A Pseudocode of Ensemble Multi-Relational Message Passing Mechanism

Algorithm 2 Ensemble Multi-Relational Message Passing Mechanism

Input: Candidate adjacency matrix $\left\{\hat{\tilde{\mathbf{A}}}^1, \cdots, \hat{\tilde{\mathbf{A}}}^r\right\}$, the feature matrix $\mathbf{H}$, the tradeoff parameters λ_1, λ_2.
Output: The final propagated embedding matrix $\mathbf{Z}^{(K)}$.
1: Initialization:$\boldsymbol{\mu} = [\frac{1}{R}, \frac{1}{R}, \cdots, \frac{1}{R}]$, $\mathbf{Z}^{(0)} = \mathbf{H}$.
2: **repeat**
3: **Step1**: update relational coefficients according Algorithm 1.
4: **Step2**: update feature embeddings according Eq.(10).
5: **until** Convergence
6: **return** $\mathbf{Z}^{(K)}$

B Details on Experimental Setup

B.1 Datasets

- **DBLP**[3] is a computer science bibliography website. We adopt a subset of DBLP extracted by [Fu *et al.*, 2020]. The authors have four research areas (Database, Data Mining, Artificial Intelligence, and Information Retrieval). Also following [Fu *et al.*, 2020], the author nodes are divided into training, validation, and testing sets of 400, 400, and 3257 nodes, respectively.
- **ACM** is also a citation network. We adopt a subset of ACM extracted by [Lv *et al.*, 2021], the papers are divided into three research areas (Database, Wireless communication, Data mining). Following [Yun *et al.*, 2019], the paper nodes are divided into training, validation, and testing sets of 600, 300, and 2125 nodes, respectively.
- **MUTAG**[4] describes the interactions between complex molecules that can be classified as isMutagenic or not. We remove relations that were used to create entity labels, i.e., isMutagenic. Following HGB [Lv *et al.*, 2021], the data set is divided into training, validation, and testing sets of 218, 54, and 68 nodes, respectively.
- **BGS**[4] describes information about relations between named rock units that can be classified as hasLithogenesis or not. We remove relations that were used to create entity labels, i.e., hasLithogenesis. Following HGB [Lv *et al.*, 2021], the data set is divided into training, validation, and testing sets of 94, 23, and 29 nodes, respectively.

B.2 Baselines

The publicly available implementations of baselines can be found at the following URLs:

- GCN: https://github.com/THUDM/HGB/tree/master/NC/benchmark/methods/GNN
- GAT: https://github.com/THUDM/HGB/tree/master/NC/benchmark/methods/GNN
- HAN: https://github.com/THUDM/HGB/tree/master/NC/benchmark/methods/HAN
- RGCN: https://github.com/tkipf/relational-gcn
- e-RGCN: https://github.com/thiviyanT/torch-rgcn

B.3 Parameter settings

During training, we set the learning rate lr = 1e-2 for all datasets, dropout rate d = 0.5 if MLP is used. We employ the validation set for early stopping and set the patience for different datasets due to different convergence situations, i.e., DBLP and ACM for 150, MUTAG for 70, and BGS for 25. We adjust the two most important hyperparameters in our model, i.e., λ_1 form {0.05, 0.1, 0.15, 0.2, 0.3, 1, 10, 100} that controls the proportion of original information of nodes; and the update rate of relational coefficients $\frac{\lambda_2}{\lambda_1}$ from (0.1, 150). The implementations of RGCN and e-RGCN are provided by their authors, the ones of all the other baselines are obtained from HGB[5]. To make a fair and reliable comparison, we turn all hyperparameters of the baselines to get the best performance.

C About the time complexity

The time complexity of EMR-GNN mainly involves from MLP and EnMP. The computational complexity for feature extraction is $O(Nd^2)$, where N is the number of nodes, and d is the dimension of the node features. The computational complexity for $\boldsymbol{\mu}$ optimization is $O(R(N^2d + Nd^2))$, where R is the number of relations. Regarding the feature update process, the complexity is $O((R + d)N^2)$. In summary, the time complexity of a K layer EMR-GNNs is $O(K(RdN^2 + RNd^2))$, RGCN and ours have the same order of magnitude.

[3] https://dblp.uni-trier.de/
[4] https://github.com/tkipf/relational-gcn/tree/master/rgcn/data
[5] https://github.com/THUDM/HGB