Joint Learning of Hierarchical Community Structure and Node Representations: An Unsupervised Approach

Ancy Sarah Tom[1] ✉, Nesreen K. Ahmed[2], and George Karypis[1]

[1] University of Minnesota, Twin Cities, MN
{tomxx030,karypis}@umn.edu
[2] Intel Labs, Santa Clara, CA
nesreen.k.ahmed@intel.com

Abstract. Graph representation learning has demonstrated improved performance in tasks such as link prediction and node classification across a range of domains. Research has shown that many natural graphs can be organized in hierarchical communities, leading to approaches that use these communities to improve the quality of node representations. However, these approaches do not take advantage of the learned representations to also improve the quality of the discovered communities and establish an iterative and joint optimization of representation learning and community discovery. In this work, we present *Mazi*, an algorithm that jointly learns the hierarchical community structure and the node representations of the graph in an unsupervised fashion. To account for the structure in the node representations, *Mazi* generates node representations at each level of the hierarchy, and utilizes them to influence the node representations of the original graph. Further, the communities at each level are discovered by simultaneously maximizing the modularity metric and minimizing the distance between the representations of a node and its community. Using multi-label node classification and link prediction tasks, we evaluate our method on a variety of synthetic and real-world graphs and demonstrate that *Mazi* outperforms other hierarchical and non-hierarchical methods.

Keywords: networks · network embedding· unsupervised learning· graph representation learning· hierarchical clustering· community detection

1 Introduction

Representation learning in graphs is an important field, demonstrating good performance in many tasks in diverse domains, such as social network analysis, user modeling and profiling, brain modeling, and anomaly detection [7]. Graphs arising in many domains are often characterized by a hierarchical community structure [13], where the communities (i.e., clusters) at the lower (finer) levels of the hierarchy are better connected than the communities at the higher (coarser) levels of the hierarchy. For instance, in a large company, the graph that captures the relations (edges) between the different employees (nodes) will tend to form communities at different levels of granularity. The communities at the lowest levels will be tightly connected corresponding to people that are part of the same team or project, whereas the communities at higher levels will be less connected corresponding to people that are part of the same product line or division.

In recent years, researchers have conjectured that when present, the hierarchical community structure of a graph can be used as an inductive bias in unsupervised node representation learning. This has led to various methods that learn node representations by taking into account a graph's hierarchical community structure. *HARP* [3] advances from the coarsest level to the finest level to learn the node representations of the graph at the coarser level, and then uses it as an initialization to learn the representations of the finer level graph. *LouvainNE* [1] uses a modularity-based [13] recursive decomposition approach to generate a hierarchy of communities. For each node, it then proceeds to generate representations for the different sub-communities that it belongs to. These representations are subsequently aggregated in a weighted fashion to form the final node representation, wherein the weights progressively decrease with coarser levels in the hierarchy. *SpaceNE* [11] constructs sub-spaces within the feature space to represent different levels of the hierarchical community structure, and learns node representations that preserves proximity between vertices as well as similarities within communities and across communities. Further, in recent times, certain GNN-based approaches [10, 18] have also been proposed which exploit the hierarchical community structure while learning node representations. However, these methods use supervised learning and require more information to achieve good results.

Though all of the above methods are able to produce better representations by taking into account the hierarchical community structure, the information flow is unidirectional—from the hierarchical communities to the node representations. We postulate that the quality of the node representations can be improved if we allow information to also flow in the other direction—from the node representations to hierarchical communities—which can be used to improve the discovered hierarchical communities. Moreover, this allows for an iterative and joint optimization of both the hierarchical community structure and the representation of the nodes.

We present *Mazi*[3], an algorithm that performs a joint unsupervised learning of the hierarchical community structure of a graph and the representations of its nodes. The key difference between *Mazi* and prior methods is that the community structure and the node representations help improve each other. *Mazi* estimates node representations that are designed to encode both local information and information about the graph's hierarchical community structure. By taking into account local information, the estimated representations of nodes that are topologically close will be similar. By taking into account the hierarchical community structure, the estimated representations of nodes that belong to the

[3] Mazi is Greek for together.

same community will be similar and that similarity will progressively decrease for nodes that are together only in progressively coarser-level communities.

Mazi forms successively smaller graphs by coarsening the original graph using the hierarchical community structure such that the communities at different levels represent nodes in the coarsened graphs. Then, iterating over all levels, *Mazi* learns node representations at each level by maximizing the proximity of the representation of a node to that of its adjacent nodes while also drawing it closer to the representation of its community. Furthermore, at each level, *Mazi* learns the communities by taking advantage both of the graph topology and the node representations. This is done by simultaneously maximizing the *modularity* of the communities, maximizing the affinity among the representations of near-by nodes by using a Skip-gram [12] objective, and minimizing the distance between the representations that correspond to a node and its parent in the next-level coarser graph.

We evaluate *Mazi* on the node classification and the link prediction tasks on synthetic and real-world graphs. Our experiments demonstrate that *Mazi* achieves an average gain of 215.5% and 9.3% over competing approaches on the link prediction and node classification tasks, respectively. The contributions of our paper are the following:

1. We develop an unsupervised approach to simultaneously organize a graph into hierarchical communities and to learn node representations that account for that hierarchical community structure. We achieve this by introducing and jointly optimizing an objective function that contains (i) modularity- and skip-gram-based terms for each level of the hierarchy and (ii) inter-level node-representation consistency terms.
2. We present a flexible synthetic generator for graphs that contain hierarchically structured communities and community-derived node properties. We use this generator to study the effectiveness of different node representation learning algorithms.
3. We show that our method learns node representations that outperform competing approaches on synthetic and real-world datasets for the node classification and link prediction tasks.

2 Definitions and Notation

Let $G = (V, E)$ be an undirected graph where V is its set of n nodes and E is its set of m edges. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ store the representation vector x_i at the ith row for $v_i \in V$. A *community* refers to a group of nodes that are better connected with each other than with the rest of the nodes in the graph. A graph is said to have a *community structure*, if it can be decomposed into communities. In many natural graphs, communities often exist at different levels of granularity. At the upper (coarser) levels, there is a small number of large communities, whereas at the lower (finer) levels, there is a large number of small communities. In general, the communities at the coarser levels are less well-connected than the finer level communities. When the communities at different levels of granularity

Table 1: Summary of notation.

Notation	Description
l	A level in the hierarchical structure.
L	The number of levels in the hierarchical communities.
G	The graph $G = (V, E, W)$, where V is the set of n nodes, E is the set of m edges, and W stores the edge weights.
v_i	A vertex in G.
$\deg(v_i)$	The degree of node v_i.
X	The node representations of G
$\mathbb{C}$	A community decomposition of G.
H	The community membership indicator vector of G.
C_i	A community in $\mathbb{C}$.
$\deg_{int}(C_i)$	The internal degree of community C_i, i.e, the number of edges that connect nodes in C_i to other nodes in C_i.
$\deg_{ext}(C_i)$	The external degree of community C_i, i.e, the number of edges that connect C_i to nodes in other communities.
$\deg(C_i)$	The overall degree of community C_i, i.e, the sum of $\deg_{int}(C_i)$ and $\deg_{ext}(C_i)$.
ID	An array containing the vertex internal degrees.
ED	An array containing the vertex external degrees.
Q	The modularity of G for a given $\mathbb{C}$ (cf. Eqn. 1).
G^l	The graph $G^l = (V^l, E^l, W^l)$ at level l
X^l	The node representations at level l.
H^l	The community structure at level l.
d	The dimension of X^l, where $l \in 1, \dots, L$.
ne^l	The number of epochs at level l.
lr^l	The learning rate at level l.
k	The context size extracted from walks.
wl	The length of random-walk.
r	The number of walks per node.
α	The weight of the contribution of node neighborhood to the overall loss.
β	The weight of the contribution of proximity to a node's community to the overall loss.
γ	The weight of the contribution of Q to the overall loss.

form a hierarchy, that is, a community at a particular level is fully contained within a community at the next level up, then we will say that the graph has a *hierarchical community structure*.

Let $\mathbb{C} = \{C_0, \dots, C_{k-1}\}$, with $V = \cup_i C_i$ and $C_i \cap C_j = \emptyset$ for $0 \le i, j < k$ be a k-way *community decomposition* of G with C_i indicating its ith community. Let H be the *community membership* indicator vector where $0 \le H[v_i] < k$ indicates v_i's community. Given a k-way community decomposition $\mathbb{C}$ of $G^l = (V^l, E^l)$, its *coarsened* graph $G^{l+1} = (V^{l+1}, E^{l+1})$ is obtained by creating k vertices—one for each community in $\mathbb{C}$—and adding an edge $(v_i, v_j) \in E^{l+1}$ if there are edges $(u_p, u_q) \in E^l$ such that $u_p \in C_i$ and $u_q \in C_j$. The weight of the (v_i, v_j) edge is set equal to the sum of the weights of all such (u_p, u_q) edges in E^l. In addition, each $v_i \in V^{l+1}$ is referred to as the parent node to all $u \in C_i$. Given $\mathbb{C}$, the *modularity* of G is defined as

$$Q = \frac{1}{2m}\left(\sum_{C_i \in \mathbb{C}} \left(\deg_{int}(C_i) - \frac{\deg(C_i)^2}{2m}\right)\right). \tag{1}$$

Q measures the difference between the actual number of edges within C_i and the expected number of edges within C_i, aggregated over all $C_i \in \mathbb{C}$. Q ranges from -0.5, when all the edges in G are between C_i and C_j, where $i \neq j$, and approaches 1.0 if all the edges are within any C_i and k is large.

Let the hierarchical community structure of G, with L levels, be represented by a sequence of successively coarsened graphs, denoted by $G, G^2, \cdots, G^L$, such that $|V| > |V^2| > \cdots > |V^L|$, wherein at each $l \in L$, the communities in G^l are collapsed to form the nodes in G^{l+1}. Every $v_i^l \in V^l$ is collapsed to a single parent node, v_j^{l+1}, in the next level coarser graph, G^{l+1}. Let us denote a model that takes the hierarchical community structure into account as hierarchical models and those that do not as flat models. Finally, we summarize all the notations in Table 1.

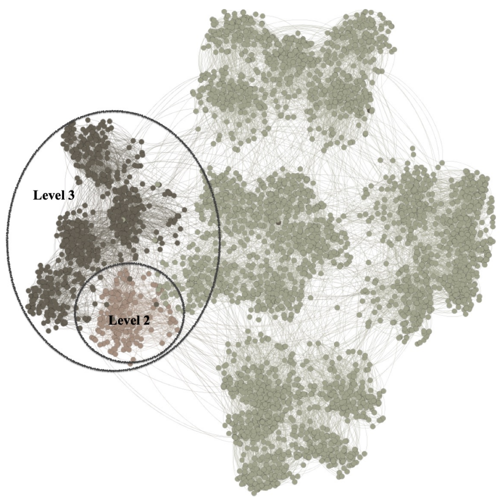


Fig. 1: A visualization of a synthetic 3K-node graph with a hierarchical community structure created by the proposed generator in Section 4. A common-ratio of 3.0 and a max. degree of 7.5 are used. A branching factor of 5 is used except at the finest level, which uses 30. Nodes in the hierarchical community structure are depicted using communities. A community in Level 3 and a sub-community in Level 2 are marked. A node's Level 1 community is itself.

3 *Mazi*

Given a graph G, *Mazi* seeks to jointly learn its node representations and its hierarchical community structure organized in L levels. *Mazi* coarsens the graphs at all levels of the hierarchy and learns representations for all nodes. At any given level, the node representation is learned such that it is similar to those of the nodes in its neighborhood, to its community and to the nodes it serves as a community to. This ensures the node representations at all levels align with the hierarchical community structure. Further, the communities at all levels are learned by utilizing node representations along with the graph topology. *Mazi* utilizes Skip-gram to model the similarity in the representations of a node and its neighbors. To model the similarity in the representations of node and its associated community, *Mazi* minimizes the distance between the respective two representations. Finally, to learn the communities, *Mazi* maximizes the *modularity* metric along with the above objectives.

Figure 1 illustrates a graph with a hierarchical community structure. From the figure, we see that the original graph (nodes of the graph is level 1 in the hierarchical structure) contains 5 large communities (level 3) in its coarsest level, each of which can be further split into 5 sub-communities (level 2). The nodes in the graph are marked such that the figure illustrates the level it belongs to in the hierarchical community structure. *Mazi* learns the representation of a node belonging to the level 2 community such that it will be similar to other nodes in that community over others. Furthermore, it will also be similar in representation to the nodes in its upper-level community at level 3, although this similarity value will be progressively lower as compared to that of the nodes in the level 2 community.

3.1 Objective Function

Mazi defines the objective function used for learning node representations using three major components. First, at each level, for each node, *Mazi* maximizes the proximity of its representation to the representation of the nodes belonging to its neighborhood using the Skip-gram objective. Second, iterating over all levels, the proximity of the representation of a node to that of its direct lineage in the embedding space is maximized. Third, the communities at each level are learned and refined by maximizing the *modularity* metric.

Modeling node proximity to its neighborhood. As previously studied, see [6], to capture the neighbourhood of a node in the representations, we seek to maximize the log-likelihood of observing the neighbors of a node conditioned on its representation using the Skip-gram model with negative sampling. Utilizing the concept of sequence-based representations, neighboring nodes of a node v_i, represented by $N(v_i)$, are sampled to form its context. Let the negative sampling distribution of v_i be denoted by P_n and the number of negative samples considered for training the loss be denoted by R. We use L_{nbr_pos} and L_{nbr_neg} to denote the loss of v_i to its neighbors and to its negative samples, respectively. Using the above, we define

$$L_{nbr\_pos} = \frac{1}{|N(v_i)|} \sum_{v_j \in N(v_i)} \log \sigma(x_i^\top x_j), \tag{2a}$$

$$L_{nbr\_neg} = R \cdot E_{v_n \sim P_n(v_i)} \log(1 - \sigma(x_i^\top x_j)). \tag{2b}$$

Taken together, we model the neighbourhood proximity of v_i as:

$$L_{nbr} = L_{nbr\_pos} + L_{nbr\_neg}. \tag{3}$$

Modeling node proximity to its community. In many domains, nodes belonging to a community tend to be functionally similar to each other in comparison to nodes lying outside the community [4]. As a consequence, we expect the representation of a node to be similar to the representation of its lineage in the hierarchy. Consider a level, l, in the hierarchical community structure of G. At l, for v_i^l, with representation x_i^l, we let the representation of its associated community (parent-node), $H^l(v_i^l)$, in the next level coarser graph, G^{l+1}, be denoted by $x^{l+1}_{H^l(v_i^l)}$. To model the relationship between v_i^l and $H^l(v_i^l)$, we use:

$$L^l_{comm} = \log \sigma\big({x_i^l}^\top x^{l+1}_{H^l(v_i^l)}\big). \tag{4}$$

As we iterate over the levels in the hierarchy of the graph, we bring together nodes in each level closer to its parent node in the next-level coarser graph in the embedding space. Consequently, the representation of a node is influenced by the communities the node belongs to at different levels.

Jointly learning the hierarchical community structure and node representations. Typically, community detection algorithms utilize the topological structure of a graph to discover communities. However, we may also take advantage of the information contained within the node representations while forming the communities at each level in the hierarchy. *Mazi* discovers the communities in the graph by jointly maximizing the *modularity* metric, described in Equation 1, at each level and minimizing the distance between the representations of a node and its community in the next level coarser graph. The communities that we learn at each level, thus, better align with the structural and the functional components of the graph at that level. At each level in the hierarchical community structure, we use Equation 3 and Equation 4 to model and learn the node representations.

Consequently, putting all the components together, we get the following coupled objective function:

$$\begin{gathered} \max_\theta \sum_{l=1}^{L} \left(\frac{1}{|V^l|} \Big(L^l_{nbr\_pos} + \alpha^l L^l_{nbr\_neg} + \beta^l L^l_{comm} \Big) + \gamma^l Q^l \right), \\ \theta = x_i^l, H^l,\ i \in 1 \dots |V|^l\ \forall l \in 1 \dots L. \end{gathered} \tag{5}$$

Since the order of the three terms that contribute to the overall objective is different, the terms are normalized with its respective order of contribution. Further, α^l, β^l and γ^l serve as regularization parameters and are added to Sub-equations (2b), (4) and (1) in the overall objective for each level l, respectively.

3.2 Algorithm

An initial hierarchical community structure of the graph at level 1, denoted by $G^1 = (V^1, E^1, W^1)$, is constructed and node representations are computed for all the levels in the hierarchy. Then, using an alternating optimization approach in a level-by-level fashion, the objective, defined previously, is optimized. The optimization updates step through the levels from the finest level graph to the coarsest level graph and then from the coarsest level graph to the finest level graph in multiple iterations. This enables the node representations at each level to align itself to its direct lineage in the embedding space, additionally refining the community structure by the information contained within this space. An outline of the overall algorithm and its complexity can be found in Algorithm 1, Appendix A.1 and Appendix A.3, respectively, in the supplementary materials.

Initializing the hierarchical community structure and node representations. A hierarchical community structure with L levels and their associated community membership vectors for G is initialized by successively employing existing community detection algorithms, such as *Metis* [9] at each level $l \in L$. The node representations at the finest level of the graph, denoted by X^1, are initialized by using existing representation learning methods such as *node2vec, DeepWalk* [6, 14]. Node representations of coarser level graphs are then initialized by computing the average of the representations of nodes that belong to a community in the previous level finer graph, G^{l-1}.

Optimization strategy. At each level, *Mazi* utilizes an alternating optimization (AO) approach to optimize its objective function. *Mazi* performs AO in a level-by-level fashion, by fixing variables belonging to all the levels except one, say denoted by l, and optimizing the variables associated with that level. At l, the community membership vector, H^l, is held fixed and the node representations, X^l, is updated. Then, X^l is fixed, and H^l is updated. Let us denote the node representation update as the X^l sub-problem, and the community membership update as the H^l sub-problem for further reference.

Node representation learning and community structure refinement. At each level l, *Mazi* computes the gradient updates for the X^l sub-problem. By holding H^l fixed, *Mazi* updates x_i^l to be closer to the representation of $v_j \in N(v_i^l)$, x_j^l, and its parent node, $x_{H^l(v_i^l)}^{l+1}$ (see Equation 3 and 4). The H^l sub-problem is then optimized using the updated X^l at l. To maximize the *modularity* objective (Q^l) in the H^l sub-problem, *Mazi* utilizes an efficient move-based approach. Consider reassigning v_i^l from its existing community C_a^l to a candidate destination community C_b^l. We note that Q^l, in Equation 1, depends on $\deg_{int}(C_i^l)$ and $\deg_{ext}(C_i^l)$, where $C_i^l \in \mathbb{C}$. Instead of computing the contribution of each community to determine Q^l, we only modify the internal and the external degrees of C_a^l and C_b^l by computing how the contribution of v_i^l to C_a^l and C_b^l changes. Utilizing this, the new community of v_i^l is determined such that it maximizes Q^l and minimizes the distance between x_i^l and $x_{H^l(v_i)}^{l+1}$.

After alternatively solving for the sub-problems X^l and H^l at level l, *Mazi* optimizes level $l+1$. These steps proceed up the hierarchy in this fashion until it reaches level $L-1$. Starting at $L-1$, the sub-problems X^{L-1} and H^{L-1} is optimized in the backward direction level-by-level using the updated representations, that is, $l = L-1, L-2, \ldots, 1$. By performing the optimization in the backward direction such as above, the node representations at the finer levels of the hierarchy are influenced by the updated representations at the coarser levels. After W such iterations, the refined node representations and community membership vectors for all levels are returned as the result of the algorithm.

4 Experiments

In order to evaluate the proposed algorithm, *Mazi*, we design experiments on real-world as well as synthetic graphs. We test *Mazi* on two major tasks: (i) link prediction and (ii) node classification. We compare *Mazi* against the following state-of-the-art baseline methods: (i) *node2vec* [6], a flat embedding model, (ii) *ComE* [2], a model respecting only a single-level in the hierarchy, (iii) *HARP* [3], (iv) *LouvainNE* [1], which are both hierarchical models, and (v) variations of the above mentioned models.

4.1 Experimental Setup

Link prediction task setup. We divide the original graph into validation (sample 5% of the edges) and test ((sample 10% of the edges)) sets, and train graph. For each positive sample (existent edge in the graph), we sample 99 negative samples (non-existent edges). We use the train graph to generate node representations. Then, for every edge in the validation and test sets, we compute its prediction score using the representations of the edge's node pairs along with that of its corresponding negative samples and determine the mean average precision. Further, to test our algorithm on link prediction using learnable decoders, we implement the *DistMult* model [16] and a 2-*layer multi-layer perceptron* (MLP). We provide the element-wise product of the representations of the nodes that comprise an edge as input to train the above models. We use 2% of the edges as the train set and 1% each for the validation and test set, with 20 negative samples for each positive edge, and report the average precision (AP) score of the test set for the best performing score on the validation set.

We run an elaborate search on the random-walk hyper-parameters. In *node2vec*, `context_size`, `walk_length`, `walks_per_node`, `p`, `q`, and #epochs select values between {2−5}, {4−10}, {5−60}, {0.1−10}, {0.1−10}, and, {1−4}, respectively. For *ComE* and *HARP*, the `context_size`, `walk_length` and `walks_per_node`

Table 2: Real-world graph dataset statistics. Experiments are conducted on the induced subgraph formed by the nodes in the largest connected component in the graph. Label rate is the fraction of nodes in the training set. The #communities in each coarsened level is equal to $\sqrt{n}$, where, n is the current level graph's number of nodes. We stop coarsening when #communities ≤ 10. The last level is the all-encompassing node.

Dataset	#nodes	#edges	#labels	#communities in coarsened levels	Label rate
BlogCatalog	10312	667966	39	$\{100, 10, 1\}$	0.17
CS-CoAuthor	18333	163788	15	$\{135, 12, 1\}$	0.08
DBLP	20111	115016	4	$\{142, 12, 1\}$	0.49

is chosen from $\{2-6\}$, $\{5-50\}$, and $\{5-30\}$, respectively. We choose parameters specific to *Mazi*, β and γ, from $\{0.25 - 2.5\}$ and $\{1.0 - 3.0\}$, respectively. We use the stochastic variation of *LouvainNE*, which is reported to obtain the best performance. We search all partitioning schemes of *LouvainNE*, used for generating the hierarchy, and use values $0.0001, 0.001, 0.01, 0.1, 1.0$ for the damping parameter. The #dimensions for all methods is 128.

Multi-label classification task setup. We use a One-vs-Rest Logistic Regression model (implemented using LibLinear [5]) with L2 regularization. We split the nodes in a graph (real-world and synthetic datasets) into train, validation and test sets. We sample a fixed number of instances, s, from each class to form a representative train set. The validation and the test set is, thereafter, formed by almost equally splitting the remaining samples. In Table 2, we detail the exact fraction of nodes (label rate) in the real-world graphs that were used to the train the model. In case of synthetic datasets, we choose 45 samples per class, leading to a label rate of 0.6. We choose the regularizer weight from the range {0.1, 1.0, 10.0}, such that it gives the best average macro F1 score on the validation set for the different methods. To generate the best performing model of the approaches for evaluation, we conduct a search over the different hyper-parameters for the synthetic and the real-world graphs.

For the synthetic graphs, in *node2vec*, `context_size`, `walk_length`, `walks_per_node`, and #epochs are chosen from $\{5, 10, 15\}$, $\{10, 20, 30\}$, $\{10, 20, 30\}$, and $1, 2$, respectively. `p` and `q` are chosen from $\{0.25 - 4\}$. *ComE*, *HARP* and *Mazi* also use the above for its parameters. *#clusters* in *ComE* has been chosen from $\sqrt{\#nodes}$ and *#labels*. Additionally, specific to *Mazi*, we choose both β and γ from $\{0.0, 1.0, 2.0\}$. For the real world graphs, the `context_size`, `walk_length`, and `walks_per_node` parameters have been varied between $\{5, 10, 15\}$, $\{10, 20, 30, 40\}$, and $\{10, 20, 30, 40\}$. `p` and `q` are chosen from $\{0.25, 0.50, 1, 2, 4\}$. The #dimensions for all methods is 128.

Table 3: Link prediction on real-world graphs. Link prediction task performance of the methods is listed in the table. All *HARP* variants use *node2vec* as the base model. The mean average precision score is reported. The results are the average of 3 runs. The observed standard deviation was less than 0.01.

	Mean Average Precision		
Method	BlogCatalog	CS-CoAuth	DBLP
node2vec	0.534	0.797	0.914
HARP w. 2 lvls	0.532	0.755	0.881
HARP w. 3 lvls	0.460	0.732	0.874
HARP w. all lvls	0.126	0.647	0.769
LouvainNE	0.035	0.270	0.397
ComE	0.389	0.745	0.896
Mazi	**0.587**	**0.824**	**0.930**

4.2 Evaluation

Real World Datasets. We evaluate the proposed algorithm on three real world networks: *BlogCatalog*, *CS-CoAuthor*, and *DBLP*. *BlogCatalog* is a social network illustrating connections between bloggers while *CS-CoAuthor* and *DBLP* are co-authorship networks. Both *DBLP* and *CS-CoAuthor* exhibit high values of *modularity*, that is, 0.83 and 0.75, respectively, while *BlogCatalog* has a relatively lower *modularity* value of 0.23. More information about each dataset is detailed in Table 2.

Evaluation on the link prediction task. Mazi demonstrates good performance over competing approaches as shown in Table 3 on the real-world datasets. We observe, in general, that *Mazi* demonstrates higher gains on datasets with higher *modularity* values. Over *node2vec*, the gains observed by *Mazi* in mean average precision (MAP) varies between 1.6% in *DBLP* to 10% in *BlogCatalog.* In comparison to *HARP*, referred to as *HARP w. all lvls* in Table 3, *Mazi* shows gains as high as 366% in *BlogCatalog.* To further study the behaviour of *HARP*, we evaluate its performance by restricting the total number of levels to 2 (*HARP w. 2 lvls*), and 3 (*HARP w. 3 lvls*). We note that both these approaches demonstrate higher MAP scores in comparison to *HARP*. We reason that since *HARP* collapses random edges and star-like structures to coarsen the graph in multiple levels, the coarsened graph in the last level may not be indicative of the global structure of the network and could serve as poor initializations. *Mazi* demonstrates gains between 3.79% and 50.89% against *ComE*. *ComE*, using gaussian mixtures to model its single-level communities, may not well capture the defining structural characteristics of the graph while generating representations. *LouvainNE*'s best performing version, as per the authors, uses random vectors for node representations at all levels in the graph's extracted hierarchy. Although LouvainNE may capture the hierarchical structure in a node's representation by performing a weighted aggregation of vectors belonging to its hierarchy, we

Table 4: Link prediction using learnable decoders on *BlogCatalog*. We report average precision score on link prediction task of the methods using **learnable decoders** - *DistMult* and 2-*layer multi-layer perceptron*. σ is short for the sigmoid function.

Method	Using σ	DistMult	2-layer MLP
node2vec	0.62	0.62	0.59
HARP w. 2 *lvls*	0.62	0.62	0.62
HARP w. 3 *lvls*	0.05	0.56	0.57
HARP w. all lvls	0.05	0.32	0.43
LouvainNE	0.07	0.08	0.12
ComE	0.47	0.47	0.46
Mazi	**0.70**	**0.70**	**0.69**

note that it may not well capture its local neighborhood. Thus, nodes that are in close proximity may not be represented similarly, and may indicate its low performance on the task. In Table 4, we report the average precision (AP) scores using learnable models, *DistMult* and a 2-*layer MLP*, on *BlogCatalog*. We note very similar trends as in Table 3 and observe that despite using learnable decoders, *Mazi* outperforms all other approaches in this task.

Evaluation on the multi-label node classification task. Table 5 reports micro and macro F1 score obtained by the methods on real-world datasets. We note that *Mazi* obtains a gain of up to 4.19% and 7.55% in macro F1 on *BlogCatalog* against *node2vec* and *HARP*, respectively. Against *LouvainNE*, *Mazi* achieves great gains on *BlogCatalog* (137%) and *CS-CoAuthor* (13%), respectively. While the gain obtained in *CS-CoAuthor* against *node2vec* and *HARP* is 0.43% and 0.85%, respectively, in macro F1, we observe that in *DBLP*, with a higher *modularity* value, the performance of *Mazi* is comparable with other approaches. *ComE* obtains a slightly better micro F1 score in *BlogCatalog*. Its choice of using gaussian mixtures to model community distributions appears to capture the weak community structure in *BlogCatalog* (*modularity* value of 0.23) well.

Synthetic Datasets. We design a novel synthetic graph generator that is capable of generating graphs with a hierarchical community structure and real-world structural properties. This is achieved by modeling the hierarchical community structure using a hierarchical tree (see Fig. 2). Each level in the hierarchical tree is a level in the hierarchical community structure, wherein the nodes of the tree forms the communities in the graph at that level. The nodes in the last level form the nodes of the generated graph. A node in the graph is generated such that, in expectation, it is able to form edges with other nodes in communities in the upper levels of the hierarchical community structure. For this, we accept a parameter, referred to as `common-ratio`, to generate L terms in geometric progression, for

Table 5: Multi-label node classification performance. Multi-label classification performance of the different methods are listed. The micro and macro F1 scores are reported. We report the scores achieved on the test set such that it achieves the best macro F1 score in the validation set chosen from the relevant hyper-parameters associated with each method. The results report the average of 3 runs. The standard deviation up to 2 decimal points is reported within the parentheses.

Method	BlogCatalog		CS-CoAuth		DBLP	
	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1
node2vec	0.3718 (0.00)	0.2430 (0.00)	0.8670 (0.00)	0.8213 (0.00)	0.2499 (0.00)	0.2314 (0.00)
HARP (n2v)	0.3602 (0.00)	0.2418 (0.00)	0.8634 (0.00)	0.8153 (0.00)	0.2515 (0.00)	0.2326 (0.00)
LouvainNE	0.2275 (0.00)	0.1051 (0.00)	0.7790 (0.00)	0.7317 (0.00)	**0.2578 (0.01)**	**0.2367 (0.01)**
Mazi	0.3874 (0.00)	**0.2499 (0.00)**	**0.8708 (0.00)**	**0.8266 (0.00)**	0.2510 (0.00)	0.2317 (0.00)
ComE	**0.4016 (0.00)**	0.2464 (0.00)	0.8696 (0.00)	0.8238 (0.00)	0.2517 (0.00)	0.2323 (0.00)

each level in the hierarchical community structure (refer to Appendix A.2 for detailed descriptions). With these terms, we compute a probability distribution for a node to form an edge with another. Higher values of the `common-ratio` result in fewer edges between nodes belonging to different communities and thus, increases *modularity* of the graph as computed by the communities in the second last level. Further, we use a power distribution to model the graph's node degrees to capture the behavior of real-world networks. Other properties that we tune are the maximum degree, number of levels, branching factor of nodes, number of leaves, among others. To aid us in the node classification task, we generate labels for nodes such that they correlate with the hierarchical structure of the graph. We discuss further details of the proposed generator in the Appendix A.2.

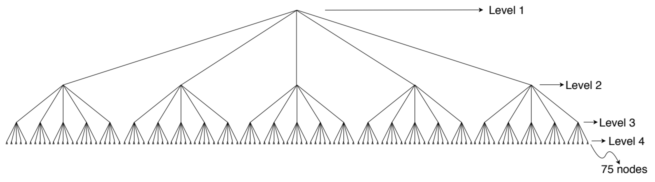
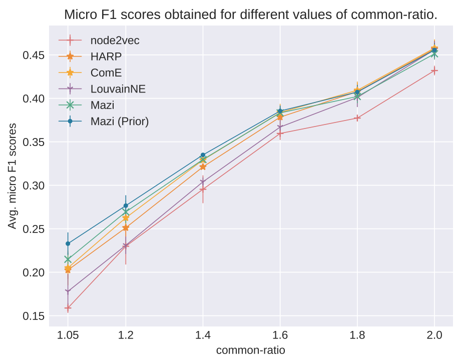


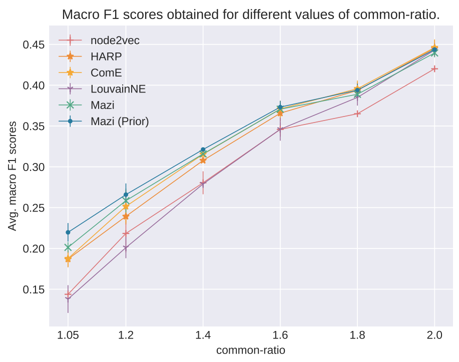
Fig. 2: The hierarchical tree structure used to generate our synthetic datasets. It has 4 levels. While the finest level uses a branching factor of 75, all other levels use 5.

In our experiments, we create a 5-level hierarchical tree with a branching factor of 5 in most levels. The branching factor in the level before the leaves is 75, thus, resulting in a total of 9375 nodes. We range `common-ratio` between $\{1.05, 1.2, 1.4, 1.6, 1.8, 2.0\}$. On average, the *modularity* of the graph for the corresponding `common-ratio` is $0.23, 0.28, 0.33, 0.37, 0.41, 0.44$. The power-law distribution parameter is 4.5 for the node degree. The maximum and the average degree of a node in the (directed) graphs we study are 187 and 33, respectively.

Evaluation on the multi-label node classification task. Figure 3 plots the micro and macro F1 scores obtained by the methods on the synthetic datasets on node classification. The average gains observed in the macro F1 scores by *Mazi (Prior)* against *node2vec* range from over 50% to 5% for the `common-ratio` value of 1.05 to 2.0. As the *modularity* of the graph, as defined by the finest level community structure, decreases, the random-walks in *node2vec* will tend to stray outside the community. The labels are, however, distributed in accordance with the community structure, and thus, could indicate its lowered performance. *Mazi (Metis)* achieves similar performance as *Mazi (Prior)* against *node2vec*, ranging from 42% to 5% for `common-ratio` 1.05 to 2.0. Further, *Mazi (Prior)* and *Mazi (Metis)* both are able to demonstrate significant benefits in comparison to *HARP* for graphs with `common-ratio` ranging from 1.05 to 1.6. The average gain obtained by *Mazi (Prior)* and *Mazi (Metis)* are as high as 19% and 9.5%, respectively, for `common-ratio` 1.05. We reason that for the graphs whose *modularity*, as defined by the prior hierarchical community structure is low, the coarsening scheme of *HARP* may not be able to capture a fitting hierarchical community structure. Thus, the representations learnt on the coarsest level may not serve as good initializations for finer levels. Against *ComE*, we observe similar gains at about 20% with `commmon-ratio` 1.05 in the F1 scores. We also observe similar trends



(a) Micro F1 Score



(b) Macro F1 Score

Fig. 3: Average micro and macro F1 scores on the synthetic graphs. Results are obtained over 3 runs with standard deviation. *HARP* method is built on the *node2vec* model, *Mazi (Prior)* uses the community structure generated by the hierarchical clustering tree, and *Mazi* uses the community structure generated by Metis. *Mazi (Metis)*, uses 4 levels in the hierarchy. The #communities in next coarser level is generated using $\sqrt{n}$, where, n is the number of nodes in the graph in the current level.

with *Mazi* over *LouvainNE*. For the lower values of common ratio, and thus, the *modularity*, we believe that *LouvainNE* may not be able to capture the local neighborhood of a node well.

4.3 Ablation Study

We study the effect of two important parameters, β and γ, in the performance of *Mazi*. We set $\beta = 0.0$ to fully ignore the contribution of the proximity between the node and its community representations while optimizing the objective. We set $\gamma = 0.0$ to fix the hierarchical community structure to its initial value and optimize only the node representations. In node classification, a non-zero value of β plays a crucial role in ensuring *Mazi*'s good performance (see Table 6). Since the representations learned are benefited by the knowledge of a hierarchical community structure, performance achieved by $\beta = 0.0$ is consistently lower than when $\beta \neq 0.0$. The effect of γ is more apparent in *Mazi* using the *Metis* community structure. Since the hierarchical community structure generated using *Metis* does not fully conform to the prior community structure and the label distribution on the synthetic graphs correlate with the finest level community structure, we note that refining the hierarchical community structure and thereby, using it to improve the representations lead to better performance of the model.

We also report the effectiveness of β and γ in link prediction in Table 7. All the datasets achieve better performance when accounting for non-zero values

Table 6: Ablation study on node classification. Macro F1 scores and %gain achieved by *Mazi (Prior)* and *Mazi* against respective versions without Q^l (Eq. 1) and without L^l_{comm} (Eq. 4) are reported for the synthetic graphs over 3 runs. The standard deviation up to 2 decimal points is reported in the parentheses.

	Mazi (Prior)					*Mazi*				
CR	$\gamma =$ 0.0	%gain w/o Q^l	$\beta =$ 0.0	%gain w/o L^l_{comm}	$\beta, \gamma \neq 0.0$	$\gamma =$ 0.0	%gain w/o Q^l	$\beta =$ 0.0	%gain w/o L^l_{comm}	$\beta, \gamma \neq 0.0$
1.2	**0.267**	-0.42 (0.61)	0.256	4.26 (0.56)	0.266	0.258	0.28 (0.89)	0.256	0.97 (0.91)	**0.259**
1.4	0.321	0.07 (0.10)	0.316	1.83 (1.12)	**0.321**	0.314	0.63 (0.04)	0.314	0.69 (0.08)	**0.316**
1.6	**0.374**	-0.10 (0.13)	0.369	1.14 (0.10)	0.373	0.371	0.03 (0.55)	0.369	0.49 (1.01)	**0.371**
1.8	0.394	0.09 (0.12)	0.387	1.77 (0.98)	**0.394**	0.388	0.27 (0.41)	0.387	0.39 (0.58)	**0.389**
2.0	0.444	0.00 (0.00)	0.437	1.48 (0.63)	**0.444**	0.439	0.20 (0.33)	0.438	0.35 (0.02)	**0.440**

Table 7: Ablation study on link prediction. Mean average precision and %gain, averaged over 3 runs, achieved by *Mazi* on link prediction over *Mazi* without Q^l (Eq. 1) and *Mazi* without L^l_{comm} (Eq. 4) is reported for the real-world graphs. The standard deviation up to 2 decimal points is reported in the parentheses.

Graph	$\gamma =$ 0.0	%gain w/o Q^l	$\beta =$ 0.0	%gain w/o L^l_{comm}	$\beta, \gamma \neq 0.0$
BlogCatalog	0.586	0.15 (0.03)	0.564	4.09 (0.09)	**0.587**
CS_CoAuthor	0.823	0.03 (0.12)	0.821	0.29 (0.09)	**0.824**
DBLP	**0.930**	-0.02 (0.08)	0.929	0.08 (0.17)	0.930

of the β. This is especially evident in the *BlogCatalog* dataset, wherein *Mazi* shows a gain as high as 4.09%. Further, the community structure refinement in *BlogCatalog* and *CS_ CoAuthor* leads to better performance when $\gamma \neq 0.0$, whereas in *DBLP*, the results obtained are comparable in both cases.

5 Related Work

Several methods model node representations using deep learning losses in supervised, semi-supervised and unsupervised settings. Amongst the unsupervised methods, the Skip-gram model is a popular approach used in the literature [14, 6] to model the local neighborhood of a node using random walks while learning its representation. However, unlike our method, these representations are inherently flat and do not account for the hierarchical community structure that is present in the network.

Existing methods have also explored jointly learning communities at a single level and the representations of the nodes in the graph [2, 15]. *ComE* [2] models the community and the node representations using a gaussian mixture formulation. *vGraph* [15] assumes each node to belong to multiple communities and a community to contain multiple nodes, and parametrizes the node-community distributions using the representations of the nodes and communities. Unlike these approaches, our approach utilizes the inductive bias introduced by the hierarchical community structure in the representations.

Recently, many unsupervised hierarchical representation learning methods, such as *HARP* [3] and *LouvainNE* [1], have been explored that leverage the multiple levels formed by hierarchical community structure in the graph. *HARP* uses an existing methods, such as *node2vec*, to generate node representations for graphs at coarser levels and initializes node representations at finer levels using these. *LouvainNE* recursively partitions each community in a graph to form sub-communities. The representations for a node in all the different sub-communities are generated and subsequently aggregated in a weighted fashion to form the final node representation. *SpaceNE* [11] represents the hierarchical community structure using sub-spaces in the feature space and learns node representations that preserves proximity between nodes as well as similarities within communities and across communities. However, all these approaches consider a static hierarchical community structure to influence the representations. In comparison, we jointly learn the node representations and the hierarchical community structure that is influenced by the node representations. In a parallel line, some GNN-based methods have been suggested to model the hierarchical structure present in the graphs while learning network representations [17, 8, 18, 10]. While *DiffPool* [17] and *AttPool* [8] learn graph representations, *HC-GNN* [18] and *GXN* [10] target node representation learning. However, these are supervised methods and use task specific losses while considering static hierarchical community structures.

6 Conclusion

This paper develops a novel algorithm, *Mazi*, for joint unsupervised learning of a given graph's node representations and hierarchical community structure. At each level in the hierarchy, *Mazi* coarsens the graph and learns its node representations and leverages them to discover communities in the hierarchy. In turn, *Mazi* uses the hierarchy to learn the representations. Experiments conducted on synthetic and real-world graph datasets in the node classification and link prediction demonstrate the competitive performance of *Mazi* compared to competing approaches.

Acknowledgements This work was supported in part by NSF (1447788, 1704074, 1757916, 1834251), Army Research Office (W911NF1810344), Intel Corp, and the Digital Technology Center (DTC) at the University of Minnesota. Many thanks to the reviewers for their helpful inputs. Also, many thanks to Saurav Manchanda for his helpful discussions. Access to research facilities was provided by the DTC & the Minnesota Supercomputing Institute.

References

1. Bhowmick, A.K., Meneni, K., Danisch, M., Guillaume, J.L., Mitra, B.: Louvainne:Hierarchical louvain method for high quality and scalable network embedding. In:Proceedings of 13th Intl. Conf. on Web Search and Data Mining (2020)
2. Cavallari, S., Zheng, V.W., Cai, H., Chang, K.C.C., Cambria, E.: Learning community embedding with community detection & node embedding on graphs. In:Proc. of 2017 ACM on CIKM (2017)
3. Chen, H., Perozzi, B., Hu, Y., Skiena, S.: Harp: Hierarchical representation learning for networks. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
4. Clauset, A., Moore, C., Newman, M.E.: Hierarchical structure and the prediction of missing links in networks. Nature 453(7191), 98–101 (2008)
5. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: A library for large linear classification. Journal of machine learning research 9(Aug), 1871–1874
6. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD conf. on KDD. pp. 855–864 (2016)
7. Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: Methods and applications. arXiv:1709.05584 (2017)
8. Huang, J., Li, Z., Li, N., Liu, S., Li, G.: Attpool: Towards hierarchical feature representation in graph convolutional networks via attention mechanism. In: Proceedings of the IEEE ICCV. pp. 6480–6489 (2019)
9. Karypis, G., Kumar, V.: Multilevel graph partitioning schemes. In: ICPP (3) ('95)
10. Li, M., Chen, S., Zhang, Y., Tsang, I.W.: Graph cross networks with vertex infomax pooling. arXiv preprint arXiv:2010.01804 (2020)
11. Long, Q., Wang, Y., Du, Lun & Song, G., Jin, Y., Lin, W.: Hierarchical community structure preserving network embedding: Subspace approach. In: 28th ACM CIKM (2019)
12. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv:1301.3781 (2013)
13. Newman, M.E.: Modularity and community structure in networks. Proceedings of the national academy of sciences 103(23), 8577–8582 (2006)
14. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD intl. conf. on Knowledge discovery and data mining (2014)
15. Sun, F.Y., Qu, M., Hoffmann, J., Huang, C.W., Tang, J.: vgraph: A generative model for joint community detection and node representation learning. arXiv preprint arXiv:1906.07159 (2019)
16. Yang, B., Yih, W.t., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575 (2014)
17. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. In: Advances in Neural Information Processing Systems. pp. 4800–4810 (2018)
18. Zhong, Z., Li, C.T., Pang, J.: Hierarchical message-passing graph neural networks. arXiv preprint arXiv:2009.03717 (2020)