Investigating the trade-off between graph structure and community semantics in community detection methods

Hossein Arjomandi
University of Illinois,
Urbana-Champaign
Illinois, USA
hm31@illinois.edu

Eunice Chan
University of Illinois,
Urbana-Champaign
Illinois, USA
ecchan2@illinois.edu

Xinyu He
University of Illinois,
Urbana-Champaign
Illinois, USA
xhe34@illinois.edu

ABSTRACT

In various scientific domains, there is a shared need to automatically derive a simplified representation or coarse-grained perspective of the interactions among components within a complex system. This task is commonly referred to as community detection in networks, akin to the search for clusters in independent vector data. Real-world graphs often include attributes assigned to their nodes, leading to a natural expectation that the identified communities should exhibit semantic coherence, with nodes within a community sharing similarities in terms of their attributes. Simultaneously, there is an additional expectation for these communities to display density and high modularity. However, these two objectives may not consistently align due to the inherent properties of the network. This introduces the question of exploring the trade-off between the quality of clusters in terms of semantics and structural aspects. In this study, we address this issue by working with both synthetic and real-world citation graphs. We have developed a benchmarking tool for evaluation and curated a dataset. Furthermore, we have compared the performance of three distinct approaches, namely Leiden [14] and two Graph Neural Network (GNN) methods, GCC [3] and GIC [6], on these datasets. Our findings provide valuable insights into the intrinsic trade-off between semantic and structural cluster qualities. We put one step forward, and investigate this trade-off in case of adding pseudo edges to similar nodes. We also evaluate the feasibility of controlling this trade-off by adding a regularization term to the objective function of GIC.

1 RELATED WORKS

1.1 Metadata-based Community Detection

In the context of evaluating community detection algorithms based on metadata, [11] presents an argument for the utility of incorporating node metadata when assessing community detection algorithms in attributed graphs. This work introduces two distinct tests to appraise the significance and the strength of the correlation between each attribute and the identified communities. In our own research, our approach differs slightly, as we acknowledge the importance of cluster semantic coherence while assuming that the user possesses this awareness.

1.2 Community Detection in Heterophily Graphs

In the rich history of neural-network-based community detection algorithms, few of them has been designed to capture the heterophily property in graphs. Instead of the homophily assumption in most GNNs, heterophily observed the fact that some nodes could be connected because they are different. As far as we know, no previous work really focus on the community detection problem in heterophily graphs, although some unsupervised/self-supervised homophily- and heterophily-aware embedding learning techniques can be applied to community detection.

Some previous work defined heterophily based on node-level homophily ratio ([15], [12]) or edge-level homophily ratio ([17], [16]), which is homophily ratio of existing edges on node-level or edge-level. [15] designed low-pass filter and high-pass filter and tried to maximize the agreement between low-pass filter and high-pass filter, where high-pass filter makes the representations of nodes that have different features from their neighbors distinct in their multi-hop neighborhood, while low-pass filter (traditional GNNs) produces similar representations for the nodes within the same multi-hop neighborhood. [5] applied a similar strategy as [15], despite that it first discriminate edges as homophily or heterophily, and only pass the homophily view to low-pass filter and heterophily view to high-pass filter. The embeddings from two views are aggregated by concatenation. [17] observed the relation between heterophily-homophily and semantic-structure. They found that the performance of NE (Network Embedding) methods that utilize network structure decreases significantly when homophily ratio is close to zero, and the performance of a NE method that only relies on raw node attributes is not affected by changes of h, but it is outperformed by other NE methods when homophily ratio is close to one. Therefore it tried to leverage both benefit of structure and attribute by learning Node feature attribute embeddings and network structure embeddings separately. But they only concatenate the two generated embeddings for downstream tasks. [2] augmented original graph based on structure and attributes, then tried to maximize the mutual information between original features and generated embeddings. It considered the heterophily setting that nodes might be more similar to r-hop neighbors instead of direct neighbors.

2 DATASET

We evaluate on two datasets: A synthetic dataset of our own design, and a pre-processed citation graph.

2.1 Synthetic Dataset

To create this graph, we generate a graph using the stochastic block model, then assign attributes based on applying noise to a attribute-base for each cluster.

First, we create six attribute-bases of 2D vectors evenly spaced around the unit circle. We then expand the dimensionality by repeating the vector 10 times then normalizing it so that it is a 20D vector.

Each of the ten clusters of the synthetic graph are assigned one attribute-base with two attribute-base getting assigned to three clusters. Each cluster has between 150 to 250 nodes with edges between each pair formed with probability between 0.05 to 0.1.

Within each cluster, each node is assigned an attribute which is the normalized sum of the attribute base vector and a vector of Gaussian noise drawn from $\mathcal{N}(0, 0.02)$.

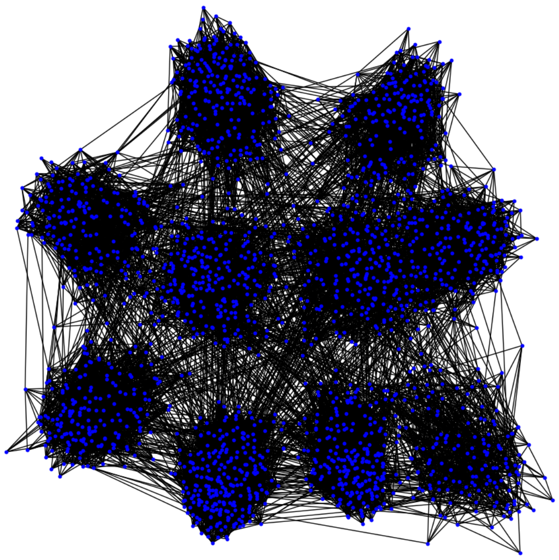


Figure 1: Visualization of the synthetic dataset.

Finally, we add inter-cluster edges. For each possible pairing between nodes of two clusters, the edge exists with probability 0.0005 except for the two groups of three clusters with the same attribute-base. For one group, the probability of edge existing is 0.0009 and for the other group, the probability is 0.001.

A visualization of the final graph is presented in Fig. 1 and ground truth when having # of cluster = 6 is presented in Fig. 2. This graph has 2001 nodes and 16264 edges.

2.2 Citation Graph

This is the Arxiv HEP-TH (high energy physics theory) citation graph [4]. It has 27770 nodes and 352807 edges where each node represents a paper in the HEP-TH section of Arxiv between January 1993 to April 2003 (124 months) and each directed edge goes from the paper that cites to the paper it cites.

The node attribute is a vector embedding of the paper's abstract. We elected to use SGPT [7] to perform the text embeddings because it had the highest overall ranking in Massive Text Embedding

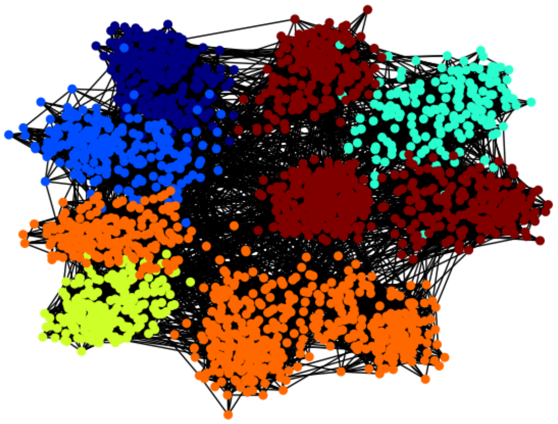


Figure 2: Ground truth of synthetic dataset with #cluster = 6.

Benchmark (MTEB) [8] that can process the entire abstract. The maximum number of tokens in an abstract is 874. This model can handle 2048 input tokens. The resulting vector has a length of 4096.

3 BASELINES

3.1 Leiden

Leiden [14] is proposed to guarantee well-connected communities. The authors observed that for both modularity [10] and CPM [13], they can not reflect the connectivity within communities, and many previous community detection algorithms may yield arbitrarily badly connected communities. To tackle this problem, Leiden is built upon Louvain [1] with three steps: local moving of nodes; refinement of the partition; aggregation of the network based on the refined partition whereas the refinement phase requires that a node is merged with a community in refined partition only if both are sufficiently well-connected to their community in original partition so that a well-connected community partition can be guaranteed.

3.2 GCC

GCC [3] designs an objective function that considers both representation learning and clustering tasks. By extracting convolved node representations, and calculating cluster memberships and cluster centroids based on representations, it tries to minimize the difference between the convolved node representations and their reconstructed cluster representatives. The cluster centroids, convolutional network, and cluster memberships are updated alternatively until convergence.

3.3 GIC

GIC [6] proposes a new unsupervised node representation learning method for graphs called Graph InfoClust (GIC). It uses two objective functions:

- Local objective: Maximizes the mutual information between each node's representation and a coarse-grain summary derived from the clusters that node belongs to. This captures local cluster-level similarities.

- Global objective: Maximizes the MI between each node's representation and the global graph summary. This captures global graph-level properties.

The clusters and cluster assignments are computed via differentiable soft clustering based on a modified k-means algorithm that clusters nodes based on their learned representations. This allows end-to-end joint optimization of clusters and representations. A contrastive learning approach is used where the model tries to discriminate between representations computed from the real graph versus a corrupted "fake" graph. This helps the model learn useful representations. The two objectives allow the model to capture both local cluster-level and global graph-level structures. Jointly optimizing clusters and representations helps improve both. And the contrastive learning framework helps learn more useful representations.

4 PROPOSED METHODS

4.1 Leiden with weighted edges

To make the Leiden algorithm take into account the semantic similarities, we add the semantic similarity between the two nodes connected by the edge to the weight of the edge (which is 1 because the edges are unweighted in these datasets) following the definition in Eq. 1. We then scale this by multiplying all weights by 100. We find this to improve performance compared to not scaling the weights.

4.2 Manipulating input adjacency matrix

To emphasize more on semantic similarity, we propose a pre-training strategy of adding(deleting) pseudo edges that aims to fuse both topology information and semantic information into the adjacency matrix. The intuition is quite simple: if two nodes are highly semantically similar but not connected, we will assume that they incline to belong to a same cluster, thus a pseudo-edge is added between them to strengthen their correlation; if two nodes are semantically very different but connected, we will assume that they might be connected due to some error, then we delete the edge between them. However, depending on the dataset, sometimes one of the assumptions might not hold true, and it might be more helpful to apply either addition or deletion. For example, pseudo edge deletion will be more helpful for noisy datasets, whereas pseudo edge addition will be more helpful if semantic similarity is positively correlated with clustering objective.

Given input adjacency matrix $\mathcal{A}$, node features $\{f_i\}$, upper threshold of semantic similarity β_U and lower threshold of semantic similarity β_L, output adjacency matrix with pseudo edges $\mathcal{A}^*$ can be formulated as

$$\mathcal{A}^*[i,j] = (\mathcal{A}[i,j] \& (\cos(f_i, f_j) > \beta_L)) \vee ((\neg \mathcal{A}[i,j]) \& (\cos(f_i, f_j) > \beta_U))$$

where $\mathcal{A}[i,j]$ denotes whether node i and j are connected. The time complexity of generating $\mathcal{A}^*$ through above formula is $O(N^2 d)$ (N is the number of nodes, d is the dimension of node features), which takes approximately 2 minutes for the citation graph we use for testing. However, for larger graphs with millions of nodes, random sampling might be needed to control time complexity.

4.3 Manipulating objective function

On the big picture, the GIC's initial objective function is defined as $\mathcal{L} = \alpha\mathcal{L}_1 + (1-\alpha)\mathcal{L}_C$ where $\mathcal{L}_1$ denotes the global summary loss, and the $\mathcal{L}_C$ discriminates between real and fake samples within each cluster, and α is a hyperparameter. We refer to this as original loss as $\mathcal{L}_{topo}$. To address the semantic-topological trade-off of GIC, we add a regularizer term to $\mathcal{L}_{topo}$. That is, at each epoch, once the soft clustering results are available, we obtain hard clustering by defining the assigned hard cluster of node k as $c = \underset{j}{\operatorname{argmax}}\, P_k(j)$, where $P_k(j)$ is the probability of node k being assigned to cluster j. Also, let $P_k = \underset{j}{\operatorname{amax}}\, P_k(j)$ be the highest clustering score of node k.

Once these hard clusters are obtained per each epoch, we sample a set of nodes from each cluster. Then, we use our prior definition of intra-cluster and inter-cluster similarity to obtain such similarities per our sampled nodes. We define our intra-cluster loss as:

$$\mathcal{L}_{intra} := -\sum_{c \in C} \sum_{i,j \in c} P_i P_j \log(f_i . f_j)$$

Where:

- C is our clusters
- f_i is the feature vector of node i

The intuition is that by minimizing $\mathcal{L}_{intra}$, for nodes i, j in the same cluster, $P_i.P_j$ would match the cosine similarity of their feature vectors.

We similarly define the inter-cluster loss as:

$$\mathcal{L}_{inter} := \sum_{c_1, c_2 \in C, c_1 \neq c_2} \sum_{i \in c_1} \sum_{j \in c_2} P_i P_j \sigma(f_i . f_j)$$

Where σ is the sigmoid function.

The intuition is that by minimizing $\mathcal{L}_{inter}$, for nodes i, j in the different clusters, high feature similarity would penalize a high $P_i P_j$.

With all three loss terms defined, we now define the overall loss as:

$$\mathcal{L}_{\text{total}} = r_{\text{topo}} \mathcal{L}_{\text{topo}} + (1 - r_{\text{topo}}) \left(r_{\text{intra}} \mathcal{L}_{\text{intra}} + (1 - r_{\text{intra}}) \mathcal{L}_{\text{inter}} \right)$$

Where r_{topo} and r_{intra} are the relative importance of the topological loss and intra-cluster loss, respectively.

5 METRICS

We evaluate the algorithms on four metrics: normalized mutual information (NMI), modularity, inter-cluster similarity, and intra-cluster similarity.

5.1 Normalized Mutual Information

Normalized Mutual Information (NMI) is the normalized form of mutual information (MI), a symmetric metric to quantifying the mutual dependence between the two variables. It is defined as:

$$I(X;Y) := \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

Table 1: Synthetic Experiment Results. Column in gray denotes lower values is better.

Methods	Dataset	Model	# of Clusters	Inter-Cluster Similarity	Intra-Cluster Similarity	Modularity	NMI
Vanilla	Synthetic (10 clusters)	Leiden (Modularity)	10	0.022 ± 0.0	0.992 ± 0.0	0.82780	1.0
Pseudo-Edges	Synthetic (10 clusters)	Leiden (Modularity)	10	0.022 ± 0.001	0.992 ± 0.0	0.82780	1.0
Weighted Edges	Synthetic (10 clusters)	Leiden (Modularity)	10	0.022 ± 0.001	0.992 ± 0.0	0.827805	1.0
Vanilla	Synthetic (10 clusters)	Leiden (CPM)	2001	0.0 ± 0.0	0.0 ± 0.0	-0.000557	0.463904
Pseudo-Edges	Synthetic (10 clusters)	Leiden (CPM)	2001	0.0 ± 0.0	0.0 ± 0.0	-0.000557	0.463904
Weighted Edges	Synthetic (10 clusters)	Leiden (CPM)	10	0.022 ± 0.001	0.992 ± 0.0	0.827805	1.0
Vanilla	Synthetic (6 clusters)	GCC	6	-0.198 ± 0.001	0.992 ± 0.001	0.713883	0.830619
Pseudo-Edges	Synthetic (6 clusters)	GCC	6	-0.199 ± 0.002	0.993 ± 0.0	0.714280	0.832397
Vanilla	Synthetic (10 clusters)	GCC	10	-0.11 ± 0.001	0.991 ± 0.003	0.538608	0.747062
Pseudo-Edges	Synthetic (10 clusters)	GCC	10	0.022 ± 0.001	0.993 ± 0.0	0.827805	1.0
Vanilla	Synthetic (6 clusters)	GIC	6	0.087 ± 0.081	0.187 ± 0.096	0.468920	0.554328
Pseudo-Edges	Synthetic (6 clusters)	GIC	6	0.123 ± 0.039	0.246 ± 0.087	0.369697	0.532149
Semantic Loss	Synthetic (6 clusters)	GIC	6	0.098 ± 0.035	0.168 ± 0.107	0.464257	0.553394
Vanilla	Synthetic (10 clusters)	GIC	10	0.077 ± 0.04	0.213 ± 0.12	0.366612	0.509225
Pseudo-Edges	Synthetic (10 clusters)	GIC	10	0.109 ± 0.021	0.196 ± 0.055	0.213982	0.473718
Semantic Loss	Synthetic (10 clusters)	GIC	10	0.108 ± 0.044	0.089 ± 0.059	0.358271	0.506893

To normalize MI, NMI depends on entropy which is defined as:

$$H(X) := -\sum_{x \in \mathcal{X}} p(x) \log(p(x))$$

The NMI between two variables is thus calculated using the following formula:

$$NMI(X,Y) := \frac{I(X;Y)}{\frac{1}{2}H(X) + H(Y)}$$

Where:

- $I(X;Y)$ represents the MI between the two variables.
- $H(X)$ and $H(Y)$ denote the entropy of variables X and Y respectively.

NMI is bounded between 0 and 1 where 0 indicates there is no mutual information between the two variables and 1 indicates there is a perfect correlation.

5.2 Modularity

Modularity [10] measures the strength of division of the communities in a graph. Graphs with high modularity have dense connections between nodes within modules but sparse connections between nodes in different modules [9]. It can be calculated as:

$$Q := \frac{1}{2|\mathcal{E}|} \sum_{ij} \left(\mathcal{A}_{ij} - \gamma \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

Where:

- $|\mathcal{E}|$ represents the number of edges
- $\mathcal{A}$ denotes the adjacency matrix of the graph
- k_i is the degree of node i
- γ is the resolution parameter (which we set to 1)
- $\delta(c_i, c_j)$ is an indicator variable that is 1 if nodes i and j belong to the same community and 0 if not.

Table 2: Citation Experiment Results. Column in gray denotes lower values is better.

Methods	Dataset	Model	# of Clusters	Inter-Cluster Similarity	Intra-Cluster Similarity	Modularity
Vanilla	Citation	Leiden (Modularity)	173	0.013 ± 0.0	0.068 ± 0.0	0.659101
Pseudo-Edges	Citation	Leiden (Modularity)	140	0.013 ± 0.0	0.068 ± 0.0	0.659101
Weighted Edges	Citation	Leiden (Modularity)	174	0.014 ± 0.0	0.082 ± 0.0	0.664400
Vanilla	Citation	Leiden (CPM)	27770	-	-	-0.000040
Pseudo-Edges	Citation	Leiden (CPM)	27770	-	-	-0.000040
Weighted Edges	Citation	Leiden (CPM)	1160	0.025 ± 0.0	0.123 ± 0.001	0.613912
Vanilla	Citation	GCC	173	0.005 ± 0.0	0.04 ± 0.0	0.413184
Pseudo-Edges	Citation	GCC	173	0.016 ± 0.0	0.086 ± 0.002	0.428722
Vanilla	Citation	GCC	21	0.165 ± 0.008	0.283 ± 0.009	0.183318
Pseudo-Edges	Citation	GCC	21	0.071 ± 0.004	0.192 ± 0.005	0.079040
Vanilla	Citation	GIC	173	0.546 ± 0.001	0.6 ± 0.005	0.080100
Pseudo-Edges	Citation	GIC	173	0.565 ± 0.0	0.611 ± 0.001	0.101749
Semantic Loss	Citation	GIC	173	0.563 ± 0.0	0.615 ± 0.003	0.113006
Vanilla	Citation	GIC	21	0.566 ± 0.001	0.58 ± 0.003	0.071996
Pseudo-Edges	Citation	GIC	21	0.568 ± 0.001	0.578 ± 0.002	0.073583
Semantic Loss	Citation	GIC	21	0.567 ± 0.001	0.589 ± 0.002	0.114889

5.3 Similarity

To compute similarity, we first define similarity S between two vectors $v_1,\ v_2$ as:

$$S := \frac{v_1 v_2}{\|v_1\|\|v_2\|} \tag{1}$$

Then, to get the inter- and intra- cluster similarities, we find the mean of a sample of the nodes.

5.3.1 Inter-Cluster. For each pair of clusters, we select 10% of the nodes in each. Then, we calculate the similarity between each of the selected nodes in the first cluster with each of the selected nodes of the second cluster. This is averaged out and the average of each average for each pair of clusters is the inter-cluster similarity.

5.3.2 Intra-Cluster. This is similar to the way we calculate the inter-cluster similarity. However, in this case, we select 10% of the nodes in the cluster and compute the similarities among all pairs of the nodes. The average of this is the intra-cluster similarity.

6 EXPERIMENTAL RESULTS

6.1 Baselines' performance

The results for running GIC, GCC and Leiden on synthetic dataset and citation network are shown in Table 1 and Table 2 respectively (NMI on citation network is not tested due to missing ground truth and Leiden (CPM) does not include the inter- and intra- cluster similarity measures because the large number of groups make it infeasible). We tested GCC on synthetic dataset with number of clusters hyperparameter set to 6 and 10.

On both synthetic and citation network, GCC tends to emphasize the intra-cluster similarity but leading to poorer inter-cluster similarity and modularity compared with Leiden. This could be partially resulted from the fact that GCC requires number of clusters hyperparameter and is sensitive to the hyperparameter setting. Moreover, nodes in our selected citation network are all paper in the HEP-TH section, which might share closer semantic information. Therefore, if we do higher order propagation on graphs in GCC, it will concentrate more on structural information while semantic information are mixed up in representations between neighbors. This neglection of fine-grained discrimination between attributes will fail to distinguish and properly cluster the nodes. For GIC, it is outperformed by both GCC and Leiden in terms of modularity and NMI. It is also outperformed by intra- and inter- cluster similarity except on citation graph that it sacrifice inter-cluster similarity to achieve higher intra-cluster similarity. The failue of GIC on synthetic graph reflect the fact that it is capturing more semantic information and less structural information as synthetic graph is easy to cluster on structural aspect.

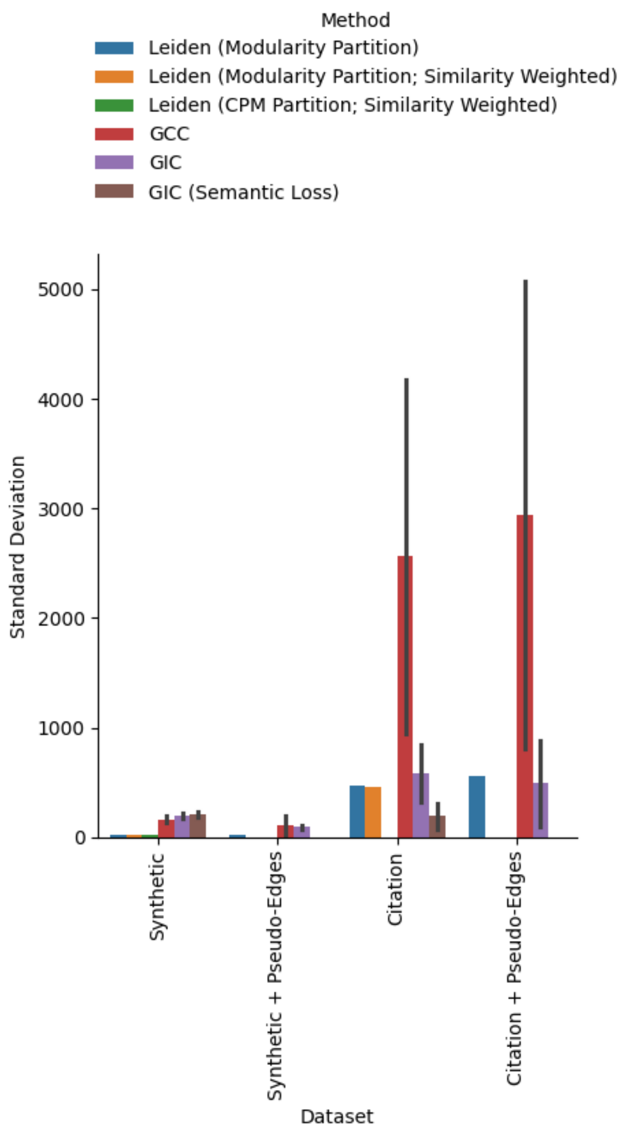


Figure 3: Cluster size comparison

To sum up, it's vital to properly balance the intrinsic trade-off between semantic and structural cluster qualities, and this trade-off can be graph-specific.

6.2 Leiden with edge weights

6.2.1 Leiden (Modularity). Leiden with modularity parition has good initial performance and modularity does not change much with the application of the methods proposed in this work. The metrics are all the same in the synthetic dataset variations, which may be due to it being a very simplistic dataset. However, on the citation dataset, inter-cluster similarity is reliablely improved although only by a small amount. Furthermore, intra-cluster similarity is reduced more significantly except for the weighted edges method, particularly for the undirected graph.

6.2.2 Leiden (CPM). Leiden with CPM partition has poor performance on the synthetic dataset. Many clusters are created, resulting in 0 for both inter-cluster similarity and intra-cluster similarity and poor modularity. When the edges are weighted, partitioning is much improved and the number of clusters shrinks dramatically from 2001 to 10 although intra-cluster similarity is very high.

A similar trend occurs on the citation dataset where the vanilla model creates a large number of clusters, pseudo-edges has similar performance, and weighted edges dramatically cuts down the number of clusters. However, in this case, performance on the directed and undirected citation graph is similar with the undirected graph being slightly better.

6.2.3 Advantage of edge weights. From the two variations of Leiden on the two datasets, it is clear the modified edge weights that take into account the semantic similarity is helpful in segmenting the graph. The additional information these weights provide allows the algorithm to better cluster the graph.

6.3 Manipulating input adjacency matrix

We conducted experiments with GIC and GCC on both datasets to test the effectiveness of manipulating input adjacency matrix. The hypeparameter setting for generating two datasets with pseudo edges are:

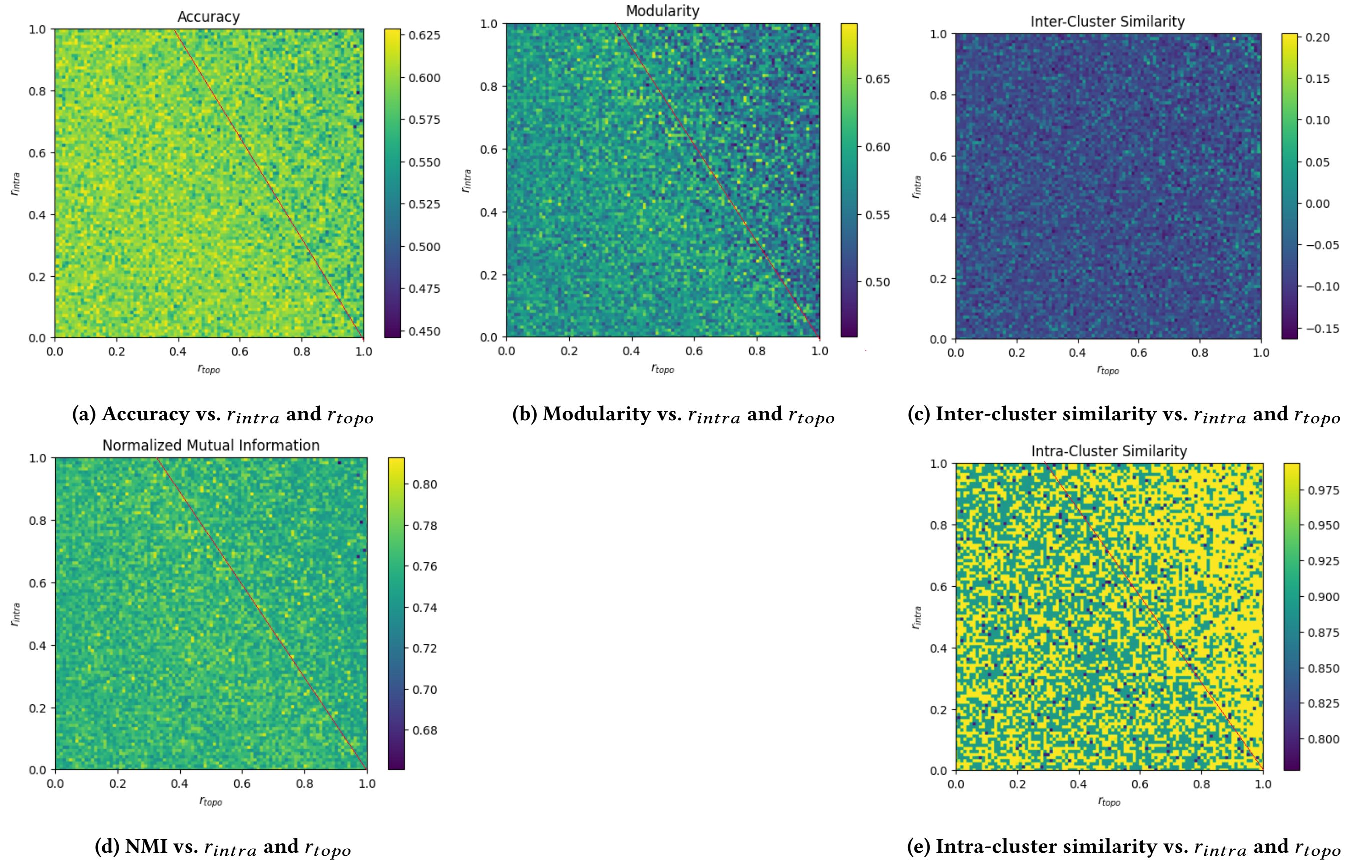
- Synthetic: $\beta_L = 0.6, \beta_U = 0.9999$
- Citation: $\beta_L = 0, \beta_U = 0.85$

Note that we only applied edge addition in terms of citation graph as we found that edge deletion will actually degrade the performance. This can be interpreted as difference in papers' abstracts does not necessarily denote irrelevance between papers.

By comparing results of 'Vanilla' and 'Pseudo-edges' on GCC and GIC in Table 1 and 2, we can observe that adding (and deleting) edges helps to improve modularity of GCC, except in the case of citation graph with # of clusters = 21. It also improve the NMI on GCC for synthetic dataset. For GIC, although adjacency matrix with pseudo edges help to increase modularity for citation graph, it also degrade the performance on synthetic graph. By observing the intra-cluster similarity and inter-cluster similarity, we can conclude that edge addition might have caused the increase in inter-cluster similarity on original graph, which lead to the result. Generally, this method might require model-specific hyperparameter tuning as different methods cluster nodes following different objectives. In our experiments, we did the hyperparameter tuning on GCC with 173 clusters on citation graph and 10 clusters on synthetic graph. Hyperparameter tuning is conducted through grid search, the search space of β_L is $[0.4, 0.7] \cup \{0\}$, step size is 0.1; search space of β_U is $[0.7, 1(0.9999)]$, step size is 0.05.

6.4 Manipulating objective function

In this experiment, we tried different values of r_{topo} and r_{intra} to see its effect on modularity, intra-cluster, and inter-cluster similarity metrics, as a way to measure the efficacy of our proposed regularization in addressing the semantic-topological trade-off in clustering. We only used the synthetic (6 cluster) graph for this experiment, and we conducted a grid search for intervals of hyperparameters spaced uniformly by increments of 0.01 in the interval



(a) Accuracy vs. r_{intra} and r_{topo}

(b) Modularity vs. r_{intra} and r_{topo}

(c) Inter-cluster similarity vs. r_{intra} and r_{topo}

(d) NMI vs. r_{intra} and r_{topo}

(e) Intra-cluster similarity vs. r_{intra} and r_{topo}

Figure 4: Evaluating semantic-topological trade-off of GIC clustering. Referring to (a), (b), and (d), we can see generally higher r_topo and higher r_intra would lead to better topological metrics, as can be seen in the region above the red line.

$[0, 1]$. The result of this experiment can be seen on Figure 4. Referring to Figure 4 (a), it can be seen that the upper-right part of the graph, above the red line, leads to higher accuracies. While it is evident that higher r_{topo} would lead to higher accuracies, interestingly, for the moderate values of r_{topo}, higher r_{intra} would lead to better accuracy. The same can be seen in Figure 4 (b), which could be expected since our synthetic graph has pre-defined ground truth clusters which high accuracies would correspond to higher modularity. No pattern can be observed in Figure 4 (c). This might be due to the fact that for the specific synthetic graph, different clusterings would result in almost the same inter-cluster similarity results. The same explanation can be made in Figure 4 (d) since intuitively NMI denotes equivalent topological metrics as accuracy and modularity in our synthetic graph. Finally, we observe the same trend in Figure 4 (e) as well. Overall, while these observations are noisy, they confirm that higher importance of topological loss or moderate value of topological loss combined with higher values of intra-cluster similarity would lead to better clustering results. The only exception was inter-cluster similarity, which could be specific to our graph.

6.5 Cluster size distribution

Fig. 3 compares the variance in cluster sizes of each approach we tried. The figure only compares for methods that generate less than or equal to 174 clusters since the larger clusters (1160 clusters and above) are significantly larger and results in much smaller standard deviations.

Comparing the methods, what immediately stands out is that GCC has very high variance in cluster sizes. Comparing between synthetic datasets and the citation datasets, variance is higher for the citation dataset. This could be attributed to the citation dataset, a real-world dataset, being a more difficult clustering problem than our synthetic dataset. One thing that is interesting is that GIC with our modified semantic loss has a slightly higher variance in cluster sizes in the synthetic dataset compared to the base GIC model, but a significantly lower variance in the citation dataset. Meanwhile, the Leiden-based approaches all presult in similar variance in cluster sizes. On the easier synthetic dataset, the cluster sizes are all about the same whereas on the citation dataset, there is consistently a large variance.

7 CONCLUSION

In this work, we emphasize the importance of finding a balance between the trade-off of structural and semantic information in community detection (clustering) problem. We proposed three methods to manipulate this balance through adding edge weights on Leiden, adding/deleting edges on GIC and GCC, and adding intra- and inter- cluster semantic similarity regularization term to the objective function of the GNN methods. We also performed extensive experiments on two datasets to examine the effectiveness of our proposed methods and further investigate the semantic-topological trade-off.

8 TEAM MEMBER ROLES

- Hossein Arjomandi (hm31)
- Eunice Chan (ecchan2)
- Xinyu He (xhe34)

8.1 Midterm

Hossein created the code to perform benchmarking and created the synthetic dataset. He also integrated the Leiden algorithm within our framework and ran it. Xinyu integrated the GCC model within our framework and ran it. Eunice integrated the MCGC model but due to technical difficulties, we do not have results for that model for this report. She also preprocessed the citation graph dataset.

For the midterm report, Hossein wrote the abstract and part of the related works. Xinyu wrote the majority of the related works, the baselines section, and the experiment results. Eunice wrote the datasets section, the metrics section, and the roles section. Everyone worked on the plan for the remaining milestones section.

8.2 Final

Hossein integrated the GIC model within our framework. With some discussions with Xinyu, he also modified the objective function to control the semantic-topological trade-off in the GIC results. He wrote the corresponding parts of the report as well. He also provided an explanation for the trade-off evaluation results on the synthetic graph.

Eunice wrote the following parts of the report and worked on the corresponding tasks: Sec. 4.1 content, Table 1 content, Table **??** content, Sec. 6.2 content, Sec. 6.4 heatmaps and associated content, Sec. 6.5 content. She ran and implemented the Leiden algorithm with the modularity partition and CPM partition as well as the variant with weighted edges. She performed comparative analysis and benchmarking of the combinations of methods, dataset, and models. She ran, performed, and visualized analysis on cluster sizes of the different approaches evaluated in the paper and visualized the relationship between the semantic hyperparameters in the objective function and the evaluated metrics.

Xinyu wrote/rewrote section 4.2, 6.1, 6.3, 7. She ran all experiments of GCC and all experiments of GIC on citation graph. She also designed and implemented the method of manipulating input adjacency matrix, and she also did the hyperparameter tuning for this method and provided the adjacency matrix with pseudo edges for other team members to test with.

REFERENCES

[1] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 10 (oct 2008), P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008

[2] Jingfan Chen, Guanghui Zhu, Yifan Qi, Chunfeng Yuan, and Yihua Huang. 2022. Towards Self-Supervised Learning on Graphs with Heterophily *(CIKM '22)*. Association for Computing Machinery, New York, NY, USA, 201–211. https://doi.org/10.1145/3511808.3557478

[3] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. 2022. Efficient Graph Convolution for Joint Node Representation Learning and Clustering. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (Virtual Event, AZ, USA) *(WSDM '22)*. Association for Computing Machinery, New York, NY, USA, 289–297. https://doi.org/10.1145/3488560.3498533

[4] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[5] Yixin Liu, Yizhen Zheng, Daokun Zhang, Vincent CS Lee, and Shirui Pan. 2023. Beyond smoothing: Unsupervised graph representation learning with edge heterophily discriminating. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 4516–4524.

[6] Costas Mavromatis and George Karypis. 2021. Graph infoclust: Maximizing coarse-grain mutual information in graphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 541–553.

[7] Niklas Muennighoff. 2022. SGPT: GPT Sentence Embeddings for Semantic Search. *arXiv preprint arXiv:2202.08904* (2022).

[8] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. MTEB: Massive Text Embedding Benchmark. *arXiv preprint arXiv:2210.07316* (2022). https://doi.org/10.48550/ARXIV.2210.07316

[9] Mark Newman. 2011. *Networks: An Introduction.* Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199206650.001.0001

[10] Mark Newman, Mark Newman, Michelle Girvan, and Michelle Girvan. 2003. Finding and evaluating community structure in networks. *Physical review. E, Statistical, nonlinear, and soft matter physics* 69 2 Pt 2 (2003), 026113. https://api.semanticscholar.org/CorpusID:169860743

[11] Leto Peel, Daniel B Larremore, and Aaron Clauset. 2017. The ground truth about metadata and community detection in networks. *Science advances* 3, 5 (2017), e1602548.

[12] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287* (2020).

[13] Vincent A Traag, Paul Van Dooren, and Yurii Nesterov. 2011. Narrow scope for resolution-limit-free community detection. *Physical Review E* 84, 1 (2011), 016114.

[14] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific reports* 9, 1 (2019), 5233.

[15] Wenhan Yang and Baharan Mirzasoleiman. 2023. Contrastive Learning under Heterophily. *arXiv preprint arXiv:2303.06344* (2023).

[16] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. 2022. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082* (2022).

[17] Zhiqiang Zhong, Guadalupe Gonzalez, Daniele Grattarola, and Jun Pang. 2022. Unsupervised network embedding beyond homophily. *arXiv preprint arXiv:2203.10866* (2022).