Supplementary Information for "Contrastive Semi-Supervised Clustering Based on Pairwise Constraint for Identifying Influential Nodes in Complex Networks"

1 Supplementary Information for Methodology

1.1 Model Optimization

Below, we briefly outline the parameter update process for the second phase (the first phase follows a similar but simpler procedure and is omitted here for brevity). Let $L_t = L_{con} + L_{KL}$, the gradient of L_t with respect to $\boldsymbol{W}$ is computed using the chain rule as follows:

$$\frac{\partial L_t}{\partial \boldsymbol{W}} = \frac{\partial L_t}{\partial \boldsymbol{Z}} \cdot \frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{W}} \tag{1}$$

$$\frac{\partial L_t}{\partial \boldsymbol{Z}} = \frac{\partial L_{KL}}{\partial \boldsymbol{Z}} + \frac{\partial L_{con}}{\partial \boldsymbol{Z}} \tag{2}$$

$$\frac{\partial L_{KL}}{\partial Z_i} = 2\sum_j (1 + \|Z_i - \mu_j\|^2)^{-1} \times (p_{ij} - q_{ij})(Z_i - \mu_j) \tag{3}$$

$$\frac{\partial L_{con}}{\partial Z_i} = 2\sum_{j=1}^{n} C_{ij}(Z_i - Z_j) \tag{4}$$

To simplify the calculation, let $\boldsymbol{A}' = \widetilde{\boldsymbol{D}}^{-\frac{1}{2}} \widetilde{\boldsymbol{A}} \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$. The gradient of $\boldsymbol{Z}$ with respect to $\boldsymbol{W}^L$ is derived as:

$$\frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{W}^L} = \Phi'(\boldsymbol{A}'\boldsymbol{H}^{L-1}\boldsymbol{W}^L) \cdot \frac{\partial \boldsymbol{A}'\boldsymbol{H}^{L-1}\boldsymbol{W}^L}{\partial \boldsymbol{W}^L} \tag{5}$$

The gradient of L_t with respect to $\boldsymbol{W}^L$ can be derived as follows:

$$\frac{\partial L_t}{\partial \boldsymbol{W}^L} = (\boldsymbol{A}'\boldsymbol{H}^{L-1}\boldsymbol{W}^L)^T \cdot \frac{\partial L_t}{\partial \boldsymbol{Z}} \odot \Phi'(\boldsymbol{A}'\boldsymbol{H}^{L-1}\boldsymbol{W}^L) \tag{6}$$

Similarly, we can also derive the gradient for the weights of the $(L-1)$-th layer:

$$\frac{\partial L_t}{\partial \boldsymbol{W}^{L-1}} = \frac{\partial L_t}{\partial \boldsymbol{Z}} \cdot \frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{H}^{L-1}} \cdot \frac{\partial \boldsymbol{H}^{L-1}}{\partial \boldsymbol{W}^{L-1}} \tag{7}$$

The gradients for the weights of the remaining layers can be derived in the same way. Finally, the weights of each layer l ($l = 1, 2, \ldots, L$) can be updated as following:

$$\boldsymbol{W}^l = \boldsymbol{W}^l - \eta \frac{\partial L_t}{\partial \boldsymbol{W}^l} \tag{8}$$

where η is the learning rate.

1.2 Time Complexity Analysis

Assuming the RE similarity is precomputed prior to model training (which can be done in $O(T_s n|E|)$ where T_s is the number of iterations in the power iteration method), we analyze the time complexity of each component as follows. Feature construction requires the most informative eigenvectors of RE similarity matrix, which can be obtained in $O(n^2)$ time by computing only a small number of the top eigenvectors (those corresponding to the largest eigenvalues). The degree normalization of the adjacency matrix with self-loop (Eq. 5 in the main manuscript) requires $O(|E|)$ operations. For the $(l+1)$-th GCN layer, the matrix multiplication between $\boldsymbol{H}^l$ and $\boldsymbol{W}^{l+1}$ costs $O(d^l d^{l+1} n)$, and the subsequent aggregation step costs $O(d^{l+1}|E|)$. Thus, the overal complexity per layer is approximately $O(d^{l+1}|E| + d^l d^{l+1} n)$. The reconstruction loss (Eq. 6 in the main manuscript) involves the term $\sigma(\boldsymbol{Z}\boldsymbol{Z}^T)$, which dominates its time complexity at $O(d^L n^2)$. For pairwise constraint generation (Eq. 7 in the main manuscript), determining constraints for unlabeled nodes requires sorting each row of the RE similarity matrix $\boldsymbol{S}$, leading to $O(n^2 \log n)$ total time. Once sorted, constraints can be retrieved directly. The contrastive loss (Eq. 8 in the main manuscript) has a time complexity of $O(d^L n^2)$, reducible to $O(d^L|E'|)$ via sparse matrix operations, where $|E'|$ is number of non-zero entries in constraint matrix and $|E'| < n^2$. The KL divergence loss (Eq. 9 in the main manuscript) mainly arises from computing the soft assignment (Eq. 10 in the main manuscript), costing $O(Kd^L n)$, with $K = 2$ being the number of clusters. Additionally, k-means clustering is applied per epoch, requiring $O(T_k K d^L n)$ time, where T_k is the number of k-means iterations. Considering that the number of GCN layers L is small, layer dimensions are comparable, T_k is typically much smaller than n^2, and $logn$ is manageable for real-world networks, the overall time complexity of PCGCN is approximately $O(T_p(d^L|E| + d^L n^2))$ where T_p is the number of training epochs.

2 Supplementary Information for Experiments

2.1 Datasets and Baselines

Datasets. The statistical information of the used networks is summarized in Table 1. To evaluate the performance of different methods, we follow the established practice [6,12,17,1] to employ SIR to annotate node influence. Specifically, the top 5% of nodes with the highest influence scores are labeled as influential, while the rest are considered non-influential. To address the class imbalance issue in node classification and graph clustering methods, we evenly partition non-influential nodes into bins based on their SIR scores and uniformly sample 10% of nodes from these bins. These sampled non-influential nodes are then combined with all influential nodes to form the final evaluation set for each network.

Baseline methods and their settings. For centrality-based methods, Degree [2] is a local centrality measure, while PageRank [4] and Eigenvector [3]

Table 1. Brief information about the networks.

Datasets	cG	cH	SC	crime	netsci	WM	TG	euroroad
Nodes	4158	8638	10320	829	379	7992	168114	1039
Edges	13422	24806	17988	1473	914	78740	6797557	1305
Ref	[10]	[10]	[9]	[9]	[10]	[10]	[13]	[9]

capture global influence. Although numerous centrality measures exist, we select these three based on empirical findings: Degree is representative of local centrality measures due to its high correlation with others, while PageRank and Eigenvector are strongly correlated with Betweenness [11] and Closeness [14] centrality, respectively. This selection ensures both efficiency and avoidance of redundancy.

For supervised methods, CNT [6] leverages a Transformer to encode each node as a sequence of its fixed-size neighbors, represented by their degrees, and uses self-attention to generate node embeddings, which are then mapped to influence scores via a fully-connected layer. M-RCNN [12] employs BFS to extract fixed-size neighborhoods and constructs three feature matrices capturing node-, community-, and macro-level (k-core) information, which are processed by a CNN to regress SIR-based influence scores. ReGCN [15] constructs node features using RE similarity, similar to PCGCN, but is trained in a supervised manner using cross-entropy loss between annotated and predicted labels. CNT and M-RCNN are pre-trained models provided by their authors, while ReGCN is trained from scratch using an 8:2 train-test split.

For unsupervised and semi-supervised methods, NMF+KM employs non-negative matrix factorization to derive node embeddings, followed by k-means clustering (i.e., a spectral clustering). LCCF [5] incorporates graph Laplacian regularization to capture the intrinsic geometric structure of the data for clustering. GRACE [8] integrates node attributes and structural information via a unified graph convolution framework. CONVERT [16] employs contrastive learning by generating two embeddings, one of which is augmented, to improve clustering performance. GCA+ [7] enhances node embeddings through contrastive learning where positive and negative samples are constructed based on node similarity combining attributes and structure. Although not specifically designed for it, these methods can address influential node identification by framing it as a clustering task. For influential node identification, the unsupervised methods cluster nodes into two groups (influential and non-influential), while GCA+ is connected to a classifier trained separately on a small set of annotated nodes, thus constituting a semi-supervised approach.

2.2 Analysis on Running Time

Fig. 1 shows the average running time per epoch of PCGCN on different networks (left) and as a function of network size (right). Note that a direct comparison

with other methods is omitted due to differences in model architectures, hyperparameters, and implementations. As shown, PCGCN requires a little time per epoch on small networks. Even on the largest network tested, the average time per epoch remains below 1.2 seconds. Although running time increases with network size, the observed growth seems following a polynomial trend with a much smaller quadratic coefficient than theoretical value of d^L. This discrepancy can be attributed to both the underlying optimization of basic operations (e.g., matrix multiplication) in PyTorch and our implementation-level accelerations, such as partitioning large sparse matrices into smaller blocks. Overall, PCGCN can be executed efficiently and scaled to large networks with appropriate implementation optimizations, even under limited computational resources.

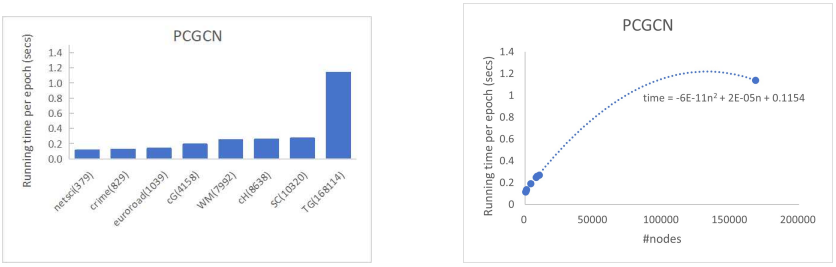

Fig. 1. The scalability of PCGCN (left: running time per epoch on different networks with number of nodes indicated in parentheses; right: running time as a function of network size).

2.3 Ablation Study

The impact of different node feature configurations. The results of PCGCN under different node feature configurations are shown in Table 2. As shown, the combined use of RE similarity-derived features and node degree ("Reg+Deg") consistently outperforms the use of RE similarity features alone ("Reg") across all networks and evaluation metrics. Furthermore, "Reg+Deg" also surpasses the node degree feature ("Deg") on all datasets except WM, where the two achieve comparable results. The comparison between "Reg" and "Deg" reveals that node degree feature lead to better performance on six out of the eight networks. These findings suggest that while the combination of both feature types yields the best overall results, the contribution of node degree feature appears to be more substantial than that of RE similarity-derived features in most cases. Nevertheless, both types of features are beneficial for the performance of PCGCN.

2.4 Parameter Analysis

Test of r_1. Fig. 2 illustrates the performance of PCGCN under different r_1 values. The model exhibits stable performance on cG, cH, crime, and euroroad across the evaluated r_1 range. A slight improvement is observed on WM and

Table 2. The results of PCGCN under different node feature configurations. "Reg" denotes RE similarity-derived features, "Deg" represents node degree, and "Reg+Deg" indicates their concatenation.

Feature	Metric	cG	cH	SC	crime	netsci	WM	TG	euroroad
Reg+Deg	ACC	**0.9599**	**0.9715**	**0.9877**	**0.9587**	**0.9434**	**0.9858**	**0.9455**	**0.8079**
	NMI	**0.7577**	**0.8313**	**0.9109**	**0.7813**	**0.6727**	**0.8950**	**0.6812**	**0.3037**
	ARI	**0.8446**	**0.8880**	**0.9510**	**0.8395**	**0.7811**	**0.9432**	**0.7919**	**0.3746**
Reg	ACC	0.8542	0.8441	0.9380	0.9504	0.9245	0.9188	0.9243	0.6887
	NMI	0.3595	0.3378	0.6673	0.7176	0.6526	0.5946	0.6165	0.0472
	ARI	0.4929	0.4669	0.7622	0.8089	0.7104	0.6941	0.7182	0.1018
Deg	ACC	0.9407	0.9707	0.9858	0.9421	0.9057	**0.9858**	0.9411	0.7881
	NMI	0.6570	0.8280	0.9001	0.7249	0.5546	**0.8950**	0.6545	0.2848
	ARI	0.7743	0.8851	0.9433	0.7794	0.6509	**0.9432**	0.7757	0.3277

TG as r_1 increases. More notably, on SC, performance improves significantly as r_1 increases from 0.05 to 0.10, then stabilizes. A similar trend is observed on netsci, where performance remains stable between $r_1 = 0.05$ and 0.10, improves substantially at $r_1 = 0.15$, and then plateaus. Overall, PCGCN demonstrates robust performance across the r_1 range of 0.05 to 0.20, with moderate to significant improvements occurring at certain thresholds in some cases. This trend is expected, as increasing r_1 provides more supervisory signal from annotated nodes, which generally enhances the model's learning capability.

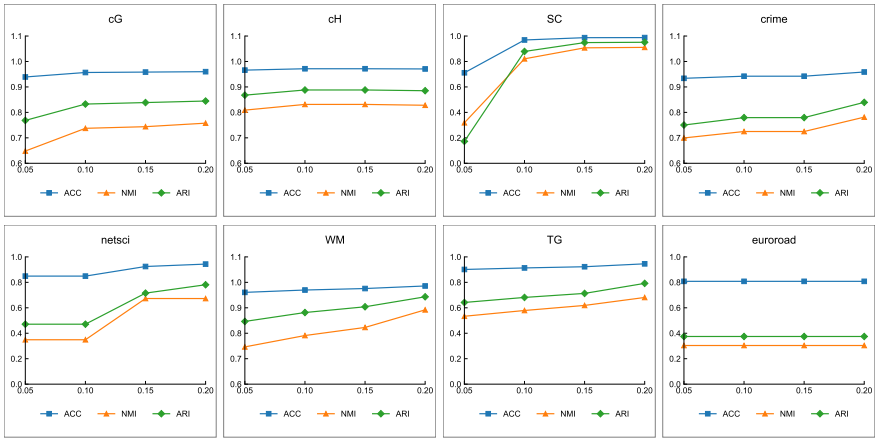

Fig. 2. The performance of PCGCN with different r_1 values.

Test of r_2. Fig. 3 shows the performance of PCGCN under different r_2 values. On cG, cH, SC, WM, and euroroad, performance remains stable as r_2 varies

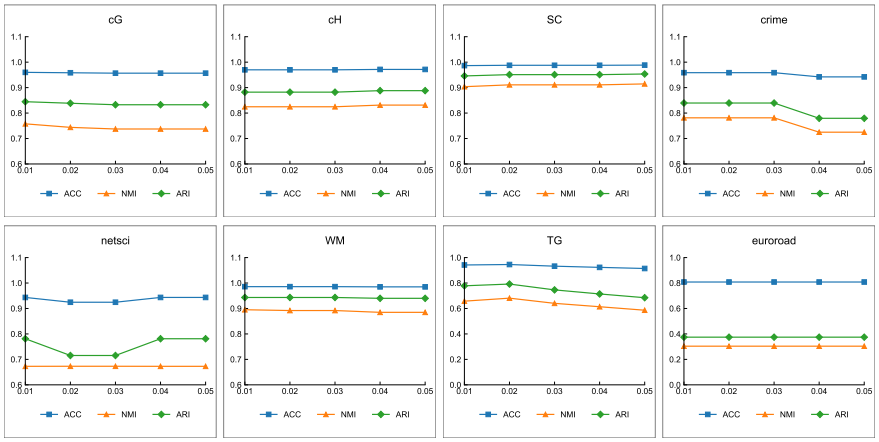

Fig. 3. The performance of PCGCN with different r_2 values.

from 0.01 to 0.05. In contrast, performance on netsci decreases at $r_2 = 0.02$, improves at $r_2 = 0.03$, and then stabilizes. For crime and TG, performance remains stable within certain intervals but slightly degrades once r_2 exceeds a threshold. Overall, PCGCN exhibits robust performance across the evaluated r_2 range, though performance degradation occurs in some networks at higher values. This degradation is likely due to the introduction of noisy node pairs into the constraints as r_2 increases, which may include less reliable similar or dissimilar nodes. These observations suggest the value of developing more selective filtering strategies for constraint construction in future work.

References

1. Bhattacharya, R., Nagwani, N.K., Tripathi, S.: Detecting influential nodes with topological structure via graph neural network approach in social networks. International Journal of Information Technology **15**(4), 2233–2246 (2023)
2. Bonacich, P.: Factoring and weighting approaches to status scores and clique identification. Journal of mathematical sociology **2**(1), 113–120 (1972)
3. Bonacich, P., Lloyd, P.: Eigenvector-like measures of centrality for asymmetric relations. Social networks **23**(3), 191–201 (2001)
4. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer networks and ISDN systems **30**(1-7), 107–117 (1998)
5. Cai, D., He, X., Han, J.: Locally consistent concept factorization for document clustering. IEEE Transactions on Knowledge and Data Engineering **23**(6), 902–913 (2010)
6. Chen, L., Xi, Y., Dong, L., Zhao, M., Li, C., Liu, X., Cui, X.: Identifying influential nodes in complex networks via transformer. Information Processing & Management **61**(5), 103775 (2024)
7. Chi, H., Ma, Y.: Enhancing contrastive learning on graphs with node similarity. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 456–465 (2024)

8. Fanseu Kamhoua, B., Zhang, L., Ma, K., Cheng, J., Li, B., Han, B.: Grace: A general graph convolution framework for attributed graph clustering. ACM Transactions on Knowledge Discovery from Data **17**(3), 1–31 (2023)
9. Kunegis, J.: Konect: the koblenz network collection. In: Proceedings of the 22nd international conference on world wide web. pp. 1343–1350 (2013)
10. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: Densification and shrinking diameters. Acm Transactions on Knowledge Discovery from Data **1**(1), 2 (2007)
11. Newman, M.E.: A measure of betweenness centrality based on random walks. Social networks **27**(1), 39–54 (2005)
12. Ou, Y., Guo, Q., Xing, J.L., Liu, J.G.: Identification of spreading influence nodes via multi-level structural attributes based on the graph convolutional network. Expert Systems with Applications **203**, 117515 (2022)
13. Rozemberczki, B., Sarkar, R.: Twitch gamers: a dataset for evaluating proximity preserving and structural role-based node embeddings. arXiv preprint arXiv:2101.03091 (2021)
14. Sabidussi, G.: The centrality index of a graph. Psychometrika **31**(4), 581–603 (1966)
15. Wu, Y., Hu, Y., Yin, S., Cai, B., Tang, X., Li, X.: A graph convolutional network model based on regular equivalence for identifying influential nodes in complex networks. Knowledge-Based Systems **301**, 112235 (2024)
16. Yang, X., Tan, C., Liu, Y., Liang, K., Wang, S., Zhou, S., Xia, J., Li, S.Z., Liu, X., Zhu, E.: Convert: Contrastive graph clustering with reliable augmentation. In: Proceedings of the 31st ACM international conference on multimedia. pp. 319–327 (2023)
17. Zhao, G., Jia, P., Zhou, A., Zhang, B.: Infgcn: Identifying influential nodes in complex networks with graph convolutional networks. Neurocomputing **414**, 18–26 (2020)