary prior mask matrix $M = (m_{ij})_{i,j=1}^{|V|}$, where $m_{ij}$ is calculated based on the relationship between the class labels of the $i$-th object and the $j$-th object in training data. The prior can reduce the searching space of possible graphs.

### D.2 Settings

**Datasets.** We evaluated the proposed LCGS on Visual Genome [Krishna *et al.*, 2017] dataset. Visual Genome (VG) is a popular benchmark for SGG, which contains 108,077 images with tens of thousands of unique object and predicate categories. Since most of categories have very limited instances, we follow previous works [Xu *et al.*, 2017; Zellers *et al.*, 2018; Tang *et al.*, 2019] to use the most frequent 150 object categories and 50 relationships. Then the entire dataset is divided into training set and test set in proportion of 70% and 30%. We further pick 5000 images from the training set as the validation set, which will be used to optimize outer variables in bilevel programming.

**Protocols.** We consider the **Predicate Classification (PredCls)** task: given the object bounding boxes and their labels in an image, predict the predicate (relationship) of given object pair.

**Metrics.** All the methods are evaluated by the Recall@K (short as R@K) that measures the fraction of ground-truth relationship triplets that appear among the top K most confident triplet predictions in an image. As mentioned in [Zellers *et al.*, 2018; Tang *et al.*, 2019], the VG dataset is highly imbalanced, we also utilize the mean Recall@K (short as mR@K) that calculates the R@K on each category independently and then averages the results to evaluate relationships in a balanced way.

**Comparative Methods.** To verify the effectiveness of our proposed framework, we compare it with four representative SGG models: **IMP** [Xu *et al.*, 2017], **Motifs** [Zellers *et al.*, 2018], **VCTree** [Tang *et al.*, 2019] and **GPS-Net** [Lin *et al.*, 2020]. Besides, we introduce a baseline model **Baseline Full**, who uses full connected graph as input of GNN model in both training and testing, which is a common strategy used by many GNN-based SGG models.

### D.3 Implementation Details

We adopt Faster-RCNN with ResNeXt-101-FPN backbone (based on [Tang, 2020]) to extract RoI features on given object bounding boxes.

We train our LCGS framework on a single RTX-3090 GPU with batch size $b = 10$. The inner objective and outer objective are optimized by SGD with inner learning rate $\alpha$ and outer learning rate $\beta$ are both set to 0.01. Due to the input of GGNN is directed graph, we use row-normalized function with $c = 0$ for renormalization. We truncate the inner objective optimization each $T$ iteration with $T = 10$, and calculate the hypergradient by improved Neumann-IFT algorithm with $K = 5$ and $S = 10$.

### D.4 Experimental Results

We present the comparative results in Table 2. Compared with other SGG methods, the proposed LCGS framework with continuous graph generator and a GGNN backbone achieves better performance on both Recall and mean Recall metrics. Moreover, LCGS is marginally better than Baseline Full on Recall metric, while significantly better on mean Recall metric. We argue that this is because the mean Recall metric depends on many infrequent predicate categories, which may be easily influenced by noisy edges due to fewer samples.

| | R@50 | R@100 | mR@50 | mR@100 |
|---|---|---|---|---|
| IMP | 59.3 | 61.3 | 9.8 | 10.5 |
| Motifs | 67.9 | 67.9 | 14.6 | 15.8 |
| VCTree | 66.2 | 68.1 | 14.9 | 16.1 |
| GPS-Net | 65.2 | 67.1 | 15.2 | 16.6 |
| IMP* | 61.1 | 63.1 | 11.4 | 12.3 |
| Motifs* | 65.3 | 67.1 | 14.5 | 15.6 |
| VCTree* | 65.9 | 67.7 | 16.4 | 17.7 |
| Baseline Full | 65.9 | 68.1 | 16.8 | 18.5 |
| LCGS | **66.2** | **68.2** | **17.5** | **19.5** |

Table 2: The performance on PredCls task. The best results are in bold. The star symbol indicates that we reproduce the results using our codes.

## References


[Franceschi *et al.*, 2019] L. Franceschi, M. Niepert, M. Pontil, and X. He. Learning discrete structures for graph neural networks. In *ICML*, 2019.

[Jin *et al.*, 2020] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *SIGKDD*, 2020.

[Johnson *et al.*, 2015] J. Johnson, R. Krishna, M. Stark, L. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *CVPR*, 2015.

[Kipf and Welling, 2017] T.N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Krishna *et al.*, 2017] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L. Li, D. Michael, S. Bernstein1, and L. Fei-Fei. Visual Genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017.

[Li *et al.*, 2016] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. In *ICLR*, 2016.

[Lin *et al.*, 2020] X. Lin, C. Ding, J. Zeng, and D. Tao. GPS-Net: Graph property sensing network for scene graph generation. In *CVPR*, 2020.

[Tang *et al.*, 2019] K. Tang, H. Zhang, B. Wu, W. Luo, and W. Liu. Learning to compose dynamic tree structures for visual contexts. In *CVPR*, 2019.

[Tang, 2020] Kaihua Tang. A scene graph generation codebase in pytorch, 2020. https://github.com/KaihuaTang/Scene-Graph-Benchmark.pytorch.

[Wu *et al.*, 2019] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples on graph data: Deep insights into attack and defense. In *IJCAI*, 2019.

[Xu *et al.*, 2017] D. Xu, Y. Zhu, C.B. Choy, and L. Fei-Fei. Scene graph generation by iterative message passing. In *CVPR*, 2017.

[Zellers *et al.*, 2018] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi. Neural motifs: Scene graph parsing with global context. In *CVPR*, 2018.