CC-GNN: A Community and Contraction-based Graph Neural Network

Zhiyuan Li*, Xun Jian†, Yue Wang‡ and Lei Chen§
*†The Hong Kong University of Science and Technology
‡Shenzhen Institute of Computing Sciences
§Data Science and Analytics Thrust, The Hong Kong University of Science and Technology (Guangzhou)
zlicw@cse.ust.hk, xjian@cse.ust.hk, yuewang@sics.ac.cn, leichen@ust.hk

2022 IEEE International Conference on Data Mining (ICDM) | 978-1-6654-5099-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDM54844.2022.00033

***Abstract*—Graph Neural Networks (GNNs) have attracted much research interest due to their successful applications on graph-structured data. Despite the effectiveness, due to the data dependency, GNNs are confronted with the neighborhood explosion and over-smoothing problems. Many sampling-based and level-down methods have been proposed to solve the neighborhood explosion problem to boost efficiency. However, they suffer from either poor accuracy or considerable runtime overheads. Moreover, the over-smoothing problem prevents GNNs from exploring more distant neighborhoods effectively. In this paper, we present a Community-and-Contraction-based Graph Neural Network (CC-GNN), which leverages community and contraction to boost the time and space efficiency of GNNs. Specifically, CC-GNN first performs contraction and retrieves the communities as super nodes, and connects them using a tailored similarity function to obtain an informative community contracted graph (CC-Graph). CC-GNN then learns the representations of the super nodes in the CC-Graph, which are used to reconstruct the representations of the original nodes. Finally, CC-GNN explores more distant neighborhoods without additional convolution layers, implicitly alleviating the over-smoothing problem. Since CC-GNN conducts the costly training on a much smaller contracted graph, the efficiency is boosted significantly. Most importantly, we have proved that the information loss of node representations caused by the CC-Graph is bounded. Extensive experimental studies verify the efficiency boost and the effectiveness.**

***Index Terms*—Graph Neural Network; Node Classification.**

I. Introduction

Graph Neural Networks (GNNs) are gaining popularity due to their successes in applications on social analysis [1], web recommendation [2], knowledge graph-based question answering [3], etc. The expressiveness of GNNs rises from effectively aggregating neighborhood information for each node. However, utilizing the graph structure increases the difficulty of scaling the model to larger datasets and training deeper models with desirable time and space efficiency because the neighborhood aggregation in GNNs breaks the data independence assumption used in other models. Hence, the vanilla GCN [4] uses a full-batch approach by storing the entire graph in a GPU. However, due to the potentially huge size of the graph and the feature dimension, the GPU memory capacity may fail to train large graphs such as Reddit [5] with 233K nodes and 115M edges.

To tackle the **scalability and efficiency** issues, many *mini-batch sampling-based* approaches [6]–[9] have been proposed. They aggregate only selected samples instead of all the neighborhoods. However, these sampling-based methods come with the following shortcomings by nature. Firstly, sampling is conducted online since different epochs require independently sampled neighborhoods. This incurs time overheads and leads to low utilization of GPUs as the sampling process dominates the entire training process [10]–[13]. Secondly, sampling methods can incur high variance which leads to slower convergence and worse accuracy [13], [14]. Lastly, sampling methods might suffer from the neighborhood explosion problem with larger graphs and deeper models, incurring significant overhead [8]. An alternative approach approximates the information diffusion in GNNs with designed propagation rules like personalized PageRank and pre-computes the propagation which is used to boost the training efficiency [15]–[17]. However, they are applicable to a limited number of GNN architectures and cannot retain the full expressive power of GNNs [18].

Recently, there are two concurrent works considering *level-down* approaches where GNNs are trained on a contracted graph for better scalability and efficiency. Huang et al. [18] leverage graph coarsening techniques in [19] to construct a contracted graph offline with high spectral similarity to the original graph and train the GNN on the contracted graph. Users need to input a reduction ratio which determines the size of the resulting contracted graph. Even though the authors provide theoretical guarantees, the model performance is sensitive to the reduction ratio. In some cases, the accuracy could drop by half when the size of the contracted graph is reduced by half [20]. Jin et al. [20] propose GCOND which borrows gradient matching to learn a smaller synthetic GNN-specific graph for training. Although GCOND is less sensitive to different reduction ratios, it requires the synthetic graph to be extremely small (~1K nodes) to fit in the GPU memory. Furthermore, the construction of the synthetic graph is online because it depends on the underlying GNN model and the downstream task. This can incur significant overhead for training different models for various tasks on the same input graph. In addition, GCOND requires user inputs of numbers of inner and outer loops for parameter updates to generate the synthetic graph in each epoch, which is difficult to determine and is not covered in the paper yet significantly affects the convergence and efficiency. Moreover, both methods could suffer from poor or inexplicable semantic data meaning of the super nodes and their connections

Xun Jian† and Yue Wang‡ are the correspondence authors.

978-1-6654-5099-7/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDM54844.2022.00033

in the contracted graph, which could lead to much information loss and degraded accuracy.

Another challenge for GNNs is the dilemma over how to explore neighborhoods at further hops away without causing **over-smoothing**. It is believed that GNNs are effective due to Laplacian smoothing [21]. The smoothing effect, however, will be enhanced with more layers, leading to indistinguishable representations of rather different nodes [21]. This dilemma restricts GNNs from exploring distant neighborhoods through more layers. Strategies including normalization [22] and distillation [23] have been proposed to address this, but they increase the online computation workload and are not desirable for efficiency.

To address the challenges and the shortcomings of existing works, we present CC-GNN, which leverages Community and Contraction to boost the time and space efficiency of Graph Neural Networks. Our goal is to construct a semantically meaningful contracted graph offline on which GNN models are applied with limited information loss incurred by the contraction. To achieve this, we leverage the homophily [24] property which states that nodes with similar features tend to be connected and share a large portion of the neighborhoods. Nodes in our contracted graph represent groups of similar nodes in the original graph, and their connections represent the similarity and interaction among different groups. In this way, we extend the information propagation from node-wise to community-wise. As an analogy with community networks, node-wise propagation means asking direct friends for the characteristic of the target person, while community-wise propagation means resorting to the community tags. Hence, CC-GNN can explore more distant graph information without adding more layers which might cause the over-smoothing problem. We observe homophily in many GNN datasets including citation, community, and product co-purchase networks because authors with similar research interests tend to collaborate, similar people will form a community, and similar products are typically purchased together. We also verify it by experiments in Appendix A.

The workflow of CC-GNN is shown in Figure 1. First, community detection and contraction is conducted on the original graph to retrieve the community membership and generate super nodes. Then, the Community Contraction Graph (CC-Graph) is built by rebuilding the adjacency between super nodes using our designed metrics. After that, representation learning is performed on the much smaller CC-Graph. Finally, we reconstruct node representations according to the community memberships. We also perform extensive experiments to evaluate the effectiveness and efficiency of CC-GNN. To summarize, we make the following contributions in this work.

- *High efficiency*. With CC-GNN, costly training operations are performed on a much smaller contracted graph, significantly reducing computation and memory costs.
- *Extended information propagation*. The information propagation is extended from node-wise to community-wise convolution in CC-GNN. With the same number of layers, CC-GNN can explore more distant graph structures. Therefore, a shallow CC-GNN can explore a comparable amount of information as a deeper version of others, alleviating the over-smoothing problem implicitly.
- *Limited information loss*. We prove the information loss for CC-GNN is bounded given the homophily property.

II. Related Work

We review the existing related works on speeding up GNN training and explain the difference of CC-GNN.

Sampling-based methods are the most popular way to address the GNN efficiency issue. They can be further divided into node-level, layer-level, and subgraph-level [14]. Subgraph-level sampling is related to CC-GNN in that it constructs the mini-batches as subgraphs. However, it propagates information node-wise while CC-GNN treats subgraphs as super nodes and propagate information community-wise. The sampling methods also have limitations in terms of effectiveness and efficiency as discussed in Section I.

Level-down methods are proposed recently to perform training on a contracted graph. Harp [25] uses collapse techniques to reduce the graph size while preserving the first and second order proximity for traditional graph embedding like random walk. However, it does not leverage communities and does not consider GNN training in the semi-supervised setting. Recently, there are two concurrent level-down works. Huang et al. [18] leverage graph coarsening techniques in [19] to construct a contracted graph offline which has high spectral similarity to the original graph. Jin et al. [20] proposes GCOND which borrows gradient matching to produce a smaller synthetic GNN-specific graph for training. However, the super nodes and their connections in their contracted graphs have poor or inexplicable semantic meaning which could lead to degraded accuracy. In contrast, CC-GNN tailors the construction of the super nodes and their connections to mimic information flow among different communities and extend the node-wise propagation to community-wise.

Graph pooling methods [26], [27] also produce a contracted graph with smaller size by supervised learning of node grouping [27] or eigen-decomposition [26]. However, they are designed for graph classification tasks and the contracted graphs are a byproduct graph in their pipeline [20]. Furthermore, the pooling methods are not designed for better efficiency since they incur extra computation cost for the pooling during online training. In contrast, CC-GNN contracts the graph offline and can be reused by different tasks.

III. Preliminaries

Graphs. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{Y})$ be an undirected, featured and labelled graph, where $\mathcal{V}$ is the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, $\mathcal{X}$ is the set of node features and $\mathcal{Y}$ is the set of node labels. The undirected edge between node u and v is denoted as $e = (u, v) \in \mathcal{E}$. The feature vector of a node $v \in \mathcal{V}$ is denoted as $x_v \in \mathcal{X}$ and the label is denotes as $y_v \in \mathcal{Y}$. The 1-hop neighborhood of a node v is denoted as $N(v) = \{u : (u, v) \in \mathcal{E}\}$.

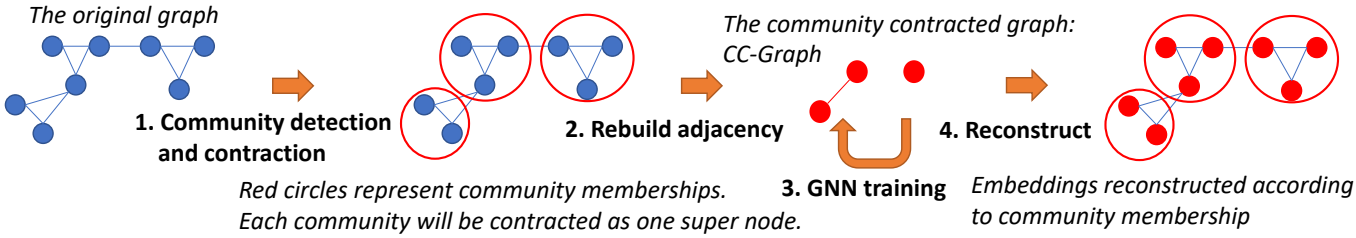


Fig. 1: Illustration of CC-GNN.

Graph Neural Networks. Denote the representations of the vertex $v \in \mathcal{V}$ at layer l as h_v^l. The node-wise update can be generalized as $h_v^l = Aggregate(h_v^{l-1}, \{h_u^{l-1} : u \in N(v)\})$. The $Aggregate(\cdot)$ function is any aggregation function from GNN models. For instance, it can be defined as

$$h_v^l = \sigma(W_l \sum_{u \in N(v)} \frac{h_u^{l-1}}{|N(v)|} + B_l h_v^{l-1}), \tag{1}$$

where $\sigma(\cdot)$ is the activation function (typically ReLU for GCN), W_k and B_k are learnable parameters.

IV. A Community and Contraction-based GCN

In this section, we present the details of our CC-GNN. We first retrieve grouped nodes as super nodes in the contracted graph. We then construct our CC-Graph by tailoring the connections among the super nodes with rich semantic meaning, which is our major contribution. The representation learning is applied on the CC-Graph followed by a reconstruction step.

A. Contraction and community detection

This step contracts the input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to a smaller set of nodes $\mathcal{V}'$ while each super node $V \in \mathcal{V}'$ represents a group of closely related nodes in the original graph for further processing. The step is further divided into two stages.

Stage I: graph contraction. We define a *Contracted Graph*.

Definition IV.1 (Contracted Graph)**.** Given a graph $\mathcal{G}$, a contraction mapping $\mathcal{M}$ from $\mathcal{V}$ to a set of super nodes $\mathcal{V}'$, and a binary adjacency rebuilding function $adj : \mathcal{V}' \times \mathcal{V}' \to \{0, 1\}$, a contracted graph $\mathcal{G}'$ is $(\mathcal{V}', \mathcal{E}')$ where $\mathcal{V}' = \mathcal{M}(\mathcal{V})$ and $\mathcal{E}' = \{(U, V) : U, V \in \mathcal{V}' \text{ and } adj(U, V) = 1\}$. The contracted graph $\mathcal{G}'$ merges nodes in $\mathcal{G}$ into super nodes according to the mapping $\mathcal{M}$. The adjacency of the super nodes is rebuilt according to adj. The contraction mapping $\mathcal{M}$ and the adjacency rebuilding function adj are called the contraction scheme.

Line collapse, edge collapse and star collapse shown in Figure 2 are popular contraction schemes [25].

Line collapse. It is proposed to address the possible long bridge between two communities. The blue nodes in Figure 2a forms a degenerated case of a linked list. They might have null community membership and occur in the contracted graph as one single super node. Such super nodes contain much less information than the super nodes representing a community, leading to biased distribution of information richness in the contracted graph. Hence, it requires line collapse for preprocessing.

Edge collapse. Edge collapse is a popular method for progressive meshes [28] and is also used for graph contraction [25]. The nature of progressive meshes and GCNs are both smoothing [21]. This serves as a motivation for us to perform the edge collapse. Specifically, based on an edge $e = (u, v)$, the incident vertices u and v will be contracted to one single node w whose incident edges are the union of those of u and v, as is illustrated in Figure 2b.

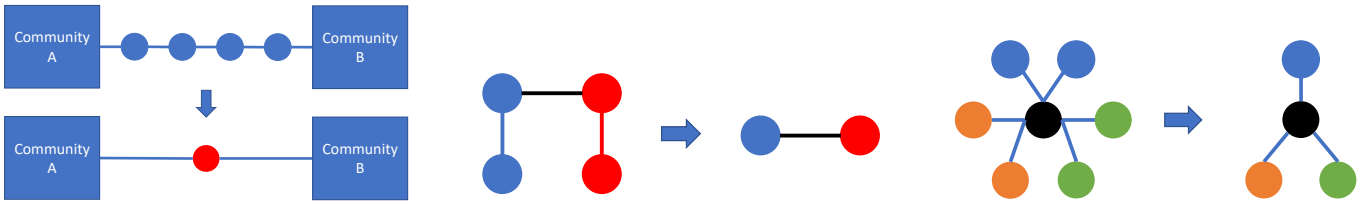
Star collapse. This is a subgraph-level contraction to address the shortcoming of edge collapse that it fails to handle the ubiquitous star-like structures in graphs as illustrated in Figure 2c. It is also used for graph contraction in [25].

We first perform star collapse followed by edge collapse because edge collapse cannot deal with star-like structures [25]. Following this, line collapse is performed to eliminate linked-list-like structures. We conduct each collapse method with one iteration alone since this stage serves as a warm-up and supporting role to eliminate undesired structures for the following community contraction stage which compresses the graph in a more significant and semantically meaningful way.

Stage II: community detection. After stage I, we retrieve a reasonably smaller graph on which community detection will be performed. We can allow overlapping or non-overlapping communities. We propose two choices of the community detection algorithm.

Community detection based on network connectivity. Label Propagation Algorithm (LPA) [29] propagates labels of nodes recursively to their neighbors and updates the labels to be the majority until convergence. Finally, each node will be assigned with a label which is used for non-overlapping community assignment. The label propagation resembles the information aggregation in GNNs, which raises our particular interest in LPA for our problem. Besides, LPA is very efficient with linear time complexity which has reasonable preprocessing cost that is cost-efficient. However, we cannot control the number and size of detected communities with LPA because the result depends on the convergence. Considering this, the resulting contracted graph might be too small for multi-layer convolution. One potential solution is to further divide the communities while this problem opens up our selection on the following community detection algorithm.

Community detection based on subgraph structures. The Clique Percolation Methods (CPM) [30] are a popular class of



(a) Line collapse (b) Edge Collapse (c) Star collapse

Fig. 2: Graph contraction

algorithms for overlapping community detection which are based on the k-cliques. A k-clique is defined as a subset of vertices of which each pair is adjacent. Two k-cliques are defined to be reachable if they share at least $k-1$ vertices. The reachability can be transitive. A community is just a maximal set of k-cliques that are reachable among each other. By tuning parameter k, we can control the number and size of communities. In addition, CPM can find group of nodes sharing similar set of neighbors which motivates our definition on structural equivalence in Definition IV.4. However, CPM is rather inefficient and might fail to terminate on large graphs.

B. Construct the CC-Graph

After retrieving the communities as the super nodes $\mathcal{V}'$, we construct a semantically meaningful contracted graph, called *community contracted graph* (CC-Graph) as defined below.

Definition IV.2 (Community Contracted Graph)**.** Given a graph $\mathcal{G}$, its community membership $\mathcal{CM}$, a similarity function $sim(\cdot,\cdot)$ and a predefined threshold δ, a Community Contracted Graph, CC-Graph, is the contracted graph which merges nodes in the same community into one super node in the CC-Graph according to the mapping $\mathcal{CM}$, and rebuilds the adjacency of the super nodes according to the similarity between pairs of communities if the similarity score $sim(\cdot,\cdot)$ is no less than δ.

In order to build such a CC-Graph, we need to define the adjacency of them $\mathcal{E}' = \{(u,v) : u,v \in \mathcal{V}'\}$ meaningfully and efficiently. We propose an edge-based similarity function to calculate the similarity scores for pairs of communities which can either be overlapping or not. Denote $C_1, C_2 \subseteq \mathcal{V}$ as two communities. Their similarity score is defined in Eq. 2.

$$sim(C_1, C_2) = \frac{|\{e = (u,v) : u \in C_1, v \in C_2\}|}{|C_1 \cup C_2|}. \quad (2)$$

If $sim(C_1, C_2) \geq \delta$, where δ is a fixed threshold, then the super nodes in CC-Graph that represent communities C_1 and C_2 are set to be adjacent. The edge between them indicates that they have a rather strong bond and influence on each other, which is useful for information propagation. The denominator in the definition servers as a normalization term to deal with communities of different sizes. The choice of δ will be examined in Section V-B.

1) Information loss of a CC-Graph: We now prove the information loss incurred by the CC-Graph is limited. Before that, we first define pairwise structural equivalence in Definition IV.3 and extend it to community level in Definition IV.4.

Definition IV.3 (Pairwise structural equivalence)**.** Two nodes $u, v \in \mathcal{G}$ are defined to be structurally equivalent with the tolerance of $\epsilon \in [0,1]$ if $\frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} \geq 1 - \epsilon$.

Definition IV.4 (Community-level structural equivalence)**.** A group of nodes $C \subseteq \mathcal{G}$ are defined to be structurally equivalent with the tolerance of $\epsilon \in [0,1]$ if $\frac{|\bigcap_{v \in C} N(v)|}{|\bigcup_{v \in C} N(v)|} \geq 1 - \epsilon$.

Structural equivalence was also defined in [31] but their definition was too strict for GCNs which accept certain fault tolerance. In contrast, our definition can accommodate some errors which is more practicable. By the homophily property discussed in Section I, we assume the proximity in the structure space in Definition IV.3 can infer proximity in the feature space. Under this assumption, the community contraction will incur limited information loss as proved in Theorem IV.1.

Theorem IV.1 (Limited information loss)**.** *Let $C \subseteq \mathcal{V}$ be the community of nodes and represent the super node in the contracted graph with feature $x_C = \frac{1}{|C|}\sum_{u \in C} x_u$ and neighbors $N(C) = \bigcup_{v \in C} N(v)$. The members of C are structurally equivalent with the tolerance of ϵ as defined in Definition IV.4 and are also similar in the feature space, i.e. $\max_{u \in C} \|x_C - x_u\| \leq \Delta$. The update rule of GNN is decomposed as $h_v = \sigma\left(Aggregate(v, N(v))\right) = F(x_v) + \frac{1}{|N(v)|}\sum_{u \in N(v)} G(x_u)$, where $F(\cdot)$ is L-Lipschitz and $G(\cdot)$ is bounded by M. Consider the contracted super node C with one layer of convolution, we have that the difference between the final representation of any node using the CC-Graph and that using the original graph is bounded by $2M\epsilon + L\Delta$.*

The proof is in Appendix G. As a remark, because the aggregation function normally involves only matrix multiplications, the values M and L are indeed matrix norms. Take Equation 1 as an example, assume σ is identity, then $M = \|W_l\| \cdot \max_{u \in V} \|h_u\|$ and $L = \|B_l\|$. Hence, the bound can be tightened using normalization techniques.

C. Representation learning on the CC-Graph

With the super nodes $\mathcal{V}'$ and their adjacency $\mathcal{E}'$ determined, we still need to construct the node features $\mathcal{X}'$ and labels $\mathcal{Y}'$ to retrieve the final contracted graph $\mathcal{G}'$. Denote $V \in \mathcal{V}'$ as both the super node in $\mathcal{G}'$ and $V \subseteq \mathcal{V}$ as the community of

nodes in $\mathcal{G}$. The feature and label of any super node $V \in \mathcal{V}'$ is set according to Equation 3.

$$x_V = \frac{1}{|V|}\sum_{v\in V} x_v; \quad y_V = Majority(\{y_v : v \in V\}) \tag{3}$$

In case of multi-label classification, set $y_V = \frac{1}{|V|}\sum_{v\in V} y_v$. Note that in case of single-label classification, the mode function and the mean function share the same final output. We then perform any existing GCN models on the CC-Graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \mathcal{X}', \mathcal{Y}')$ to retrieve the community representations $\mathcal{H}' = \{h_V : V \in \mathcal{V}'\}$.

D. Reconstruction for node representations

Denote the community membership of node v as $\mathcal{CM}(v) = \{V \in \mathcal{V}' : v \in V\}$, which is the set of communities to which v belongs. The reconstructed representation h_v of v is given by Eq. 4. The effectiveness of this straightforward approach is proved in Theorem IV.1 which bounds the difference between the community representation and the original node representation. Alternatively, the reconstruction can be refined by additional training restricted inside each community using the learnt matrices in the previous step as initialization. However, the trade-off sacrifices efficiency and the accuracy again is not significant.

$$h_v = \frac{1}{|\mathcal{CM}(v)|}\sum_{V\in\mathcal{CM}(v)} h_V \tag{4}$$

Complexity analysis. The time and space complexity of training CC-GNN is $O(L\left\|A'\right\|_0 F + nLF^2)$ and $O(nLF + LF^2)$, respectively, where L is the number of layers, F is the feature dimension, $n \ll N$ is the number of nodes of the contracted graph, and $\left\|A'\right\|_0 \ll \left\|A\right\|_0$ is the number of nonzeros in the adjacency matrix A' of the contracted graph. The space complexity can be further reduced to $O(b'LF+LF^2)$ by mini-batch training, where b' is the batch size.

V. Experiments

A. Experimental Setup

Datasets. We evaluate CC-GNN on node classification tasks on the following datasets: (1) citation network Cora; (2) citation network Pubmed; (3) community network Reddit; (4) product co-purchase network ogbn-products (OGBN); (5) citation network OGB10M. The datasets Cora, Pubmed and Reddit are collected from Deep Graph Library (DGL) [5], while OGBN is collected from Open Graph Benchmark (OGB) [32]. The first four datasets are commonly used in the literature to benchmark the effectiveness and efficiency for node classification tasks. Additional large scale experiments are performed on a graph containing 10M nodes (OGB10M) sampled from the ogbn-papers100M dataset from OGB. We generate a total of 160 different labels for all the sampled nodes in OGB10M since the ogbn-papers100M dataset has little labelled data. The statistics and sources of the datasets are summarized in Table I.

TABLE I: Statistics of Graph Datasets.

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	Num. of classes	Feature dim.	Source
Cora	2,708	10,556	7	1,433	DGL[1]
Pubmed	19,717	88,651	3	500	DGL[2]
Reddit	232,965	114,615,892	41	602	DGL[3]
OGBN	2,449,029	123,718,280	47	100	OGB[4]
OGB10M	10,000,000	13,436,086	160	128	OGB[5]

Competitors. We compare CC-GNN with eight popular competitors: (1) Vanilla GCN [4]: a full-batch training method; (2) GraphSAGE [6]: a node-level sampling method; (3) FastGCN [7]: a layer-level sampling method; (4) Cluster-GCN [8]: a subgraph-level sampling method; (5) GraphSAINT [9]: a SOTA subgraph-level sampling method; (6) GBP [17]: a SOTA work for scalability; (7) GCoarsen [18]: graph coarsening for GNNs; (8) GCOND [20]: graph condensation for GNNs. We implement CC-GNN using vanilla GCN [4] for training on the CC-Graph. The default parameters for all the models are 2 layers, hidden dimension of 16 for Cora and 128 for all other datasets, and a maximum of 100 epochs, which are common practices in the literature [4], [8]. For fair comparison with the two level-down approaches GCoarsen and GCOND, we use the same GNN model and reduction ratio, which suggests the same size of the contracted graph and the same level of efficiency, and compare the test accuracy for effectiveness. When GCOND encounters GPU OOM for the same reduction ratio, we use the largest possible reduction ratio that can fit in the GPU. For model specific settings, please refer to Appendix B.

Metrics. We measure effectiveness by the accuracy score in percentage. We benchmark time efficiency using average training time per epoch in seconds. Space efficiency is measured by (1) the maximal GPU memory usage in MB during training; and (2) VmHWM in GB which signifies the required amount of physical memory at peak during the entire execution.

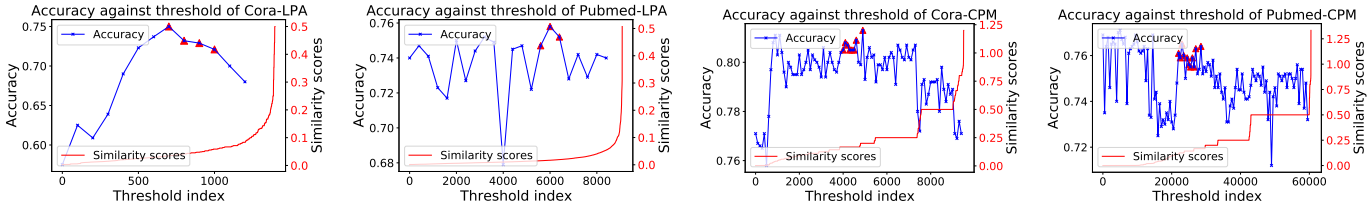
All experiments are deployed on a server with two Intel(R) Xeon(R) Silver 4210 CPUs, eight GeForce RTX 2080 Ti GPUs and 92 GB available memory. All the models use one single GPU except that vanilla GCN is trained on CPU for Reddit and OGBN due to OOM. Additional details on the community detection methods are in Appendix C. We will explore the modules and parameter settings of CC-GNN as ablation study in Section V-B and compare with SOTA models in Section V-C. Our implementation is at https://github.com/fr8nkL/CC-GNN.

B. Effectiveness and Efficiency of CC-GNN

We now study how the choices of the threshold value δ in Section IV-B for the similarity scores influence the effectiveness. The impact and choice of the community structure is studied in Appendix D. The efficiency is examined in Appendix E.

Varying threshold values for effectiveness. The impact of the threshold value δ in Section IV-B on the effectiveness is

[1]https://docs.dgl.ai/generated/dgl.data.CoraGraphDataset.html
[2]https://docs.dgl.ai/generated/dgl.data.PubmedGraphDataset.html
[3]https://docs.dgl.ai/generated/dgl.data.RedditDataset.html
[4]https://ogb.stanford.edu/docs/nodeprop/#ogbn-products
[5]https://ogb.stanford.edu/docs/nodeprop/#ogbn-papers100M



(a) Cora-LPA. (b) Pubmed-LPA (c) Cora-CPM. (d) Pubmed-CPM.

Fig. 3: Accuracy against similarity threshold index.

TABLE II: Comparison of test set accuracy. "$*$" means runtime error. "-" means the model does not terminate within 10 hours.

Dataset	Cora	Pubmed	Reddit	OGBN	OGB10M
Vanilla GCN	81.50 ± 0.60	76.40 ± 0.30	93.60 ± 0.68	70.65 ± 0.00	OOM
GraphSAGE	83.30 ± 0.40	75.85 ± 0.30	95.74 ± 0.34	77.02 ± 0.00	OOM
FastGCN	**84.80** ± 0.20	77.53 ± 0.47	93.05 ± 0.65	18.40 ± 1.00	OOM
Cluster-GCN	76.10 ± 0.00	74.50 ± 0.60	**96.60** ± 0.00	76.35 ± 0.26	OOM
GraphSAINT	29.75 ± 1.95	$*$	94.41 ± 0.11	69.50 ± 0.00	OOM
GBP	81.00 ± 0.00	**81.20** ± 0.00	92.30 ± 0.00	69.10 ± 0.00	50.20 ± 0.00
GCoarsen	70.27 ± 1.27	66.09 ± 2.87	-	-	-
GCOND	73.77 ± 3.44	69.90 ± 2.41	89.40 ± 0.70	53.64 ± 0.50	-
CC-GNN	82.16 ± 0.64	76.30 ± 0.34	90.10 ± 0.23	**77.11** ± 0.29	**50.24** ± 0.00

tested using Cora and Pubmed. We aim to derive a strategy from the two small datasets to quickly pinpoint an optimal threshold value for the larger datasets. We vary the threshold values to retrieve the corresponding accuracy scores on different datasets and plot the results in Figure 3, whose x-axis is the index of valid similarity scores of pairs of communities in ascending order which is chosen as the threshold. The discovered strategy is to set the elbow of the plots of the similarity scores as the threshold. For each subplot in Figure 3, the accuracy achieves a global maximum or near global maximum when the threshold is set near the elbow of the red plot of similarity scores (indicated by the red triangles). The underlying reason could be the strength of connection between communities. For those pairs of communities whose similarity scores are low, there are few inter-community edges compared to their sizes, hence the model can well tolerate the incurred information loss. Besides, a large dataset can have more than 10M valid positive similarity scores between pairs of communities. If the threshold δ is too low, then the CC-Graph will be too dense to distinguish the super nodes linked by too many meaningless and misleading edges. Hence, the accuracy is increasing with respect to the threshold value in general to the left of the elbow. Meanwhile, if the threshold is set too high, then the CC-Graph will be too sparse, incurring significant information loss. This explains the decreasing trend of the accuracy to the right of the elbow. Therefore, a good candidate value for the threshold should be near the elbow of the similarity scores.

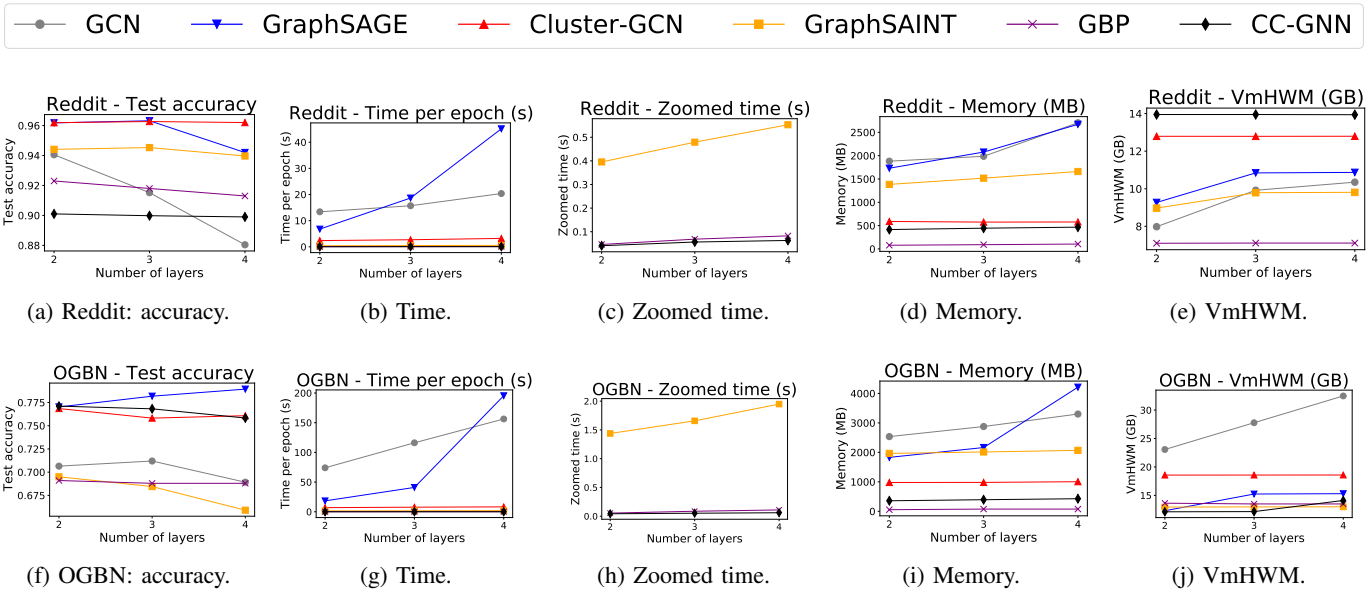
C. Comparison with State-of-the-Art

We compare the effectiveness and efficiency of CC-GNN with SOTA. We also evaluate the influence of varying number of layers to show how CC-GNN tackle the dilemma over how to explore distant neighborhoods without causing over-smoothing.

1) Effectiveness: We compare the effectiveness of CC-GNN with other models. The settings of CC-GNN follow Table VII.

Comparison of effectiveness with SOTA. Table II shows the accuracy comparison with SOTA. The accuracy of CC-GNN is on average 96% of the highest accuracy. For the large dataset OGBN, CC-GNN beats all the competitors. The satisfactory accuracy of CC-GNN gives empirical support to Theorem IV.1 and illustrates the power of the community structure in GNNs. However, the accuracy for Reddit is 6.5% below the best. The reason might be Reddit's extreme large average degree, causing the size of the union of the neighbors to be significantly larger than that of the join. Hence, the tolerance ϵ in Definition IV.4 is much larger, enlarging the error bound in Theorem IV.1. Notably, using the same contraction ratio which indicates the same level of training efficiency, CC-GNN consistently outperforms the other two level-down models GCoarsen and GCOND on all the tested datasets, especially on the large sparse OGBN dataset. We believe it is attributed to the different semantic richness of the contracted graph. By leveraging the community structures and tailoring the interactions among the communities, CC-GNN produces contracted graphs of richer semantic meaning and achieves better effectiveness. Additionally, vanilla GCN and FastGCN can adapt well to smaller datasets, while the Cluster-GCN and GraphSAINT are more suitable for larger ones. The reason might be that the former two models are sensitive to noises which are more prevalent in large graphs while the latter two models incur much information loss in small graphs due to subgraph-level sampling. Besides, the reported accuracy of GraphSAINT and GBP on OGBN is lower than that reported for Amazon due to the different data splits.

Varying number of layers. Further, we test the impact of the number of layers L on the effectiveness of the models



(a) Reddit: accuracy. (b) Time. (c) Zoomed time. (d) Memory. (e) VmHWM.

(f) OGBN: accuracy. (g) Time. (h) Zoomed time. (i) Memory. (j) VmHWM.

Fig. 4: Impact of number of layers.

TABLE III: Varying layers for CC-GNN and GCOND.

	Reddit			OGBN		
Num. of layers	2	3	4	2	3	4
GCOND	89.40	61.47	52.18	53.64	35.11	44.14
CC-GNN	**90.10**	**89.98**	**89.90**	**77.11**	**76.82**	**75.83**

with the two larger datasets Reddit and OGBN. FastGCN is excluded because the released code cannot naturally adapt to more layers. GCoarsen fails to terminate within 10 hours and is excluded. Figure 4a, 4f and Table III show the trend of accuracy scores of the models against the increasing L. In general, the trend is decreasing, revealing the impact of over-smoothing. Specifically, vanilla GCN and GCOND have severe accuracy drop when L increases, which demonstrates the over-smoothing problem. In contrast, the accuracy of CC-GNN remains stable. We believe that the stable performance of CC-GNN is again attributed to the semantically meaningful CC-Graph compared with GCOND. Moreover, on OGBN, the 2-layer CC-GNN of accuracy 77.11% is comparable with the 4-layer GraphSAGE of the highest accuracy 78.94% which incurs 4631x more training time per epoch. This demonstrates the effectiveness of CC-GNN's generalized information aggregation among communities which allows the model to explore more distant information without additional layers. It alleviates the over-smoothing problem implicitly since there is no need to pay extra cost to "go deep" during training.

2) Efficiency: With the effectiveness illustrated above, we aim to validate the time and space efficiency using the large datasets Reddit, OGBN and OGB10M. Since the same reduction ratio indicates the same level of efficiency for the level-down approaches, GCoarsen and GCOND are compared in terms of effectiveness alone using the same reduction ratio. We also show the varying efficiency as the numbers of layers varies. CC-GNN has reasonable preprocessing cost including the community contraction as shown in Appendix F.

Comparison of efficiency with SOTA. Table IV summarizes the model efficiency on Reddit and OGBN. CC-GNN achieves the best time efficiency and the second best GPU memory efficiency for all the datasets. Note that CC-GNN and GBP consume far less GPU memory. For the even larger dataset OGB10M, CC-GNN beats GBP in terms of GPU memory efficiency as shown in Table V. This highlights the efficiency gain of contracting the graph for training. Table IV also suggests that sampling algorithms from different classes have different power on efficiency boost. In general, the power increases from node to subgraph level. The two special methods, GBP and CC-GNN further boost the time and space efficiency by one order of magnitude. One remark is that CC-GNN has less memory usage and VmHWM on the larger dataset OGBN than the smaller Reddit. The possible reasons are: (1) the feature dimension of OGBN is 6 times smaller, hence less GPU memory usage; (2) OGBN has much fewer valid similarity scores, hence smaller VmHWM. The VmHWM of GraphSAINT and GBP is smaller than other models because they resort to C++ for preprocessing which is more memory efficient than Python. CC-GNN can achieve even better memory efficiency by migrating our Python code for preprocessing to C++.

We further compare the efficiency on the large OGB10M dataset. Only GBP and CC-GNN are tested because others either encounter OOM issue or do not terminate within 10 hours. Table V summarizes the results. With hidden dimension of 128, CC-GNN consistently outperforms GBP for different number of layers in terms of training time and GPU usage. With the hidden dimension increased to 512, GBP encounters OOM for more than two layers, while CC-GNN can still run on a single GPU. This illustrates the strong scalability of CC-GNN.

TABLE IV: Comparison of time and space efficiency with two layers on Reddit and OGBN.

	Reddit			OGBN		
	Time	Mem	VmHWM	Time	Mem	VmHWM
Vanilla GCN	13.3871	1882	7.98	74.0441	2535	23.07
GraphSAGE	6.7667	1731	9.27	18.3628	1831	12.30
FastGCN	7.1585	1003	16.90	11.6464	1176	19.90
Cluster-GCN	2.3816	589	12.79	6.9433	978	18.57
GraphSAINT	0.3955	1384	8.97	1.4377	1965	12.90
GBP	0.0461	**79**	**7.10**	0.0535	**55**	13.63
CC-GNN	**0.0407**	140	13.95	**0.0422**	102	**12.12**

TABLE V: Comparison of time and space efficiency on OGB10M.

	Time per epoch (s)						GPU Memory (MB)					
Hidden dim.	128			512			128			512		
Num. of layers	2	3	4	2	3	4	2	3	4	2	3	4
GBP	1.8672	2.3999	2.9415	1.8933	OOM	OOM	3174	3907	3907	8791	OOM	OOM
CC-GNN	**0.4366**	**0.0559**	**1.1091**	**1.0342**	**1.1096**	**2.0474**	**562**	**699**	**837**	**1436**	**1981**	**2526**

Varying number of layers. Further, we test the impact of increasing number of layers L on the efficiency of different models using the two large datasets, Reddit and OGBN. The previous parameter setting is followed. The results are shown in Figure 4. CC-GNN consistently achieves the best performance on the training time per epoch and second best performance on the required amount of memory during training. The VmHWM measure on Reddit is poor, whose reason might be its huge number of valid similarity scores of pairs of communities. We also observe that CC-GNN has linear increase of the time and space complexity with respect to L. In contrast, GraphSAGE suffers the most from the increasing L, demonstrating an exponential growth of time and space complexity. Specifically, the four-layer GraphSAGE on OGBN requires 195.4247 seconds per epoch for training. Even though the four-layer GraphSAGE achieves the highest accuracy of 78.94% for OGBN, the considerable training time requirement makes the four-layer GraphSAGE model too expensive for typical users striking for a balance between effectiveness and efficiency. In this case, the shallow version of CC-GNN which takes only 0.0422 seconds per epoch for training but achieves an accuracy of 77.11% becomes the most appropriate candidate.

Summaries: (1) CC-GNN on average consumes 132.2x less training time and 9.3x less GPU memory compared with the tested models; (2) The accuracy of CC-GNN is on average 96% of the highest accuracy, providing empirical support to Theorem IV.1; (3) on OGBN, a 2-layer CC-GNN of accuracy 77.11% requires 4631x less training time per epoch than the 4-layer GraphSAGE of the highest accuracy 78.94%, demonstrating that our extended information aggregation among communities can alleviate the over-smoothing problem implicitly.

VI. Conclusion

We present CC-GNN, a new graph training algorithm that leverages the community and contraction to boost the time and space efficiency of GNN models. We prove the limited information loss of our method and empirically demonstrate its effectiveness and efficiency through extensive experiments. The community contracted graph enables remarkable efficiency improvement and extends information propagation to community-level which achieves satisfactory accuracy and implicitly alleviates the over-smoothing problem.

Acknowledgement

Lei Chen's work is partially supported by National Key Research and Development Program of China Grant No. 2022YFE0200500 and 2018AAA0101100, the Hong Kong RGC GRF Project 16209519, CRF Project C6030-18G, C1031-18G, C5026-18G, CRF C2004-21GF, AOE Project AoE/E603/18, RIF Project R6020-19, Theme-based project TRS T41-603/20R, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Microsoft Research Asia Collaborative Research Grant, HKUST-NAVER/LINE AI Lab, Didi-HKUST joint research lab, HKUST-Webank joint research lab grants and HKUST Global Strategic Partnership Fund (2021 SJTU-HKUST). Yue Wang is partially supported by China NSFC (No. 62002235) and Guangdong Basic and Applied Basic Research Foundation (No. 2019A1515110473).

References

[1] C. Li and D. Goldwasser, "Encoding social information with graph convolutional networks for political perspective detection in news media," in *ACL*, 2019.

[2] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *SIGKDD*, 2018.

[3] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. Cohen, "Open domain question answering using early fusion of knowledge bases and text," in *EMNLP*, 2018.

[4] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *ICLR*, 2017.

[5] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.
[6] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017.
[7] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," in *ICLR*, 2018.
[8] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *SIGKDD*, 2019.
[9] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graph-saint: Graph sampling based inductive learning method," in *ICLR*, 2020.
[10] S. Gandhi and A. P. Iyer, "P3: Distributed deep graph learning at scale," in *OSDI*, 2021.
[11] Z. Lin, C. Li, Y. Miao, Y. Liu, and Y. Xu, "Pagraph: Scaling gnn training on large graphs via computation-aware caching," in *SoCC*, 2020.
[12] J. Yang, D. Tang, X. Song, L. Wang, Q. Yin, R. Chen, W. Yu, and J. Zhou, "Gnnlab: A factored system for sample-based gnn training over gpus," in *EuroSys*, 2022.
[13] J. Thorpe, Y. Qiao, J. Eyolfson, S. Teng, G. Hu, Z. Jia, J. Wei, K. Vora, R. Netravali, M. Kim, and G. H. Xu, "Dorylus: Affordable, scalable, and accurate GNN training with distributed CPU servers and serverless threads," in *OSDI*, 2021.
[14] W. Cong, R. Forsati, M. Kandemir, and M. Mahdavi, "Minimal variance sampling with provable guarantees for fast training of graph neural networks," in *SIGKDD*, 2020.
[15] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *ICML*, 2019.
[16] A. Bojchevski, J. Klicpera, B. Perozzi, A. Kapoor, M. Blais, B. Rózemberczki, M. Lukasik, and S. Günnemann, "Scaling graph neural networks with approximate pagerank," in *SIGKDD*, 2020.
[17] M. Chen, Z. Wei, B. Ding, Y. Li, Y. Yuan, X. Du, and J.-R. Wen, "Scalable graph neural networks via bidirectional propagation," in *NeurIPS*, 2020.
[18] Z. Huang, S. Zhang, C. Xi, T. Liu, and M. Zhou, "Scaling up graph neural networks via graph coarsening," in *SIGKDD*, 2021.
[19] A. Loukas, "Graph reduction with spectral and cut guarantees." *J. Mach. Learn. Res.*, vol. 20, no. 116, pp. 1–42, 2019.
[20] W. Jin, L. Zhao, S. Zhang, Y. Liu, J. Tang, and N. Shah, "Graph condensation for graph neural networks," in *ICLR*, 2022.
[21] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *AAAI*, 2018.
[22] L. Zhao and L. Akoglu, "Pairnorm: Tackling oversmoothing in gnns," in *ICLR*, 2020.
[23] B. Yan, C. Wang, G. Guo, and Y. Lou, "Tinygnn: Learning efficient graph neural networks," in *SIGKDD*, 2020.
[24] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology*, 2001.
[25] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, "Harp: Hierarchical representation learning for networks," in *AAAI*, 2018.
[26] Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang, "Graph convolutional networks with eigenpooling," in *SIGKDD*, 2019.
[27] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *NeurIPS*, 2018.
[28] H. Hoppe, "Progressive meshes," in *SIGGRAPH*, 1996.
[29] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, 2007.
[30] F. Reid, A. McDaid, and N. Hurley, "Percolation computation in complex networks," in *ASONAM*, 2012.
[31] J. Liang, S. Gurukar, and S. Parthasarathy, "Mile: A multi-level framework for scalable graph embedding," *arXiv preprint arXiv:1802.09612*, 2018.
[32] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," in *NeurIPS*, 2020.

APPENDIX

A. Homophily Property

We verify the homophily property by experiments. Table VI reports the average Δ in Theorem IV.1 and the average pairwise distance of node feature vectors for the entire graph. It shows that nodes in the same community have more similar features than other nodes, confirming the homophily property.

TABLE VI: Homophily property in GNN datasets.

Dataset	Avg. Δ in Theorem IV.1	Avg. pairwise distance
Cora	0.22831966	0.35847301
Pubmed	0.17903374	0.27096435
Reddit	19.84883856	32.25965134

B. Model Specific Settings

The links to the source code of the baselines can be found at https://github.com/fr8nkL/CC-GNN. For GraphSAGE, we use the default settings for neighborhood sampling with size 10 and 25 for two layers. We use the sampling sizes of 5, 10 and 25 for three layers and 5, 10, 15 and 25 for four layers to prevent the otherwise OOM issue. For Cluster-GCN, we follow the settings for the partition size and batch size in the original paper in the effectiveness comparison. In the efficiency comparison, we fix the batch size and scale the partition size to keep the number of batches per epoch comparable with other models. Edge sampling is used for GraphSAINT since it is efficient and effective according to the results in [9]. Other settings for FastGCN, GraphSAINT, GBP, GCoarsen and GCOND are the same as default. For CC-GNN, Table VII summarizes the community detection method, threshold settings, and the reduction ratios in percentage for different datasets that we use to compare with other models.

TABLE VII: Method of community detection, num. of similarity scores, threshold ID, and reduction ratio for CC-GNN.

Dataset	Method	# sim	ID	Ratio (%)
Cora	CPM	9,508	4,000	9.23
Pubmed	CPM	60,468	30,000	1.85
Reddit	LPA	20,718,936	20,663,000	6.67
OGBN	LPA	1,033,728	917,200	0.84
OGB10M	LPA	11,580,631	11,275,000	0.85

C. Community Detection Methods

For the community detection method CPM [30], we manually create a new community for each node without community membership for the CC-graph. We fix $k = 3$ for CPM for good community quality. We use unit weighting for LPA [29].

D. Effectiveness of CC-GNN

Varying community detection methods. The impact of the two community detection methods LPA [29] and CPM [30] on effectiveness (accuracy) is tested on Cora and Pubmed. Other larger datasets are not tested because CPM fails to terminate on them. For the highest accuracy score, CPM defeats LPA on Cora by 82.16% vs 75.00% with a margin of 7.16%. CPM also defeats LPA on Pubmed by 76.30% vs 75.80% with a margin of 0.5%. The reason might be the finer structural equivalence of the communities detected by CPM. Therefore, we choose CPM for Cora and Pubmed. For the other large datasets, we choose LPA for better efficiency of preprocessing.

E. Efficiency of CC-GNN

Varying threshold values. We test the training time per epoch under different threshold values using the Reddit dataset. The result is shown in Figure 5. Clearly, if the threshold is larger, then the training time per epoch will be faster due to the reduced number of edges in the CC-graph. Hence, in terms of efficiency, a larger threshold value is more desired. Meanwhile, the accuracy tends to decrease after the elbow. Hence, combining effectiveness and efficiency, we conclude that the elbow is the optimal setting for the threshold.

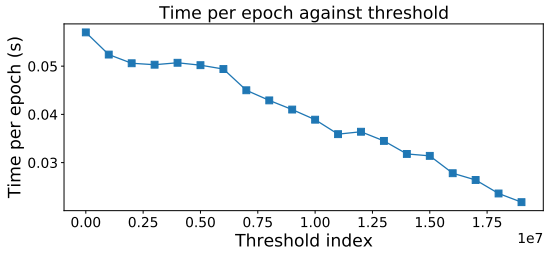


Fig. 5: Time per epoch against threshold for Reddit.

F. Total Preprocessing and Training Time

We compare the natural logarithm of total preprocessing and training time against the number of epochs of CC-GNN with other methods using the 2-layer setting on the OGBN dataset in Figure 6. The y-value at $x = 0$ is the preprocessing time. It shows that although the community detection incurs much overhead, it is compensated for by the training speedup with many training epochs. The community detection phase can be further accelerated using parallel and C++ implementations.

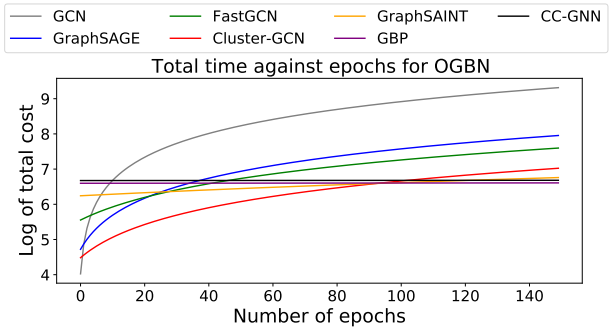


Fig. 6: Natural logarithm of total preprocessing and training time against number of epochs.

G. Proof of Theorem IV.1

Proof. The case of nodes outside of the community C is a degenerated case of nodes inside the community C because the contraction only affects their adjacency but not their features. Hence, we only consider the case of nodes inside the community C. Let $v \in C$ be an arbitrary node inside the community C. Denote its embedding before the activation using the original graph by h_v and its embedding before the activation using the contracted graph by h_C which is the community's embedding. Note that the activation function will only make their difference smaller. By structural equivalence, we have

$$|N(v)| \geq \left|\bigcap_{v\in C} N(v)\right| \geq (1-\epsilon)\left|\bigcup_{v\in C} N(v)\right| = (1-\epsilon)\,|N(C)| \tag{5}$$

Because F is L-Lipschitz, we have

$$\|F(x_C) - F(x_v)\| \leq L\,\|x_C - x_v\| \leq L \max_{u\in C} \|x_C - x_u\| \leq L\Delta \tag{6}$$

Therefore, we have

$$\begin{aligned}
&\|h_C - h_v\| \\
&= \left\| F(x_C) + \frac{\sum_{u\in N(C)} G(x_u)}{|N(C)|} - F(x_v) - \frac{\sum_{u\in N(v)} G(x_u)}{|N(v)|} \right\| \\
&\leq \left\| \frac{\sum_{u\in N(C)} G(x_u)}{|N(C)|} - \frac{\sum_{u\in N(v)} G(x_u)}{|N(v)|} \right\| + \|F(x_C) - F(x_v)\| \\
&\leq \left\| \frac{1}{|N(C)|} \sum_{u\in N(C)\setminus N(v)} G(x_u) \right\| \\
&\quad + \left\| \left(\frac{1}{|N(v)|} - \frac{1}{|N(C)|} \right) \sum_{u\in N(v)} G(x_u) \right\| + L\Delta \\
&\leq \frac{1}{|N(C)|} \sum_{u\in N(C)\setminus N(v)} \|G(x_u)\| \\
&\quad + \left(\frac{1}{|N(v)|} - \frac{1}{|N(C)|} \right) \sum_{u\in N(v)} \|G(x_u)\| + L\Delta
\end{aligned} \tag{7}$$

by Eq. 6 and triangle inequality. Now continue from Eq. 7. By G bounded by M and triangle inequality, we have

$$\begin{aligned}
&\|h_C - h_v\| \\
&\leq \frac{1}{|N(C)|} \times |N(C) \setminus N(v)| \times M \\
&\quad + \left(\frac{1}{|N(v)|} - \frac{1}{|N(C)|} \right) \times |N(v)| \times M + L\Delta \\
&= \frac{|N(C)| - |N(v)|}{|N(C)|} \times M + \left(1 - \frac{|N(v)|}{|N(C)|} \right) \times M + L\Delta \\
&= 2M \left(1 - \frac{|N(v)|}{|N(C)|} \right) + L\Delta \\
&\leq 2M\epsilon + L\Delta \text{ by Eq. 5}
\end{aligned} \tag{8}$$

□