Node Representation Learning Assisted By Maximal Cliques

Navjot Singh*
University of California, Los Angeles
Los Angeles, California, USA
navjotsingh@ucla.edu

Edward W Huang
Amazon
Palo Alto, California, USA
ewhuang@amazon.com

Nikhil Rao
Amazon
Palo Alto, California, USA
nikhilsr@amazon.com

Karthik Subbian
Amazon
Palo Alto, California, USA
ksubbian@amazon.com

ABSTRACT

Graph Neural Networks (GNNs) have recently emerged as powerful tools for learning latent node representations from large-scale graphs. However, most GNN models only consider pairwise or local patch connectivity of nodes when constructing representations, ignoring higher-order structural information in the graph. Higher-order structures such as cliques are at the scale of small subgraphs and encode group interactions between sets of nodes, which can provide richer substructure information on the graph.

In this paper, we propose a new objective for learning node representations. This objective accounts for local neighborhood information of nodes as well as their higher-order connectivity information, captured via maximal cliques. Our framework is agnostic to the choice of representation learning model and can be used with any representation learning GNN model in literature.

We provide experimental results for evaluating the representations learned by our clique-guided approach on multiple real-world datasets and popular GNN models. Our results demonstrate that utilizing clique structures in the training process improves the performance of the GNN models. We also compare our representations to baselines that incorporate other notions of higher-order information.

KEYWORDS

graph neural networks, maximal cliques, self-supervised training

1 INTRODUCTION

Graphs describe complex relationship structures between data in many fields, such as social network analysis [5], genomics [4], and communication networks [20]. Many important problems in the machine learning community involve graph-structured datasets. For e.g., in social networks, we may be interested in predicting the existence of a link between users [27]. In biology, we may want to identify proteins in a disease pathway given their interaction graph [1, 24]. In recommender systems, we may want to predict if a user will like a particular product based on their past purchases [11, 14]. A fundamental aspect of these settings involves learning a *representation* for elements in the graph, such as nodes, edges, or the graph itself. These representations encode the structural connectivity of the underlying graph and any feature inputs associated with the graph elements. Learning high-quality representations can improve downstream tasks, such as predicting characteristics of graph elements (e.g., node/edge/graph classification).

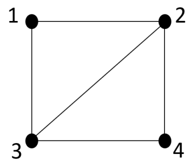

Figure 1: Example undirected graph with cliques

In this work, we focus on the paradigm of self-supervised representation learning for nodes, where labels are absent and the representations are learned solely with the graph connectivity information. Specifically, we study this problem in the context of Graph Neural Networks (GNNs). GNNs have emerged as a powerful tool to learn rich representations from graphs in a data-driven manner [9, 30]. There has been a significant amount of interest in this area, with many self-supervised GNN models having been proposed in literature [8, 9, 18, 25, 26].

Typically, GNNs learn node-level representations by aggregating pairwise or local neighborhood information in the graph. Most of these methods do not use higher-order structural structures, which encode interactions between groups rather than pairs of nodes. These structures are crucial to the organization of many complex networks [21, 28, 29]. An important and well-studied higher-order structure in graph theory is a *clique*, which is a complete subgraph in which each node can reach every other node in exactly one hop. A *maximal clique* is a clique that cannot be extended any further by including an additional node. These structures encode dense connectivity properties of the underlying graph and define the boundary of highly related sets of nodes in the networks. Nodes in the same cliques may share common properties or have similar functions within the network [21].

Maximal cliques have several interesting properties that are relevant to the representation learning problem. They are isomorphic structures where all nodes in a clique carry equal importance to the clique, relaxing the need to distinguish these nodes from each other. In addition, they are better at extracting overlapping structures in natural small world graphs when compared to other higher-order structures like clusters. As an example, consider the graph presented in Figure 1. The Fiedler vector[1] of this graph used in

*Work done during internship at Amazon LLC.

[1]The Fiedler vector of a connected, undirected graph $\mathcal{G}$ is defined as the eigenvector corresponding to the second smallest eigenvalue of the Laplacian matrix of $\mathcal{G}$.

spectral clustering partitions the nodes into the sets $\{1, 2, 3\}$ and $\{4\}$. Meanwhile, it is obvious that there is an overlapping structure in this example, $\{1, 2, 3\}$ and $\{2, 3, 4\}$, which forms the two maximal cliques. While maximal cliques provide useful connectivity information about the node interactions, they come with computational costs. It is well known that the problem of extracting all maximal cliques in a graph is NP-complete [15].

In this paper, we utilize the structural information of maximal cliques in a graph to improve the quality of node representations learned by self-supervised GNN models. To this end, we propose a novel method to extract a set of maximal cliques by greedily probing nodes of interest, defined by a centrality measure. This provides an efficient heuristic to yield a candidate set of maximal cliques covering all nodes in the graph. Using the obtained clique assignments, we formulate a novel objective for the self-supervised training of GNN models, which takes into account both the traditional graph reconstruction objective as well as an objective based on clique information. In this clique-based objective, representations of nodes belonging to the same maximal clique are brought closer to each other, while representations of nodes that have very little overlap across different maximal cliques are pulled farther apart. Such representations provide richer information about the spatial context of nodes compared to traditional self-supervised GNNs. We show that our objective leads to significant performance gains in downstream tasks. Our contributions are as follows:

- We introduce the concept of probing a graph to discover maximal cliques. Specifically, we formulate the problem of finding a maximal clique given a probe as a relaxed linear program, which can be solved efficiently via popular off-the-shelf solvers. Using this idea, we design a greedy probing strategy based on node centrality measures to yield a set of maximal cliques covering all the nodes in the graph.
- We develop a novel objective for self-supervised GNN training to learn representations accounting for both local and higher order connections of a node. Our proposed approach is agnostic to the choice of the base GNN model used in training and can be used on top on any popular representation learning model which learns latent node representations.
- We provide experimental results on real-world datasets and demonstrate the effectiveness of learning representations using our novel objective for multiple popular GNN models in the literature. We further provide comparison of our clique-guided representation learning approach to learning with other notions of higher order structures like clusters.

2 PROPOSED FRAMEWORK

In this section, we describe our end-to-end framework for the self-supervised learning and evaluation of clique-guided node representations. Before delving into the technical aspects of our proposed framework, we first establish the notations we will use.

2.1 Notation and Approach

We denote with $\mathbb{R}^d$ the set of real valued $d-$dimensional vectors. We work with the family of undirected, connected graphs. We consider two settings of interest in the paper: (i) General graphs with number of nodes n, with adjacency matrix denoted by $A = [a_{ij}] \in \mathbb{R}^{n\times n}$. (ii) Bipartite graphs with m, n nodes in respective partitions with the bi-adjacency matrix denoted by $B = [b_{ij}] \in R^{m\times n}$. We denote by $\mathbb{1}_n$ the vector of all ones in $\mathbb{R}^n$.

At a high level, the goal of this work is to learn node representations by taking into account maximal clique structures in the underlying graph. Representations learned in such a manner contain rich information about the graph connectivity structure at the local node level and the higher-order organization of the nodes. Thus, they are useful for downstream applications. To this end, the technical aspects of our approach to learn and evaluate clique-guided node representations comprise three stages:

- Stage I: Efficient maximal clique extraction from the underlying graph via a novel probing technique.
- Stage II: Perform self-supervised node representation learning incorporating the learned maximal clique information.
- Stage III: Utilize the learned node representations on downstream applications.

The technical details of our approach for Stage I and Stage II are discussed in the subsections below. Stage III forms the bulk of our experimental results presented in Section 3, where we demonstrate the effectiveness of learning clique-guided representations for various graph datasets and compare them with related schemes.

2.2 Clique Extraction (Stage I)

It is known that the problem of finding all maximal cliques in a graph is NP-complete [15]. Hence, we resort to a heuristic greedy approach as an alternative. We consider instead the problem of finding a maximal clique containing a node of interest, which we call a *probe.* We consider a relaxed linear program formulation for this problem, the solution of which can be rounded to integer values to yield the maximal cliques containing the probe.

2.2.1 Cliques in general undirected graphs. The optimization program for the clique containing the probed node p in a general undirected graph is given as follows:

$$\max_{\mathbf{x}\in[0,1]^n} \langle \mathbf{x}, \mathbf{1}_n \rangle \quad (1)$$
$$\texttt{subject to: } x_i + x_j \leq 1 + a_{ij}, \ \forall\, i, j \in [n]$$
$$x_p = 1$$

The solution is given by vector $\mathbf{x}$, which is constrained to have value 1 at the probed node p. The solution can be rounded to the nearest integers in $\{0, 1\}$, denoting whether the corresponding node is included in the clique.

The constraint $x_i + x_j \leq 1$ when $a_{ij} = 0$ ensures that when the nodes i and j are not connected, they cannot have a value of 1 simultaneously and be included in the clique. Thus, any node not connected to the probed node p cannot be included in the solution for the probe, implying that we must only search in a 1-hop neighborhood around the probed node p to find its corresponding maximal clique. The resulting optimization procedure is efficient, as the search space depends on the degree of the probed node rather than the total number of nodes in the graph.

Algorithm 1 presents a greedy scheme for finding a set of maximal cliques covering all nodes in the graph. We take as input the adjacency matrix A and an ordered list of nodes $\mathcal{R}$ ranked by some importance measure (e.g., a centrality score of the nodes in the

Algorithm 1 Greedy scheme for finding maximal cliques

```
Input: Symmetric adjacency matrix $A = [a_{ij}]$ of size $n \times n$ of the underlying undirected graph, list of node centrality rankings in descending order (denoted by $\mathcal{R}$).
Initialize: Empty set $\mathcal{N}_c$. Empty list $\mathfrak{L}_c$.
1: while $|\mathcal{N}_c| \neq n$ do
2:     Consider the highest ranking node in $\mathcal{R}$ not present in $\mathcal{N}_c$. Let the index of this node in the graph be $p$.
3:     Solve the Linear Program in (1) for probe at node $p$.
4:     Round the obtained solution $\mathbf{x}$ to integer values in $\{0, 1\}$.
5:     Define $\mathcal{N} = \{i : x_i = 1, 1 \leq i \leq n\}$, denoting the set of nodes which are included in the clique.
6:     Append $\mathcal{N}$ to the list of cliques $\mathfrak{L}_c$.
7:     Update the set of discovered nodes as:
           $\mathcal{N}_c \leftarrow \mathcal{N}_c \cup \mathcal{N}$
8: end while
Output: List of cliques $\mathfrak{L}_c$
```

graph). In this paper, we use the PageRank algorithm [22] to generate these scores. We initialize an empty set $\mathcal{N}_c$, which stores nodes that already belong to an extracted clique.

The algorithm starts by taking the highest-ranking node in $\mathcal{R}$ and not yet in the $\mathcal{N}_c$. This node is the probed node, p. The algorithm then proceeds to solve the linear program in Equation (1) with p and the adjacency matrix. The linear program can be solved with any open-source solver (e.g., ECOS [6], OSQP [23], CVXOPT [2]). The solution, $\mathbf{x}$, is rounded to the nearest integer in $\{0, 1\}$. The nodes in the clique have a value of 1 in the solution $\mathbf{x}$, which we store in $\mathcal{N}$. We append the found clique to the clique list $\mathfrak{L}_c$. Finally, the nodes in the clique $\mathcal{N}$ are included in the set $\mathcal{N}_c$.

In the following iterations, the algorithm again picks the highest-ranking node not in $\mathcal{N}_c$, terminating when all nodes in the graph are included in some discovered clique ($|\mathcal{N}_c| = n$). At each iteration, the probed node p always ends up in the set $\mathcal{N}_c$. Thus, the algorithm is guaranteed to terminate.

The idea of using a ranked list is that the nodes with the highest centrality score are generally a part of a dense subgraph and have large degrees. Thus, probing nodes in order of their centrality scores ensures that larger cliques are found within the first few iterations and the resulting procedure terminates quickly.

2.2.2 *Bicliques in biparite graphs.*

The optimization problem in (1) cannot directly solve for bicliques, since bipartite graphs can only have connectivity between its two disjoint partitions. We propose the following relaxed linear program for a pair of probes:

$$\begin{aligned} \max_{\mathbf{u}\in[0,1]^m,\, \mathbf{v}\in[0,1]^n} \quad & \langle \mathbf{u}, \mathbf{1}_m \rangle + \langle \mathbf{v}, \mathbf{1}_n \rangle \\ \texttt{subject to:} \quad & u_i + v_j \leq 1 + b_{ij}, \ \forall\, i \in [m], j \in [n] \\ & u_p = 1,\ v_q = 1 \end{aligned} \tag{2}$$

This program can be seen as a relaxation of the program considered in [12, Section 3.1], suitably modified to yield a maximal biclique containing a pair of probed nodes, each lying in a separate part of the graph. The desired solution is given by the pair $\mathbf{u}, \mathbf{v}$, denoting the nodes belonging to the obtained biclique, with $\mathbf{u}$ from

Algorithm 2 Greedy scheme for finding maximal bicliques

```
Input: Bipartite adjacency matrix $B = [b_{ij}]$ of size $m \times n$ of the underlying undirected bipartite graph, list of node centrality rankings in descending order (denoted by $\mathcal{R}$).
Initialize: Empty set $\mathcal{N}_c$. Empty list $\mathfrak{L}_c$.
1: while $|\mathcal{N}_c| \neq m + n$ do
2:     Consider the highest ranking node in $\mathcal{R}$ not present in $\mathcal{N}_c$. Let the index of this node in the graph be $p$.
3:     Let $q$ be the highest degree neighbor of the node $p$ which is not in $\mathcal{N}_c$.
4:     Solve the Linear Program in (2) for probes at node $p$ and $q$.
5:     Round the obtained solution $\mathbf{u}, \mathbf{v}$ to integer values in $\{0, 1\}$.
6:     Define $\mathcal{N}_1 = \{i : u_i = 1, 1 \leq i \leq m\}$ and $\mathcal{N}_2 = \{j + m : v_j = 1, 1 \leq j \leq n\}$, denoting the set of nodes which are included in the biclique.
7:     Append the obtained biclique $\mathcal{N}_1 \cup \mathcal{N}_2$ to the list of bicliques $\mathfrak{L}_c$.
8:     Update the set of discovered nodes as:
           $\mathcal{N}_c \leftarrow \mathcal{N}_c \cup (\mathcal{N}_1 \cup \mathcal{N}_2)$
9: end while
Output: List of cliques $\mathfrak{L}_c$
```

the part of size m and $\mathbf{v}$ from the other part of the graph. The two probed nodes p, q are included in the biclique, which is captured by the constraint $u_p = 1$ and $v_q = 1$. The constraint $u_i + u_j \leq 1$ when $b_{ij} = 0$ ensures that nodes not connected across the partitions are not included in the biclique. As before, this implies that only the 1-hop neighborhood around the probed nodes are candidates for the biclique, narrowing the search space.

We present our proposed method of probing a bipartite graph and finding a set of maximal bicliques covering all nodes in the graph in Algorithm 2. The development is similar to that of Algorithm 1 but with pairwise probes on the graph partitions.

In the list of cliques $\mathfrak{L}_c$ generated by Algorithms 1 and 2, a node may belong to multiple maximal cliques or bicliques. To utilize the discovered cliques in node representation learning, we encode the clique assignments of each node as a binary vector. Specifically, for the case of an undirected, connected graph with n nodes, consider that Algorithm 1 produces M cliques. For a node $i \in [n]$, we define the M-length binary vector $\mathbf{c}^{(i)}$ with the j^{th} entry for $j \in [M]$ as:

$$\mathbf{c}_j^{(i)} = \begin{cases} 1, & \text{if } i \text{ belongs to clique } j \\ 0, & \text{otherwise} \end{cases}$$

A similar binary vector can also be constructed for the case of bicliques. The clique assignment vectors for the the nodes $\{\mathbf{c}^{(i)}\}_{i=1}^{n}$ are passed to Stage II to learn node representations.

2.3 Representation Learning (Stage II)

We propose a joint training loss for learning node representations, which simultaneously accounts for the graph reconstruction objective while integrating the clique assignment information of the

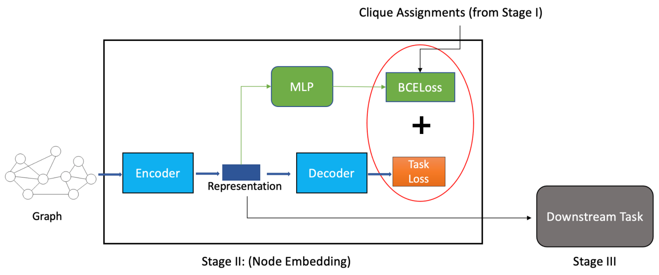

Figure 2: The training framework for the self-supervised learning of node representations (Stage II). The inputs are the graph structure and the clique information from Stage I. The learned node representations can be used in a downstream task of interest (Stage III).

nodes. Formally, for the graph $\mathcal{G}$ with n nodes and the clique assignments $\{\mathbf{c}^{(i)}\}_{i=1}^{n}$ from Stage I, this loss is given as:

$$F = f_{rec}(\mathcal{G}) + f_{clique}(\{\mathbf{c}^{(i)}\}_{i=1}^{n}) \tag{3}$$

In (3), $f_{rec}(\mathcal{G})$ is the reconstruction loss for learning node representations. This loss comes from the base representation-learning model, which typically relies on predicting the connectivity structure of $\mathcal{G}$. This is often achieved through an encoder-decoder framework where the latent node representations (output of encoder) minimize some reconstruction criteria. For example, the *Graph Auto-Encoder* [18] reconstruction loss comprises the cross-entropy loss associated with predicting the entries of the graph's adjacency matrix via the node embeddings, while the *Deep Graph Infomax* [26] reconstruction loss maximizes the mutual information between the node representations and a global graph summary vector.

The second loss $f_{clique}(\{\mathbf{c}^{(i)}\}_{i=1}^{n})$ in Equation (3), which we call the *clique loss*, captures a clique assignment prediction loss for the nodes. Specifically, using the intermediate node representations obtained from the underlying base representation learning model, we form a supervised learning task where the inputs are the node representations and we seek to predict the clique assignment vectors $\{\mathbf{c}^{(i)}\}_{i=1}^{n}$. In this work, we achieve this by training a simple multilayer perceptron (MLP) model on the input representations with a binary cross-entropy loss (as a node may possibly belong to more than one clique). Thus, the latent node representations that are learned to minimize $f_{rec}(\mathcal{G})$ simultaneously predict the clique assignment vector associated with each node.

A diagrammatic description of the training process is given in Figure 2. At the end of the self-supervised training process, we obtain representations for all nodes in the graph. These representations account for both the local neighborhood information (by minimizing the reconstruction loss) and the higher-order connectivity information (by minimizing the clique loss). These representations can then be passed to downstream tasks (Stage III).

Note that the clique loss $f_{clique}(\{\mathbf{c}^{(i)}\}_{i=1}^{n})$ in (3) can be replaced with the prediction of any other higher-order structure (e.g., cluster label assignment). We include these alternatives as baselines (Section 3) and demonstrate that integrating clique assignments provides better performance than other higher-order structures.

3 EXPERIMENTAL RESULTS

We now provide quantitative results to demonstrate the effectiveness of our clique-guided representation learning framework. We compare our method to several baseline GNN architectures, on multiples datasets, and also to other methods that use clustering information to learn node representations.

3.1 Datasets and Downstream Tasks

We first describe the graph datasets and the associated downstream tasks we used to evaluate learned node representations. We also briefly discuss the clique (or biclique) structures obtained and comment on their utility for learning better representations. Table 1 contains dataset statistics. Appendix **??** contains additional details and network architectures for the downstream tasks.

3.1.1 MovieLens 100K [10]. This is a bipartite graph of user-movie entities with a rating associated with each edge. A set of bicliques in this dataset naturally groups together users who have similar movie preferences. Using Algorithm 2 on this dataset yields 97 bicliques.

For the downstream task, we consider a like/dislike prediction task to evaluate the representations. During evaluation, edge representations of user-movie pairs are constructed by concatenating the node representations.

3.1.2 Movie-Movie (100K). Derived from the MovieLens 100K dataset, this a non-bipartite graph of movie connections in the underlying graph. For a given movie-movie pair, we evaluate the Jaccard index [13] between their user sets (say, given by U_1 and U_2) as:

$$J(U_1, U2) = \frac{|U_1 \cap U_2|}{|U_1 \cup U_2|}$$

For movie-movie pairs with non-intersecting user sets, $J(U_1, U_2) = 0$. A movie-movie pair is connected if its Jaccard index is greater than the mean of the Jaccard index over all connected pairs.

A clique in this graph groups movies which are frequently watched together by the users, and thus might share similar themes or belong to a common genre. Algorithm 1 extracts 121 cliques from the constructed movie-movie graph. We only keep cliques with size greater than 70. This results in 51 cliques that cover about 90% of all nodes in the movie-movie graph. Nodes that do not belong to any of the 51 resulting cliques do not contribute to the clique assignment loss in Stage II.

For downstream evaluation, we consider a multi-label prediction task where we predict the genres a given movie belongs to.

3.1.3 MOOC dataset [19]. This is a bipartite graph comprising actions participating students take throughout a course. A biclique in this graph groups together students who take similar action throughout the course and have potentially similar learning styles. Algorithm 2 yields 94 cliques.

We consider a link prediction task for evaluating the representations based on predicting a set of edges/non-edges of the original graph. The set of positive edges are held out when learning self-supervised representations in Stage II.

3.1.4 Movie-Movie (1M). Similar to the Movie-Movie (100K) graph, this graph is derived from the larger MovieLens (1m) dataset [10]. We only keep cliques with size greater than 90, which yields 160 cliques that collectively cover about 88% of all nodes in the graph.

Table 1: Dataset statistics

	MovieLens100K	Movie-Movie (100K)	ACT-MOOC	Movie-Movie (1m)
# Nodes	2,625	1,682	7,114	3,706
# Edges	100,000	360,493	411,749	2,117,749

Downstream evaluation is performed via a multi-label genre prediction task for the movies.

To assess the quality of the learned representations on the downstream task, we consider test accuracy and F1-score as the performance metrics.

3.2 Representation Learning Models

The representation learning model in Stage II takes as input the underlying graph structure with any associated features and the clique information from Stage I, then constructs node representations in a self-supervised manner. We consider the following models:

(i) Graph Auto-Encoder (GAE) [18]: Node representations are learned in an encoder-decoder framework by reconstructing the adjacency matrix of the underlying graph.
(ii) GraphSAGE [8]: This model learns a set of aggregator functions that are used to construct node embeddings by utilizing feature information from a node's local neighborhood.
(iii) Graph Convolutional Networks (GCN) [17]: To learn the representations in a self-supervised manner, we formulate a link-prediction task utilizing edge representations formed by concatenating node representations obtained by GCN layers.
(iv) Deep Graph Infomax (DGI) [26]: This model relies on learning node representations by sampling local patch information for each node and maximizing the mutual information in the patch and a global graph readout function. The optimization problem for learning the representations via mutual information relies on a contrastive learning approach.

For the downstream evaluation task for all datasets, we train an MLP model on the learned representations to predict the associated dataset labels. For edge prediction tasks (MovieLens100K and MOOC datasets), the edge representations are constructed by concatenating the node representations. Appendix **??** contains implementation details for the models.

3.3 Baselines

Our proposed scheme to learn self-supervised node representations relies on the optimizing the learning objective in (3), which utilizes the local dense partitioning of the nodes via the clique loss. We include comparisons with other partitioning methods that capture higher-order connectivity and be used in place of the clique loss.

(i) METIS: The METIS software package [16] provides tools for producing high-quality graph partitions for large-scale undirected graphs in an efficient manner. The number of partitions k to be produced is provided as input to the algorithm. In our experimental results given below, **METIS**-k denotes the METIS algorithm with k number of partitions.
(ii) Louvain: Louvain partitioning [3] is a popular method for fast community extraction in large scale graphs. The algorithm does not take any inputs apart from the graph structure and is based on modularity optimization.
(iii) Node2vec + k-means: Node2vec [7] is an algorithm for learning low-dimensional node representations for a graph by optimizing a neighborhood-preserving objective. Using the traditional k-means clustering method on the resulting node representations provides an effective way to construct graph partitions. In our results provided below, **node2vec + k-means (k)** denotes the node2vec algorithm follows by k-means clustering with k input clusters.
(iv) No label: This baseline corresponds to providing no higher-order connectivity information of the graph when learning the representations in Stage II. This is the default representation learning method of the base GNN models.

3.4 Results

We summarize the performance metrics for the different datasets in Table 2 to Table 5. For each dataset, we provide the comparison between our clique-based representation learning framework and the different baselines described in Section 3.3. For each of these higher-order structures (as well as the no-label baseline), we provide the evaluation metrics for the associated downstream task averaged over 5 runs, for each GNN model listed in Section 3.2.

We observe that across the board, utilizing higher-order information structures to learn node representations leads to better downstream performance than using only vanilla GNNs. Moreover, among the higher-order structures considered, our proposed clique-guided node representations yield the best downstream performance across almost all underlying GNN models and datasets. We conclude that leveraging maximal cliques in addition to local connectivity can lead to more informative representations for downstream applications.

4 CONCLUSIONS AND FUTURE WORK

We proposed a framework for learning node representations guided by maximal clique structures in the graph to capture higher-order connectivity information of the nodes. Our heuristic method for finding maximal cliques is based on optimizing a linear program (LP), which can be done efficiently using any off-the-shelf LP solver. Moreover, our training framework can be applied on top of any popular base GNN self-supervised learning model in literature to learn clique-guided representations. We demonstrate the effectiveness of our method by evaluating the learned representations on downstream tasks for various real-world datasets with different base GNN models.

While cliques are effective in capturing higher-order connections, one might also be interested in considering quasi-cliques, where we can relax the requirement for complete connectivity, and instead only ask for a fraction of edges to be present in the subgraph. These

Table 2: Performance comparison for different techniques on the MovieLens 100K dataset.

Technique	GAE		GCN		DGI		GraphSAGE	
	Test acc.	F1 score	Test acc.	F1 score	Test acc.	F1 score	Test acc.	F1 score
No label	69.5	0.742	67.7	0.727	70.5	0.748	69.3	0.738
Louvain	69.2	0.739	65.6	0.719	68.5	0.733	68.4	0.734
METIS-10	68.8	0.735	68.1	0.729	68.4	0.734	69.4	0.738
METIS-20	69.7	0.743	68.2	0.728	68.8	0.737	69.3	0.739
Node2vec + k-means (10)	69.1	0.737	67.7	0.727	68.9	0.739	69	0.738
Node2vec + k-means (20)	69.6	0.742	67.2	0.727	69.4	0.741	69.4	0.74
Node2vec + k-means (50)	69.5	0.743	67.8	0.729	69.7	0.746	69.4	0.742
Clique (ours)	**70.5**	**0.749**	**69.1**	**0.739**	**70.5**	**0.752**	**70.1**	**0.744**

Table 3: Performance comparison for different techniques on the Movie-Movie (100K) dataset.

Technique	GAE		GCN		DGI		GraphSAGE	
	Test acc.	F1 score	Test acc.	F1 score	Test acc.	F1 score	Test acc.	F1 score
No label	91.4	0.336	91.2	0.338	91.3	0.356	90.8	0.332
Louvain	91.6	0.335	**91.5**	0.356	90.8	0.345	**91.7**	0.334
METIS-10	91.4	0.331	91.2	0.348	91.7	0.359	91.2	0.344
METIS-20	91.6	0.344	91.2	0.358	91.7	0.384	91.4	0.355
Node2vec + k-means (10)	91.4	0.328	91.3	0.359	91.3	0.349	91.2	0.346
Node2vec + k-means (20)	91.5	0.341	91.3	0.365	91.6	0.386	91.6	0.361
Node2vec + k-means (50)	91.5	0.337	91.2	0.354	91.6	0.366	91	0.342
Clique (ours)	**91.7**	**0.351**	**91.5**	**0.383**	**91.9**	**0.409**	91.1	**0.369**

Table 4: Performance comparison for different techniques on the ACT-MOOC dataset.

Technique	GAE		DGI		GraphSAGE	
	Test acc.	F1 score	Test acc.	F1 score	Test acc.	F1 score
No label	88.5	0.891	89	0.894	89.6	0.902
Louvain	90.3	0.906	91.6	0.918	90.9	0.912
METIS-10	90.7	0.91	92	0.922	90.3	0.908
METIS-20	91.6	0.918	91.9	0.92	91.5	0.918
Node2vec + k-means (10)	91	0.912	91.2	0.914	91.2	0.914
Node2vec + k-means (20)	91.1	0.913	91.6	0.918	91.4	0.916
Node2vec + k-means (50)	91	0.913	90.4	0.907	91.1	0.914
Clique (ours)	**93.7**	**0.938**	**95.7**	**0.957**	**94.5**	**0.946**

quasi-cliques can potentially yield larger overlapping structures that can capture node dependencies at a larger scale than regular cliques. We pose the experimental exploration of quasi-cliques as a future step to the work in this paper. Another interesting direction to explore is learning representations in a inductive manner using clique structures. For example, the representation of a new node can be derived as a function of the representations of other nodes that appear in the same clique. For graphs without node features or models that must be re-trained for new data, this provides a fast and efficient way to learn representations for new nodes without re-training. It is also of interest to see how such an approach can be applied in conjunction with existing inductive GNN representation learning approaches like GraphSAGE.

Our paper demonstrates the benefits of utilizing higher-order connection information in learning rich latent representations. We hope that researchers and practitioners alike can find inspiration to build models capturing this rich structural information and perform research in this direction.

REFERENCES

[1] Monica Agrawal, Marinka Zitnik, and Jure Leskovec. 2018. Large-scale analysis of disease pathways in the human interactome. In *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2018: Proceedings of the Pacific Symposium*. World Scientific, 111–122.

[2] Martin S Andersen, Joachim Dahl, Lieven Vandenberghe, et al. 2013. CVXOPT: A Python package for convex optimization. *Available at cvxopt. org* 54 (2013).

[3] 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 (2008), P10008.

Table 5: Performance comparison for different techniques on the Movie-Movie (1M) dataset.

Technique	GAE		GCN		DGI		GraphSAGE	
	Test acc.	F1 score	Test acc.	F1 score	Test acc.	F1 score	Test acc.	F1 score
No label	0.907	0.181	90.7	0.098	90.6	0.098	90.6	0.099
Louvain	89.8	0.157	90.7	0.091	90.7	0.088	90.81	0.094
METIS-10	90.7	0.183	90.8	0.102	90.5	0.104	90.65	0.106
METIS-20	90.2	0.161	90.7	0.133	90.7	0.105	90.71	0.106
Node2vec + k-means (10)	89.8	0.176	**90.9**	0.075	90.8	0.113	90.85	0.118
Node2vec + k-means (20)	89.5	0.165	90.8	0.096	90.7	0.119	90.74	0.121
Node2vec + k-means (50)	90.3	0.186	90.6	0.096	**90.9**	0.115	90.88	0.119
Clique (ours)	**91**	**0.194**	90.8	**0.136**	90.8	**0.138**	**90.91**	**0.123**

[4] Sergiy Butenko and Wilbert E Wilhelm. 2006. Clique-detection models in computational biochemistry and genomics. *European Journal of Operational Research* 173, 1 (2006), 1–17.

[5] Luís Cavique, Armando B Mendes, and Jorge MA Santos. 2016. Clique communities in social networks. In *Quantitative Modelling in Marketing and Management.* World Scientific, 469–490.

[6] Alexander Domahidi, Eric Chu, and Stephen Boyd. 2013. ECOS: An SOCP solver for embedded systems. In *2013 European Control Conference (ECC).* IEEE, 3071–3076.

[7] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining.* 855–864.

[8] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems.* 1025–1035.

[9] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. *IEEE Data Eng. Bull.* 40, 3 (2017), 52–74. http://sites.computer.org/debull/A17sept/p52.pdf

[10] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[11] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web.* 507–517.

[12] Dorit S Hochbaum. 1998. Approximating clique and biclique problems. *Journal of Algorithms* 29, 1 (1998), 174–200.

[13] Paul Jaccard. 1901. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bull Soc Vaudoise Sci Nat* 37 (1901), 547–579.

[14] Wang-Cheng Kang, Eric Kim, Jure Leskovec, Charles Rosenberg, and Julian McAuley. 2019. Complete the look: Scene-based complementary product recommendation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 10532–10541.

[15] Richard M Karp. 1972. Reducibility among combinatorial problems. In *Complexity of computer computations.* Springer, 85–103.

[16] George Karypis and Vipin Kumar. 1997. METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. (1997).

[17] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[18] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[19] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* ACM, 1269–1278.

[20] Ming-Xia Li, Wen-Jie Xie, Zhi-Qiang Jiang, and Wei-Xing Zhou. 2015. Communication cliques in mobile phone calling networks. *Journal of Statistical Mechanics: Theory and Experiment* 2015, 11 (2015), P11007.

[21] Zhenqi Lu, Johan Wahlström, and Arye Nehorai. 2018. Community detection in complex networks via clique conductance. *Scientific reports* 8, 1 (2018), 1–16.

[22] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[23] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. 2020. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation* 12, 4 (2020), 637–672. https://doi.org/10.1007/s12532-020-00179-2

[24] Mengying Sun, Sendong Zhao, Coryandar Gilvary, Olivier Elemento, Jiayu Zhou, and Fei Wang. 2020. Graph convolutional networks for computational drug development and discovery. *Briefings in bioinformatics* 21, 3 (2020), 919–935.

[25] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations.*

[26] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep graph infomax. *arXiv preprint arXiv:1809.10341* (2018).

[27] Rui Wang, Bicheng Li, Shengwei Hu, Wenqian Du, and Min Zhang. 2019. Knowledge graph embedding via graph attenuated attention networks. *IEEE Access* 8 (2019), 5212–5224.

[28] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. 2017. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining.* 555–564.

[29] Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. 2021. High-order structure exploration on massive graphs: A local graph clustering perspective. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15, 2 (2021), 1–26.

[30] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.