Pre-train and Refine: Towards Higher Efficiency in K-Agnostic Community Detection without Significant Quality Degradation

Anonymous Author(s)

ABSTRACT

Community detection (CD) is a classic graph inference task that partitions nodes of a graph into densely connected groups. While many CD methods have been proposed with either impressive quality or efficiency, balancing the two aspects remains a challenge. This study explores the potential of deep graph learning to achieve a better trade-off between the quality and efficiency of K-agnostic CD, where the number of communities K is unknown. We propose PRoCD (Pre-training & Refinement for Community Detection), a simple yet effective method that reformulates K-agnostic CD as the binary node pair classification. PRoCD follows a *pre-training & refinement* paradigm inspired by recent advances in pre-training techniques. We first conduct the *offline pre-training* of PRoCD on small synthetic graphs covering various topology properties. Based on the inductive inference across graphs, we then *generalize* the pre-trained model (with frozen parameters) to large real graphs and use the derived CD results as the initialization of an existing efficient CD method (e.g., InfoMap) to further *refine* the quality of CD results. In addition to benefiting from the transfer ability regarding quality, the *online generalization* and *refinement* can also help achieve high inference efficiency, since there is no time-consuming model optimization. Experiments on public datasets with various scales demonstrate that PRoCD can ensure higher efficiency in K-agnostic CD without significant quality degradation.

KEYWORDS

Community Detection, Graph Clustering, Inductive Graph Inference, Pre-training & Refinement

ACM Reference Format:
Anonymous Author(s). 2018. Pre-train and Refine: Towards Higher Efficiency in K-Agnostic Community Detection without Significant Quality Degradation. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Community detection (CD) is a classic graph inference task that partitions nodes of a graph into several groups (i.e., communities) with dense linkage distinct from other groups [12]. It has been validated that the extracted communities may correspond to substructures of various real-world complex systems (e.g., functional groups in protein interactions [11]). Many network applications (e.g., cellular network decomposition [9] and Internet traffic profiling [30]) are thus formulated as CD. Due to the NP-hardness of some typical CD objectives (e.g., modularity maximization [24]), balancing the quality and efficiency of CD on large graphs remains a challenge. Most existing approaches focus on either high quality or efficiency.

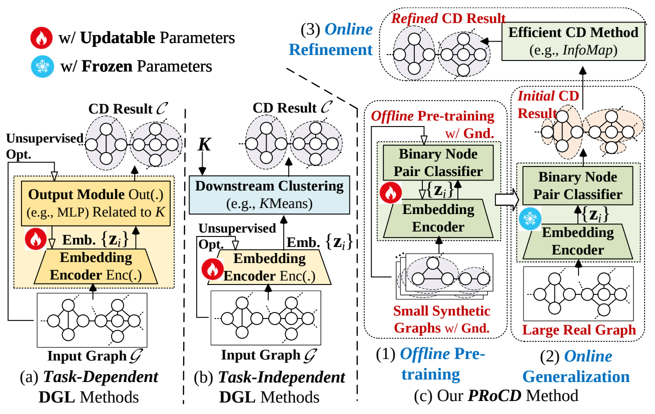


Figure 1: Overview of mainstream (a) *task-dependent* and (b) *-independent* DGL methods as well as (c) our PRoCD method (with a novel *pre-training & refinement* paradigm) for CD.

On the one hand, efficient CD methods usually adopt heuristic strategies or fast approximation w.r.t. some relaxed CD objectives to obtain feasible CD results (e.g., the greedy maximization [3] and semi-definite relaxation [48] of modularity). Following the graph embedding framework, which maps nodes $\mathcal{V} = \{v_i\}$ of a graph into low-dimensional vector representations $\{\mathbf{z}_i \in \mathbb{R}^d\}$ ($d \ll |\mathcal{V}|$) with major topology properties preserved, several efficient embedding approaches have also been proposed based on approximated dimension reduction (e.g., random projection for high-order topology [56]). Some of them are claimed to be community-preserving and able to support CD using a downstream clustering module [2, 13]. Despite the high efficiency, these methods may potentially suffer from quality degradation due to the information loss of heuristics, approximation, and relaxation.

On the other hand, recent studies have demonstrated the ability of deep graph learning (DGL) techniques [38, 57], e.g., those based on graph neural networks (GNNs), to achieve impressive quality of various graph inference tasks including CD [52, 55]. However, their powerful performance usually relies on iterative optimization algorithms (e.g., gradient descent) that direct sophisticated models to fit complicated objectives, which usually have high complexities.

In this study, we explore the potential of DGL to ensure *higher efficiency in CD without significant quality degradation*, compared with mainstream efficient and DGL methods. It serves as a possible way to achieve a better trade-off between the two aspects. Different from existing methods [6, 10, 46] with an assumption that the number of communities K is given, we consider the more challenging yet realistic K-agnostic CD, where K is unknown. One should simultaneously determine K and the corresponding CD result.

Dilemmas. As shown in Fig. 1 (a) and (b), existing DGL-based CD techniques usually follow the unified graph embedding framework and can be categorized into the (i) *task-dependent* and (ii) *task-independent* approaches. We argue that they may suffer from the following limitations w.r.t. our settings.

First, *existing DGL methods may be inapplicable to K-agnostic CD*. Given a graph $\mathcal{G}$, some *task-dependent* approaches [6, 46, 49] (see Fig. 1 (a)) generate embeddings $\{\mathbf{z}_i\}$ via an embedding encoder $\text{Enc}(\cdot)$ (e.g., a multi-layer GNN) and feed $\{\mathbf{z}_i\}$ into an output module $\text{Out}(\cdot)$ (e.g., a multi-layer perceptron (MLP)) to derive the CD result C, i.e., $\{\mathbf{z}_i\} = \text{Enc}(\mathcal{G})$ and $C = \text{Out}(\{\mathbf{z}_i\})$. Since parameters in $\text{Out}(\cdot)$ are usually with the dimensionality related to K, these *task-dependent* methods cannot output a feasible CD result when K is unknown. Although some *task-independent* methods [20, 33, 54] (see Fig. 1 (b)) do not contain $\text{Out}(\cdot)$ related to K, their original designs still rely on a downstream clustering algorithm (e.g., KMeans) with $\{\mathbf{z}_i\}$ and K as required inputs.

Second, *the standard transductive inference of some DGL techniques may result in low efficiency*. As illustrated in Fig. 1 (a) and (b), during the inference of CD on each single graph, these transductive methods must first optimize their model parameters from scratch via unsupervised objectives, which is usually time-consuming. Some related studies [20, 46, 54] only consider *transductive inference* regardless of its low efficiency.

Focusing on the CD with a given K and *task-independent* architecture in Fig. 1 (b), recent research [33] has validated that the *inductive inference*, which (i) trains a DGL model on historical graphs $\{\mathcal{G}_t\}$ and (ii) generalizes it to new graphs $\{\mathcal{G}'\}$ without additional optimization, can help achieve high inference efficiency on $\{\mathcal{G}'\}$. One can extend this method to K-agnostic CD by replacing the downstream clustering module (e.g., KMeans) with an advanced algorithm unrelated to K (e.g., DBSCAN [41]). Our experiments indicate that such a naive extension may still have low inference quality and efficiency compared with conventional baselines.

Contributions. We propose PRoCD (Pre-training & Refinement for Community Detection), a simple yet effective method as illustrated in Fig. 1 (c), to address the aforementioned limitations.

A novel model architecture. To derive feasible results for K-agnostic CD in an end-to-end way, we reformulate CD as the binary node pair classification. Unlike existing end-to-end models that directly assign the community label $c_i \in \{1, \cdots, K\}$ of each node v_i (e.g., via an MLP in Fig. 1 (a)), we develop a binary classifier, with a new design of *pair-wise temperature parameters*, to determine whether a pair of nodes (v_i, v_j) are partitioned into the same community. Given the classification result w.r.t. a set of node pairs $\mathcal{P} = \{(v_i, v_j)\}$ sampled from a graph $\mathcal{G}$, we construct another auxiliary graph $\tilde{\mathcal{G}}$, from which a feasible CD result of $\mathcal{G}$ can be extracted. Each connected component in $\tilde{\mathcal{G}}$ corresponds to a community in $\mathcal{G}$, with the number of components as the estimated K.

A novel learning paradigm. Inspired by recent advances in graph pre-training techniques, PRoCD follows a novel *pre-training & refinement* paradigm with three phases as in Fig. 1 (c). Based on the assumption that one has enough time to prepare a well-trained model in an offline way (i.e., *offline pre-training*), we first pre-train PRoCD on small synthetic graphs with various topology properties (e.g., degree distributions) and high-quality community ground-truth. We then generalize PRoCD (with frozen parameters) to large real graphs $\{\mathcal{G}'\}$ (i.e., *online generalization*) and derive corresponding CD results $\{\bar{C}'\}$ via only one feed-forward propagation (FFP) of the model (i.e., inductive inference of DGL). In some pre-training techniques [45, 50], *online generalization* only provides an initialization of model parameters, which is further fine-tuned w.r.t. different tasks. Analogous to this motivation, we treat $\bar{C}'$ as the initialization of an efficient CD method (e.g., *InfoMap* [40]) and adopt its output C' as the final CD result, which refines $\bar{C}'$ (i.e., *online refinement*). In particular, the *online generalization* and *refinement* may benefit from the powerful transfer ability of inductive inference regarding quality while ensuring high inference efficiency, since there is no time-consuming model optimization.

Note that our *pre-training & refinement* paradigm is different from existing graph pre-training techniques [45, 50]. We argue that they may suffer from the following issues about CD. Besides pre-training, these methods may require another optimization procedure for the fine-tuning or prompt-tuning of specific tasks on $\{\mathcal{G}'\}$, which is time-consuming and thus cannot help achieve high efficiency on $\{\mathcal{G}'\}$. To the best of our knowledge, related studies [4, 21, 44] merely focus on supervised tasks (e.g., node classification) and do not provide unsupervised tuning objectives for CD.

A better trade-off between quality and efficiency. Experiments on datasets with various scales demonstrate that PRoCD can ensure higher inference efficiency in CD without significant quality degradation, compared with running a refinement method from scratch. In some cases, PRoCD can even achieve improvement in both aspects. Therefore, we believe that this study provides a possible way to obtain a better trade-off between the two aspects.

2 RELATED WORK

In the past few decades, many CD methods have been proposed based on different problem statements, hypotheses, and techniques as reviewed in [52, 55]. Table 1 summarizes some representative CD approaches according to their original designs. Most of them focus on either high efficiency or quality.

Conventional efficient CD methods use heuristic strategies or fast approximation w.r.t. some relaxed CD objectives to derive feasible CD results, including the (i) greedy modularity maximization in *FastCom* [8] and *Louvain* [3], (ii) semi-definite relaxation of modularity in *Locale* [48], (iii) label propagation heuristic in *LPA* [36], as well as (iv) Monte Carlo approximation of stochastic block model (SBM) in *MC-SBM* [26] and *Par-SBM* [27].

Recent studies have also demonstrated the powerful potential of DGL to ensure high quality of CD, following the architectures in Fig. 1 (a) and (b). However, some methods (e.g., *DNR* [54], *DCRN* [20], and *DMoN* [46]) only considered the inefficient *transductive* inference, with time-consuming model optimization in the inference of CD on each single graph. Other approaches (e.g., *ClusNet* [49], *LGNN* [6], *GAP* [23], and *ICD* [33]) considered *inductive* inference across graphs. Results of *ICD* validated that the *inductive* inference can help achieve a better trade-off between quality and efficiency in *online generalization* (i.e., directly generalizing a model

Table 1: Summary of some representative CD methods.

Methods	*FastCom*	*GraClus*	*MC-SBM*	*Par-SBM*	*GMod*	*LPA*	*InfoMap*	*Louvain*	*Locale*	*ClusNet*	*LGNN*	*GAP*	*DMoN*	*DNR*	*DCRN*	*ICD*
References	[8]	[10]	[26]	[27]	[8]	[36]	[40]	[3]	[48]	[49]	[6]	[23]	[46]	[54]	[20]	[33]
Focus	High Efficiency									High Quality						Q&E
Techniques	Heuristic strategies or fast approximation w.r.t. relaxed CD objectives									*Task-dependent* DGL				*Task-independent* DGL		
Inference	Running the algorithm from scratch for each graph									*Transductive & Inductive*			*Transductive*			*Inductive*
K-agnostic	✘	✘	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘	✘	✘	✘	✘

pre-trained on historical graphs $\{\mathcal{G}_t\}$ to new graphs $\{\mathcal{G}'\}$). Nevertheless, the quality of *online generalization* may be affected by the possible distribution shift between $\{\mathcal{G}_t\}$ and $\{\mathcal{G}'\}$. In contrast, *online generalization* is used to construct the initialization of *online refinement* in our PRoCD method, which can ensure better quality on $\{\mathcal{G}'\}$. Moreover, most *task-dependent* DGL methods (e.g., *ClusNet*, *LGNN*, *GAP*, and *DMoN*) are inapplicable to K-agnostic CD, since they usually contain an output module related to K. Although *task-independent* approaches (e.g., *DNR*, *DCRN*, and *ICD*) do not contain such a module, their original designs still rely on a pre-set K (e.g., KMeans as the downstream clustering algorithm).

Different from the aforementioned methods, our PRoCD method can ensure higher efficiency in K-agnostic CD without significant quality degradation via a novel model architecture following the *pre-training & refinement* paradigm, as highlighted in Fig. 1 (c).

As reviewed in [45, 50], existing graph pre-training techniques usually follow the paradigms of (i) *pre-training & fine-tuning* (e.g., *GCC* [35], *L2P-GNN* [22], and *W2P-GNN* [4]) as well as (ii) *pre-training & prompting* (e.g., *GPPT* [43], *GraphPrompt* [21], and *ProG* [44]). In addition to *offline pre-training*, these methods rely on another optimization procedure for the fine-tuning or prompt-tuning of specific inference tasks, which is usually time-consuming. Moreover, most of them merely focus on supervised tasks (e.g., node classification, link prediction, and graph classification) and do not provide unsupervised tuning objectives for CD. Therefore, one cannot directly apply these pre-training methods to ensure the high efficiency in CD without quality degradation.

3 PROBLEM STATEMENTS & PRELIMINARIES

In this study, we consider the disjoint CD on undirected graphs. A graph can be represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V} = \{v_1, \cdots, v_N\}$ and $\mathcal{E} = \{(v_i, v_j) | v_i, v_j \in \mathcal{V}\}$ as the sets of nodes and edges. The topology of $\mathcal{G}$ can be described by an adjacency matrix $\mathbf{A} \in \{0, 1\}^{N \times N}$, where $\mathbf{A}_{ij} = \mathbf{A}_{ji} = 1$ if $(v_i, v_j) \in \mathcal{E}$ and $\mathbf{A}_{ij} = \mathbf{A}_{ji} = 0$ otherwise. Since CD is originally defined only based on graph topology [12], we assume that graph attributes are unavailable.

Community Detection (CD). Given a graph $\mathcal{G}$, CD aims to partition the node set $\mathcal{V}$ into K subsets (defined as communities) $C = \{C_1, \cdots, C_K\}$ ($C_r \cap C_s = \emptyset$ for $\forall r \neq s$) s.t. (i) the linkage within each community is dense but (ii) that between communities is relatively loose. We define that a CD task is K-agnostic if the number of communities K is unknown for a given graph $\mathcal{G}$, where one should simultaneously determine K and the corresponding CD result C. We consider K-agnostic CD in this study. Whereas, some existing methods [10, 46, 49] assume that K is given for each input $\mathcal{G}$ and thus cannot tackle such a challenging yet realistic setting.

Modularity Maximization. Mathematically, CD can be formulated as the modularity maximization objective [24], which maximizes the difference between the exact graph topology (described by $\mathbf{A}$) and a randomly generated case. Given $\mathcal{G}$ and K, it aims to derive a partition C that maximizes the following modularity metric:

$$\max_{C} \operatorname{Mod}(\mathcal{G}, K) := \frac{1}{2M} \sum_{r=1}^{K} \sum_{v_i, v_j \in C_r} [\mathbf{A}_{ij} - \frac{\deg(v_i)\deg(v_j)}{2M}], \quad (1)$$

where $\deg(v_i)$ is the degree of node v_i; M is the number of edges. Objective (1) can be rewritten into the following matrix form:

$$\min_{\mathbf{H}} -\operatorname{tr}(\mathbf{H}^T \mathbf{Q} \mathbf{H}) \text{ s.t. } \mathbf{H}_{ir} = \begin{cases} 1, & v_i \in C_r \\ 0, & \text{otherwise} \end{cases}, \quad (2)$$

where $\mathbf{Q} \in \mathbb{R}^{N \times N}$ is defined as the modularity matrix with $\mathbf{Q}_{ij} := [\mathbf{A}_{ij} - \deg(v_i)\deg(v_j)/(2M)]$; $\mathbf{H} \in \{0, 1\}^{N \times K}$ indicates the community membership of C. Our method does not directly solve the aforementioned problem. Instead, we try to extract informative community-preserving features from $\mathbf{Q}$.

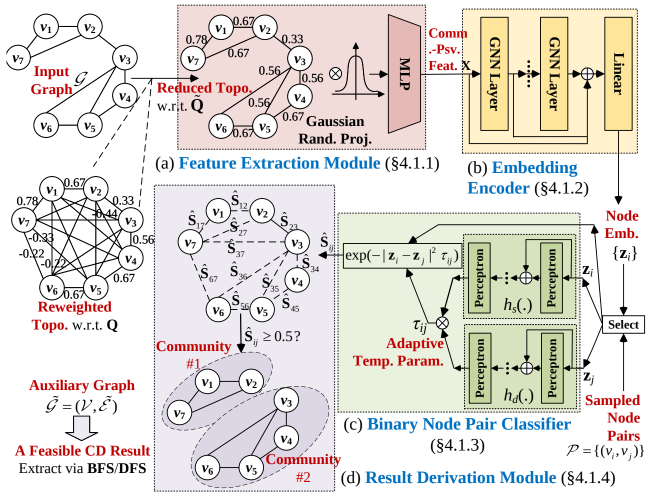
***Pre-training & Refinement* Paradigm**. To achieve a better trade-off between the quality and efficiency, we propose a *pre-training & refinement* paradigm based on the inductive inference of DGL. We represent a DGL model as $C = f(\mathcal{G}; \Theta)$, which derives the CD result C given a graph $\mathcal{G}$, with Θ as the set of trainable model parameters. As in Fig. 1 (c), the proposed paradigm includes (i) *offline pre-training*, (ii) *online generalization*, and (iii) *online refinement*.

In *offline pre-training*, we first generate a set of synthetic graphs $\mathcal{T} = \{\mathcal{G}_1, \cdots, \mathcal{G}_T\}$ via a generator (e.g., SBM [15]). The generation of each graph $\mathcal{G}_t \in \mathcal{T}$ includes the topology $(\mathcal{V}_t, \mathcal{E}_t)$ and community assignment ground-truth $C^{(t)}$. We then pre-train f on $\mathcal{T}$ (i.e., updating Θ) based on an objective regarding $\{(\mathcal{V}_t, \mathcal{E}_t)\}$ and $\{C^{(t)}\}$ in an *offline* way. After that, we generalize f to a large real graph $\mathcal{G}'$ with frozen Θ (i.e., *online generalization*), which can derive a feasible CD result $\tilde{C}'$ via only one FFP of f (i.e., inductive inference). Analogous to the fine-tuning in existing pre-training techniques, we treat $\tilde{C}'$ as the initialization of a conventional efficient CD method (e.g., *InfoMap* [40]) and adopt its output C as the final CD result, which further refines $\tilde{C}'$ (i.e., *online refinement*).

Evaluation Protocol. In real applications, it is usually assumed that *one has enough time to prepare a well-trained model in an offline way* (e.g., pre-training of LLMs). After deploying f, one may achieve high efficiency in the online inference on graphs $\{\mathcal{G}'\}$ (e.g., via one FFP of f without any model optimization). In contrast, conventional methods cannot benefit from pre-training but have to run their algorithms on $\{\mathcal{G}'\}$ from scratch, due to the inapplicability to inductive inference. Our evaluation focuses on the *online inference* of CD on $\{\mathcal{G}'\}$ (e.g., *online generalization* and *refinement* of PRoCD), which is analogous to the applications of foundation models. For instance, users benefit from the powerful online inference of LLMs (e.g., generating high-quality answers in few seconds) but do not need to train them using a great amount of resources.

4 METHODOLOGY

We propose PRoCD following a novel *pre-training & refinement* paradigm as highlighted in Fig. 1 (c). In this section, we elaborate



Input Graph $\mathcal{G}$
Reduced Topo. w.r.t. $\tilde{\mathbf{Q}}$
Gaussian Rand. Proj.
MLP
Comm.-Psv. Feat. X
(a) Feature Extraction Module (§4.1.1)
GNN Layer
GNN Layer
Linear
(b) Embedding Encoder (§4.1.2)
Node Emb. $\{\mathbf{z}_i\}$
Reweighted Topo. w.r.t. Q
Auxiliary Graph $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}})$
A Feasible CD Result Extract via BFS/DFS
Community #1
Community #2
$\hat{\mathbf{S}}_{ij} \geq 0.5?$
$\exp(-|\mathbf{z}_i - \mathbf{z}_j|^2 \tau_{ij})$
Adaptive Temp. Param.
Perceptron
$h_s(.)$
$h_d(.)$
Select
Sampled Node Pairs $\mathcal{P} = \{(v_i, v_j)\}$
(c) Binary Node Pair Classifier (§4.1.3)
(d) Result Derivation Module (§4.1.4)

Figure 2: Overview of the model architecture of PRoCD.

on the (i) model architecture, (ii) *offline pre-training*, as well as (iii) *online generalization and refinement* of PRoCD.

4.1 Model Architecture

To enable PRoCD to derive feasible results for K-agnostic CD in an end-to-end architecture, we reformulate CD as the binary node pair classification. For each graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we introduce an auxiliary variable $\mathbf{S} \in \{0,1\}^{N \times N}$ ($N := |\mathcal{V}|$), where $\mathbf{S}_{ij} = \mathbf{S}_{ji} = 1$ if nodes v_i and v_j are partitioned into the same community and $\mathbf{S}_{ij} = \mathbf{S}_{ji} = 0$ otherwise. Given a set of node pairs $\mathcal{P} = \{(v_i, v_j)\}$, we also rearrange corresponding elements in $\mathbf{S}$ as a vector $\mathbf{y} \in \{0,1\}^{|\mathcal{P}|}$, where $\mathbf{y}_l = \mathbf{S}_{ij} = \mathbf{S}_{ji}$ w.r.t. each node pair $p_l = (v_i, v_j) \in \mathcal{P}$. Fig. 2 provides an overview of the model architecture with a running example. Given a graph $\mathcal{G}$, the model samples a set of node pairs $\mathcal{P}$ and derives the estimated value of $\mathbf{y}_l = \mathbf{S}_{ij}$ w.r.t. each $p_l = (v_i, v_j) \in \mathcal{P}$, from which a feasible CD result C can be extracted.

4.1.1 ***Feature Extraction Module.*** As shown in Fig. 2 (a), we first extract community-preserving features, arranged as $\mathbf{X} \in \mathbb{R}^{N \times d}$ ($d \ll N$), for an input graph $\mathcal{G}$ from the modularity maximization objective (2). The modularity matrix $\mathbf{Q}$ in (2) is a primary component regarding graph topology. It can be considered as a reweighting of original topology, where nodes (v_i, v_j) with similar neighbor-induced features $(\mathbf{Q}_{i,:}, \mathbf{Q}_{j,:})$ are more likely to belong to a common community. To minimize the objective in (2), which is equivalent to maximizing the modularity metric in (1), $\mathbf{Q}_{ij}$ with a large value indicates that (v_i, v_j) are more likely to be partitioned into the same community (e.g., $\mathbf{Q}_{17} = \mathbf{Q}_{71} = 0.78$ in Fig. 2). Therefore, we believe that $\mathbf{Q}$ encodes key characteristics regarding community structures.

Note that $\mathbf{Q} \in \mathbb{R}^{N \times N}$ is usually dense. To utilize the sparsity of topology, we reduce $\mathbf{Q}$ to a sparse matrix $\tilde{\mathbf{Q}} \in \mathbb{R}^{N \times N}$, where $\tilde{\mathbf{Q}}_{ij} = \mathbf{Q}_{ij}$ if $(v_i, v_j) \in \mathcal{E}$ and $\tilde{\mathbf{Q}}_{ij} = 0$ otherwise. Although the reduction may lose some information, it enables the model to be scaled up to large sparse graphs without constructing an $N \times N$ dense matrix. Our experiments demonstrate that PRoCD can still derive high-quality CD results using $\tilde{\mathbf{Q}}$. Given $\tilde{\mathbf{Q}}$, we derive $\mathbf{X}$ via

$$\mathbf{X} = \mathrm{MLP}(\tilde{\mathbf{Q}}\Omega), \text{ with } \Omega \in \mathbb{R}^{N \times d} \sim \mathcal{N}(0, 1/d). \tag{3}$$

Concretely, the Gaussian random projection [1], an efficient dimension reduction technique that can preserve the relative distances between input features with rigorous theoretical guarantees, is first applied to $\tilde{\mathbf{Q}}$. We then incorporate non-linearity into the reduced features using an MLP.

4.1.2 ***Embedding Encoder.*** Given the extracted features $\mathbf{X}$, we then derive low-dimensional node embeddings $\{\mathbf{z}_i \in \mathbb{R}^d\}$ using a multi-layer GNN with skip connections, as shown in Fig. 2 (b). Here, we adopt GCN [16] as an example building block. One can easily extend the model to include other advanced GNNs. Let $\mathbf{Z}^{[s-1]}$ and $\mathbf{Z}^{[s]}$ be the input and output of the s-th GNN layer, with $\mathbf{Z}^{[0]} = \mathbf{X}$. The multi-layer GNN can be described as

$$\begin{aligned} &\mathbf{Z} = \mathrm{LN}(\mathrm{Linear}(\tilde{\mathbf{Z}})), \ \tilde{\mathbf{Z}} = \textstyle\sum_s \mathbf{Z}^{[s]}, \\ &\mathbf{Z}^{[s]} = \mathrm{LN}(\tanh(\hat{\mathbf{D}}^{-0.5}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-0.5}\mathbf{Z}^{[s-1]}\mathbf{W}^{[s]})), \end{aligned} \tag{4}$$

where $\hat{\mathbf{A}} := \mathbf{A} + \mathbf{I}_N$ is the adjacency matrix containing self-edges; $\hat{\mathbf{D}}$ is the degree diagonal matrix w.r.t. $\hat{\mathbf{A}}$; $\mathbf{W}^{[s]} \in \mathbb{R}^{d \times d}$ is a trainable weight matrix; $\mathrm{LN}(\mathbf{U})$ denotes the row-wise l_2 normalization on a matrix $\mathbf{U}$ (i.e., $\mathbf{U}_{i,:} \leftarrow \mathbf{U}_{i,:}/|\mathbf{U}_{i,:}|_2$); $\mathbf{Z} \in \mathbb{R}^{N \times d}$ is the matrix form of the derived embeddings, with $\mathbf{Z}_{i,:} = \mathbf{z}_i \in \mathbb{R}^d$ as the embedding of node v_i. In each GNN layer, $\mathbf{Z}^{[s]}_{i,:}$ is an intermediate representation of node v_i, which is the non-linear aggregations (i.e., weighted mean) of features w.r.t. $\{v_i\} \cup \mathrm{Nei}(v_i)$, with $\mathrm{Nei}(v_i)$ as the set of neighbors of v_i. Since this aggregation operation forces nodes (v_i, v_j) with similar neighbors $(\mathrm{Nei}(v_i), \mathrm{Nei}(v_j))$ (i.e., dense local topology) to have similar representations $(\mathbf{Z}^{[s]}_{i,:}, \mathbf{Z}^{[s]}_{j,:})$, the multi-layer GNN can enhance the ability of PRoCD to derive community-preserving embeddings. To obtain $\mathbf{Z}$, we first sum up the intermediate representations $\{\mathbf{Z}^{[s]}\}$ of all the layers and apply a linear mapping to the summed representation $\tilde{\mathbf{Z}}$. Furthermore, the row-wise l_2 normalization is applied. It forces $\{\mathbf{z}_i\}$ to be distributed in a unit sphere, with $|\mathbf{z}_i - \mathbf{z}_j|^2 = 2 - 2\mathbf{z}_i\mathbf{z}_j^T$.

4.1.3 ***Binary Node Pair Classifier.*** As highlighted in Fig. 2 (c), given a pair of nodes (v_i, v_j) sampled from a graph $\mathcal{G}$, we use a binary classifier to estimate $\mathbf{S}_{ij}$ based on embeddings $(\mathbf{z}_i, \mathbf{z}_j)$ and determine whether (v_i, v_j) are partitioned into the same community. A widely adopted design of binary classifier is as follow

$$\hat{\mathbf{S}}_{ij} = \mathrm{sigmoid}(\mathbf{z}_i\mathbf{z}_j^T/\tau), \tag{5}$$

with τ as a pre-set temperature parameter. Instead of directly using (5), we introduce a novel binary classifier with *pair-wise adaptive temperature parameters* $\{\tau_{ij}\}$, which can derive a more accurate estimation of $\mathbf{S}$ to derive high-quality CD results as demonstrated in our experiments. Our binary classifier can be described as

$$\begin{aligned} &\hat{\mathbf{S}}_{ij} = \exp(-|\mathbf{z}_i - \mathbf{z}_j|^2\tau_{ij}) = \exp(2\tau_{ij}(\mathbf{z}_i\mathbf{z}_j^T - 1)), \\ &\text{with } \tau_{ij} = h_s(\mathbf{z}_i)h_d(\mathbf{z}_j)^T, \end{aligned} \tag{6}$$

where h_s and h_d are MLPs with the same configuration. In (6), $\mathbf{S}_{ij}$ is estimated based on the distance between $(\mathbf{z}_i, \mathbf{z}_j)$ and a pair-wise temperature parameter τ_{ij}. Different node pairs $\{(v_i, v_j)\}$ may have different parameters $\{\tau_{ij}\}$ determined by embeddings $\{(\mathbf{z}_i, \mathbf{z}_j)\}$.

4.1.4 ***Result Derivation Module.*** As illustrated in Fig. 2 (d), we develop a result derivation module, which outputs a feasible CD

Algorithm 1: Deriving a Feasible CD Result

```
Input: input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; number of sampled node pairs $n_S$
Output: a feasible CD result $C$ w.r.t. $\mathcal{G}$
1  $\mathcal{P} \leftarrow \mathcal{E}$//Initialize the set of node pairs $\mathcal{P}$ using edge set $\mathcal{E}$
2  for sample_count from 1 to $n_S$ do
3      randomly sample a node pair $p = (v_i, v_j)$
4      add the sampled $p = (v_i, v_j)$ to $\mathcal{P}$
5  derive $\hat{\mathbf{y}}$ w.r.t. $\mathcal{P}$ via one FFP of the model
6  $\tilde{\mathcal{E}} \leftarrow \emptyset$//Initialize edge set of auxiliary graph $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}})$
7  for each node pair $p_l = (v_i, v_j) \in \mathcal{P}$ do
8      if $\hat{\mathbf{y}}_l > 0.5$ then
9          add $p_l = (v_i, v_j)$ to $\tilde{\mathcal{E}}$
10 extract connected components of $\tilde{\mathcal{G}}$ via DFS/BFS on $\tilde{\mathcal{E}}$
11 treat each extracted component as a community to form $C$
```

result C based on a set of node pairs sampled from the input graph $\mathcal{G}$. Algorithm 1 summarizes the procedure of this module.

In lines 1-4, we construct a set of node pairs $\mathcal{P} = \{(v_i, v_j)\}$ (e.g., dotted lines in Fig. 2 (d)), which includes (i) all the edges of input graph $\mathcal{G}$ (e.g., (v_1, v_7) and (v_3, v_4) in Fig. 2 (d)) and (ii) n_S randomly sampled node pairs (e.g., (v_6, v_7) and (v_3, v_7) in Fig. 2 (d)).

In line 5, we derive the estimated values $\{\hat{\mathbf{S}}_{ij}\}$ w.r.t. node pairs in $\mathcal{P}$ (via one FFP of the model) and rearrange them as a vector $\hat{\mathbf{y}} \in \mathbb{R}^{|\mathcal{P}|}$. In particular, $\hat{\mathbf{y}}_l = \hat{\mathbf{S}}_{ij}$ represents the probability that a pair of nodes $p_l = (v_i, v_j)$ are partitioned into a common community.

In lines 6-9, we further construct an auxiliary graph $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}})$ based on $\hat{\mathbf{y}}$ and $\mathcal{P}$. $\tilde{\mathcal{G}}$ has the same node set $\mathcal{V}$ as the input graph $\mathcal{G}$ but a different edge set $\tilde{\mathcal{E}}$. For each $p_l = (v_i, v_j) \in \mathcal{P}$, we add (v_i, v_j) to $\tilde{\mathcal{E}}$ (i.e., preserving this node pair as an edge in $\tilde{\mathcal{G}}$) if $\hat{\mathbf{y}}_l > 0.5$. $\tilde{\mathcal{G}}$ may contain multiple connected components (e.g., the two components in Fig. 2 (d)). All the edges $\{e_l = (v_i, v_j)\}$ within a component are with high values of $\{\hat{\mathbf{y}}_l = \hat{\mathbf{S}}_{ij}\}$, indicating that the associated nodes are more likely to belong to the same community.

In lines 10-11, we derive a feasible CD result C w.r.t. $\mathcal{G}$ by extracting all the connected components of $\tilde{\mathcal{G}}$ via the depth-first search (DFS) or breadth-first search (BFS). Each connected component of $\tilde{\mathcal{G}}$ corresponds to a unique community in $\mathcal{G}$ (e.g., the two communities in Fig. 2 (d)), with the number of components as the estimated number of communities K. Therefore, the aforementioned designs enable our PRoCD method to tackle K-agnostic CD.

4.2 Offline Pre-Training

We conducted the *offline pre-training* of PRoCD on a set of synthetic graphs $\mathcal{T} = \{\mathcal{G}_1, \cdots, \mathcal{G}_T\}$. One significant advantage of using synthetic pre-training data is that we can simulate various properties of graph topology with different levels of noise and high-quality ground-truth by adjusting parameters of the synthetic generator.

In particular, we consider a challenging setting of pre-training PRoCD on small synthetic graphs (e.g., with $N \approx 5 \times 10^3$ nodes) but generalize it to large real graphs (e.g., $N > 10^6$). One reason of using small pre-training graphs is that it enables PRoCD to construct dense $N \times N$ matrices in pre-training objectives (e.g., $\hat{\mathbf{S}} \in \mathbb{R}^{N \times N}$ for binary classification) and thus fully explore the topology and community ground-truth of pre-training data. In contrast, constructing dense $N \times N$ matrices for large graphs may be intractable. Although there may be distribution shifts between the (i) small pre-training graphs $\{\mathcal{G}_t\}$ and (ii) large graphs $\{\mathcal{G}'\}$ to be partitioned, our experiments demonstrate that *offline pre-training* on $\{\mathcal{G}_t\}$ is essential for PRoCD to derive feasible CD results $\{\bar{C}'\}$ for $\{\mathcal{G}'\}$. By using $\{\bar{C}'\}$ as the initialization, their quality can be further refined via an efficient CD method.

4.2.1 ***Generation of Pre-Training Data.*** As a demonstration, we adopt the degree-corrected SBM (DC-SBM) [15] implemented by `graph-tool`[1] to generate synthetic pre-training graphs. Besides topology, the generator also produces community assignments w.r.t. a specified level of noise, which can be used as the high-quality ground-truth of CD. Our experiments also validate that the synthetic ground-truth can help PRoCD derive a good initialization for *online refinement*. Whereas, community ground-truth is usually unavailable for existing methods [46, 49] (pre-)trained on real graphs. Instead of using fixed generator parameters, we let them follow certain probability distributions to simulate various topology properties (e.g., distributions of node degrees and community sizes) and noise levels of community structures. Namely, different graphs may be generated based on different parameter settings sampled from certain distributions. Due to space limits, we leave details about the parameter setting and generation algorithm in Appendix A.

4.2.2 ***Pre-Training Objectives & Algorithms.*** For each synthetic graph $\mathcal{G}_t$, suppose there are N_t nodes partitioned into K_t communities. We use $\mathbf{M}^{(t)}$ to denote a vector/matrix associated with $\mathcal{G}_t$ (e.g., $\hat{\mathbf{y}}^{(t)}$ and $\hat{\mathbf{S}}^{(\mathbf{t})}$). Given a graph $\mathcal{G}_t$, some previous studies [46, 49] adopt a relaxed version of the modularity maximization objective (2) for model optimization and demonstrate its potential to capture community structures. This relaxed objective allows $\mathbf{H}^{(t)} \in \{0,1\}^{N_t \times K_t}$ in (2), which describes hard community assignments, to be continuous values (e.g., $\mathbf{H}^{(t)} \in \mathbb{R}_+^{N_t \times K_t}$ derived via an MLP). Note that we reformulate CD as the binary node pair classification with $\mathbf{S}^{(t)} \in \{0,1\}^{N_t \times N_t}$. $\hat{\mathbf{S}}^{(t)} \in \mathbb{R}_+^{N_t \times N_t}$ derived from the binary classifier can be considered as the relaxation of $\mathbf{S}^{(t)}$. Based on $\hat{\mathbf{S}}^{(t)}$, we rewrite the modularity maximization objective (1) to the following relaxed form for each graph $\mathcal{G}_t$:

$$\mathcal{L}_{\text{MOD}}(\mathcal{G}_t) := -[\frac{1}{M} \sum_{e_l \in \mathcal{E}_t} \hat{\mathbf{y}}_l^{(t)} - \frac{\lambda}{4M^2} \sum_{ij} (\mathbf{D}^{(t)} \hat{\mathbf{S}}^{(t)} \mathbf{D}^{(t)})_{ij}], \quad (7)$$

where $\hat{\mathbf{y}}^{(t)}$ is the rearrangement of elements in $\hat{\mathbf{S}}^{(t)}$ w.r.t. the edge set $\mathcal{E}_t$; $\mathbf{D}^{(t)}$ is the degree diagonal matrix of $\mathcal{G}_t$; $\lambda > 0$ is the resolution parameter of modularity maximization [37], with $\lambda < 1$ (> 1) favoring the partition of large (small) communities.

In addition, we utilize the synthetic community ground-truth C_t of each graph $\mathcal{G}_t$ to enhance the ability of PRoCD to capture community structures. Note that different graphs $\{\mathcal{G}_t\}$ may have different numbers of communities $\{K_t\}$. However, some DGL-based CD methods [6, 46, 49] are only designed for graphs with a fixed K. Community labels are also permutation-invariant. For instance, $(c_1, c_2, c_3, c_4, c_5) = (1, 1, 1, 2, 2)$ and $(c_1, c_2, c_3, c_4, c_5) = (2, 2, 2, 1, 1)$ represent the same community assignment, with c_i as the label assignment of node v_i. Hence, the standard multi-class cross entropy objective cannot be directly applied to integrate community ground-truth. PRoCD handles these issues by reformulating CD as

[1]https://graph-tool.skewed.de/

Algorithm 2: *Offline Pre-training*

```
Input: synthetic pre-training graphs $\mathcal{T} = \{\mathcal{G}_1, \cdots, \mathcal{G}_T\}$ and their
       ground-truth; hyper-parameters $\{\lambda, \alpha\}$; number of epochs $n_P$;
       learning rate $\eta$
Output: optimized model parameters $\Theta^*$
1 initialize model parameters $\Theta$
2 for epoch from 1 to $n_P$ do
3     for each $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t) \in \mathcal{T}$ do
4         construct $\mathbf{S}^{(t)}$ w.r.t. ground-truth $C^{(t)}$
5         get input feature $\mathbf{X}^{(t)}$ w.r.t. $\mathcal{G}_t$ via (3)
6         get estimated values $\hat{\mathbf{S}}^{(t)}$ w.r.t. $\mathcal{G}_t$ via one FFP
7         arrange elements w.r.t. $\mathcal{E}_t$ in $\hat{\mathbf{S}}^{(t)}$ as $\hat{\mathbf{y}}^{(t)}$
8         update model parameters $\Theta \leftarrow \mathrm{Opt}(\eta, \Theta, \partial\mathcal{L}_{\mathrm{PTN}}(\mathcal{G}_t)/\partial\Theta)$
9 save the optimized model parameters $\Theta^*$
```

the binary node pair classification with auxiliary variables $\{\mathbf{S}^{(t)}\}$, where the dimensionality and values of $\{\mathbf{S}^{(t)}\}$ are unrelated to $\{K_t\}$ and permutations of community labels. Given a graph $\mathcal{G}_t$, one can construct $\mathbf{S}^{(t)}$ based on ground-truth $C^{(t)}$ and derive $\hat{\mathbf{S}}^{(t)}$ via one FFP of the model. We introduce the following binary cross entropy objective that covers all the $N_t \times N_t$ node pairs:

$$\mathcal{L}_{\mathrm{BCE}}(\mathcal{G}_t) := -\sum_{v_i, v_j \in \mathcal{V}} \left[\begin{array}{l} \mathbf{S}_{ij}^{(t)} \log \hat{\mathbf{S}}_{ij}^{(t)} + \\ (1 - \mathbf{S}_{ij}^{(t)}) \log(1 - \hat{\mathbf{S}}_{ij}^{(t)}) \end{array} \right]. \tag{8}$$

Finally, we formulate the pre-training objective w.r.t. $\mathcal{G}_t$ as

$$\mathcal{L}_{\mathrm{PTN}}(\mathcal{G}_t) := \mathcal{L}_{\mathrm{MOD}}(\mathcal{G}_t) + \alpha \mathcal{L}_{\mathrm{BCE}}(\mathcal{G}_t), \tag{9}$$

where $\alpha > 0$ is the hyper-parameter to balance $\mathcal{L}_{\mathrm{MOD}}$ and $\mathcal{L}_{\mathrm{BCE}}$. The *offline pre-training* procedure is concluded in Algorithm 2, where the Adam optimizer with learning rate η is applied to update model parameters Θ; the updated Θ after n_P epochs are then saved for *online generalization and refinement*. Since only the embedding encoder and binary classifier contain model parameters to be learned, we do not need to apply Algorithm 1 of the result derivation module to derive a feasible CD result in *offline pre-training*.

4.3 Online Generalization & Refinement

After *offline pre-training*, we can directly generalize PRoCD to large real graphs $\{\mathcal{G}'\}$ (with frozen model parameters Θ^*) and derive feasible CD results $\{\bar{C}'\}$ via only one FFP of the model and Algorithm 1 (i.e., *online generalization*).

Recent advances in graph pre-training techniques [50] demonstrate that pre-training may provide good initialized model parameters for downstream tasks, which can be further improved via a task-specific fine-tuning procedure. To the best of our knowledge, most existing graph pre-training methods merely consider supervised tasks (e.g., node classification) but do not provide unsupervised tuning objectives for CD. A straightforward fine-tuning strategy for CD is applying the modularity maximization objective (7) to large real graphs $\{\mathcal{G}'\}$ (e.g., gradient descent w.r.t. a relaxed version of (2) with large dense matrices $\{\mathbf{Q}'\}$), which is usually intractable. Analogous to fine-tuning, we introduce the *online refinement* phase with a different design. Instead of fine-tuning, we treat the CD result $\bar{C}'$ derived in *online generalization* as the initialization of a conventional efficient CD method (e.g., *LPA* [36], *InfoMap* [40], and *Locale* [48] in our experiments) and run this method to derive a refined CD result C'.

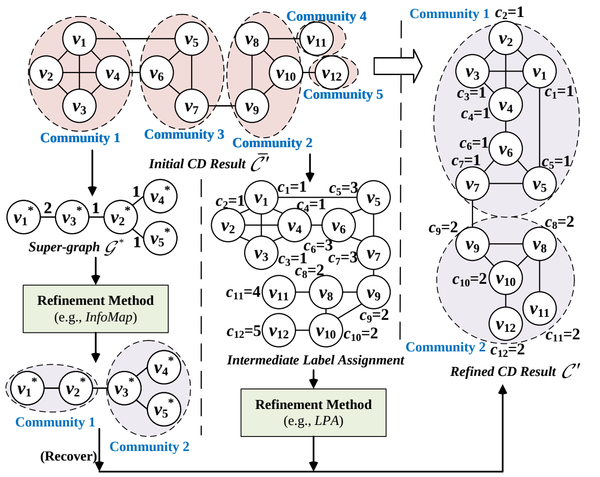

Community 1
Community 3
Community 2
Community 4
Community 5
Initial CD Result $\bar{C}'$
Super-graph $\mathcal{G}^*$
Refinement Method (e.g., *InfoMap*)
Community 1
Community 2
(Recover)
Intermediate Label Assignment
Refinement Method (e.g., *LPA*)
Community 1
Community 2
Refined CD Result C'

Figure 3: Illustration of the two strategies to construct the initialization of *online refinement*.

Algorithm 3: *Online Generalization & Refinement*

```
Input: large real graph $\mathcal{G}'$ to be partitioned; saved model parameters $\Theta^*$
Output: CD results $C'$ w.r.t. $\mathcal{G}'$
1 get input feature $\mathbf{X}'$ w.r.t. $\mathcal{G}'$ via (3)
2 get a feasible CD result $\bar{C}'$ w.r.t. $\mathcal{G}'$ via Algorithm 1
3 construct the initialization for refinement based on $\bar{C}'$
4 get refined CD result $C'$ via an efficient refinement method
```

Table 2: Statistics of datasets.

Datasets	N	E	Min	Max	Avg Deg	Density
Protein [42]	81,574	1,779,885	1	4,983	43.6	5e-4
ArXiv [47]	169,343	1,157,799	1	13,161	13.7	8e-5
DBLP [53]	317,080	1,049,866	1	343	6.6	2e-5
Amazon [53]	334,863	925,872	1	549	5.5	2e-5
Youtube [53]	1,134,890	2,987,624	1	28,754	5.3	5e-6
RoadCA [18]	1,957,027	2,760,388	1	12	2.82	1e-6

As shown in Fig. 3, we adopt two strategies to construct the initialization. First, a super-graph $\mathcal{G}^*$ can be extracted based on $\bar{C}'$, where we merge nodes in each community $\bar{C}'_r \in \bar{C}'$ as a super-node (e.g., v_3^* w.r.t. $\bar{C}'_3 = \{v_5, v_6, v_7\}$ in Fig. 3) and set the number of edges between communities as the weight of each super-edge (e.g., 2 for (v_1^*, v_2^*) in Fig. 3). We then use $\mathcal{G}^*$ as the input of an efficient method that can handle weighted graphs (e.g., *InfoMap* and *Locale*) to derive a CD result w.r.t. $\mathcal{G}^*$, which is further recovered to the result C' w.r.t. $\mathcal{G}'$. Second, we also use $\bar{C}'$ as the intermediate label assignment of a method that iteratively updates community labels (e.g., *LPA*) and refines the label assignment until convergence (e.g., refining c_6 from 3 to 1 in Fig. 3). Compared with running the refinement method on $\mathcal{G}'$ from scratch, *online refinement* may be more efficient, because the constructed initialization reduces the number of nodes to be partitioned and iterations to update community labels. It is also expected that PRoCD can transfer the capability of capturing community structures from the pre-training data to $\{\mathcal{G}'\}$.

Algorithm 3 concludes the aforementioned procedure. The complexity of *online generalization* (i.e., lines 1-2) is $O((N+M)d^2)$, where N and M are the numbers of nodes and edges; d is the embedding dimensionality. The complexity of *online refinement* (i.e., lines 3-4) depends on the refinement method, which is usually efficient. We leave detailed complexity analysis in Appendix B.

Table 3: Evaluation results w.r.t. efficiency metric of inference time↓ (sec) and quality metric of modularity↑.

	Protein			ArXiv			DBLP			Amazon			Youtube			RoadCA		
	K	**Time↓**	**Mod↑**	*K*	**Time↓**	**Mod↑**	*K*	**Time↓**	**Mod↑**	*K*	**Time↓**	**Mod↑**	*K*	**Time↓**	**Mod↑**	*K*	**Time↓**	**Mod↑**
MC-SBM	170	1143.93	0.1463	303	1014.63	0.5205	430	646.91	0.7620	368	799.05	0.8860	186	6608.67	0.2027	324	8492.20	0.9491
Par-SBM	1494	39.18	0.4721	4936	40.26	0.0529	10854	28.04	0.2797	10813	39.40	0.3094	27948	236.61	0.0901	39987	149.10	0.5970
GMod	69	5840.49	0.6319	1317	6693.91	0.5797	OOT	OOT	OOT	1669	12729.36	0.8703	OOT	OOT	OOT	OOT	OOT	OOT
Louvain	60	46.23	0.6366	155	49.07	0.7059	209	83.50	0.8209	240	48.10	0.9262	6105	173.70	0.7218	376	408.25	0.9923
RaftGP-C	218	69.82	0.6124	213	79.11	0.6147	39	81.99	0.6594	53	87.66	0.7530	1	254.77	0.0000	256	470.05	0.8072
RaftGP-M	185	69.44	0.6199	194	80.10	0.6107	39	83.48	0.6644	54	88.24	0.7561	131	272.82	0.6659	256	469.14	0.8043
LouNE+DBSCAN	32	77.20	0.6362	170	214.55	0.7058	737	109.87	0.8065	252	64.45	0.9255	17908	671.52	0.6228	19081	236.46	0.9531
SktNE+DBSCAN	371	16.18	0.1808	388	542.92	0.3533	15341	45.60	0.5716	13283	49.85	0.7047	17799	485.72	0.1076	59726	1410.91	0.4073
ICD-C+DBSCAN	1277	89.30	0.4961	256	296.02	0.0011	3694	1795.04	0.1047	1865	2027.30	0.0278	OOT	OOT	OOT	OOT	OOT	OOT
ICD-M+DBSCAN	1091	88.47	0.5651	264	294.13	0.0019	3950	984.98	0.1156	2478	872.11	0.0430	OOT	OOT	OOT	OOT	OOT	OOT
LPA	1919	132.32	0.5925	11467	113.30	0.3752	46630	146.22	0.6196	41693	76.49	0.7090	111887	370.15	0.5520	606249	245.54	0.5524
PRoCDw/ *LPA*	1499	**32.07**	**0.6046**	10765	**19.93**	**0.6188**	38684	**20.10**	**0.6336**	34270	**20.40**	**0.7240**	93566	**68.72**	**0.6150**	391735	**67.74**	**0.5765**
Improve.		+75.8%	+2.1%		+82.4%	+64.9%		+86.3%	+2.3%		+73.3%	+2.1%		+81.4%	+11.4%		+72.4%	+4.4%
GC+*LPA*	1985	32.52	0.5790	12854	21.57	0.5290	44494	30.04	0.5906	41513	29.40	0.5723	124044	69.17	0.2710	442623	106.89	0.4838
InfoMap	11	26.38	0.1972	69	19.69	0.6332	527	29.11	0.8195	13	28.72	0.8037	956	114.63	0.6959	3	259.44	0.6455
PRoCDw/ *IM*	18	**15.60**	**0.1996**	42	**17.08**	**0.6349**	298	**17.23**	**0.8222**	417	**16.66**	**0.9222**	438	**83.01**	0.6811	145	**79.48**	**0.9889**
Improve.		+40.9%	+1.2%		+13.3%	+0.3%		+40.8%	+0.3%		+42.0%	+14.7%		+27.6%	-2.1%		+69.4%	+53.2%
GC+*IM*	9	33.84	0.1318	48	24.28	0.4735	305	27.99	0.8086	4	28.01	0.4189	687	104.85	0.6919	3	129.74	0.6561
Locale	51	43.04	0.6409	150	46.66	0.7159	298	40.08	0.8375	392	30.85	0.9343	3441	216.12	0.7353	410	172.48	0.9935
PRoCDw/ *Lcl*	35	**16.93**	0.6362	64	**31.76**	0.7133	147	**24.15**	**0.8376**	173	**18.94**	0.9319	1647	**161.24**	0.7315	330	**90.88**	0.9930
Improve.		+60.7%	-0.7%		+31.9%	-0.4%		+39.8%	+0.01%		+38.6%	-0.3%		+25.4%	-0.5%		+47.3%	-0.1%
GC+*Lcl*	33	44.35	0.6343	150	45.17	0.6988	196	36.87	0.8254	344	30.09	0.9220	3228	194.49	0.7315	327	130.57	0.9928

Table 4: Detailed inference time (sec) of PRoCD.

Datasets	Methods	Feat	FFP	Init	Rfn
Protein	**PRoCD** w/ *LPA*	2.84	4.42	6.50	18.31
	PRoCD w/ *InfoMap*				1.84
	PRoCD w/ *Locale*				3.17
ArXiv	**PRoCD** w/ *LPA*	1.49	0.99	4.50	12.96
	PRoCD w/ *InfoMap*				10.11
	PRoCD w/ *Locale*				24.79
DBLP	**PRoCD** w/ *LPA*	1.97	1.28	4.19	12.66
	PRoCD w/ *InfoMap*				9.79
	PRoCD w/ *Locale*				16.71
Amazon	**PRoCD** w/ *LPA*	2.08	1.99	3.81	12.51
	PRoCD w/ *InfoMap*				8.78
	PRoCD w/ *Locale*				11.06
Youtube	**PRoCD** w/ *LPA*	5.81	2.75	15.13	45.02
	PRoCD w/ *InfoMap*				59.32
	PRoCD w/ *Locale*				137.54
RoadCA	**PRoCD** w/ *LPA*	8.10	2.34	13.14	44.16
	PRoCD w/ *InfoMap*				55.90
	PRoCD w/ *Locale*				67.31

5 EXPERIMENTS

5.1 Experiment Setup

Datasets. We evaluated the inference efficiency and quality of our PRoCD method on 6 public datasets with statistics depicted in Table 2, where N and M are the numbers of nodes and edges. Note that these datasets do not provide ground-truth about the community assignment and the number of communities K. Some more details regarding the datasets can be found in Appendix C.1.

Baselines. We compared **PRoCD** over 11 baselines.

(i) *MC-SBM* [26], (ii) *Par-SBM* [27], (iii) *GMod* [8], (iv) *LPA* [36], (v) *InfoMap* [40], (vi) *Louvain* [3], and (vii) *Locale* [48], and (viii) *RaftGP* [13] are efficient end-to-end methods that can tackle K-agnostic CD. Note that some other approaches whose designs rely on a pre-set K (e.g., *FastCom* [8], *GraClus* [10], *ClusterNet* [49], *LGNN* [6], and *DMoN* [46]) could not be included in our experiments.

In addition, we extended (ix) *LouvainNE* [2], (x) *SketchNE* [51], and (xi) *ICD* [33], which are state-of-the-art efficient graph embedding methods, to K-agnostic CD by combining the derived embeddings with DBSCAN [41], an efficient clustering algorithm that does not require K. In particular, *LouvainNE* and *ICD* are claimed to be community-preserving methods. *RaftGP-C*, *RaftGP-M*, *ICD-C*, and *ICD-M* denote different variants of *RaftGP* and *ICD*.

As a demonstration, we adopted *LPA*, *InfoMap*, and *Locale* (an advanced extension of *Louvain*) for the *online refinement* of **PRoCD**, forming three variants of our method. In Fig. 3, the first strategy of constructing the refinement initialization has a motivation similar to graph coarsening (GC) techniques [5], which merge a large graph $\mathcal{G}'$ into a super-graph $\mathcal{G}^*$ (i.e., reducing the number of nodes to be partitioned) via a heuristic strategy. Whereas, **PRoCD** constructs $\mathcal{G}^*$ based on the output of a pre-trained model. To highlight the superiority of *offline pre-training* and inductive inference beyond GC, we introduced another baseline that provides the initialization (with the same number of super-nodes as **PRoCD**) for *online refinement* using the efficient GC strategy proposed in [19].

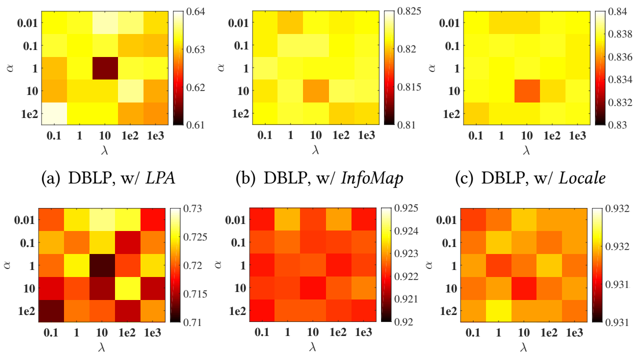
ICD is an inductive baseline with a paradigm similar to **PRoCD**, including *offline (pre-)training* on historical graphs and *online generalization* to new graphs. However, there is no *online refinement* in *ICD*. We conducted the *offline pre-training* of *ICD* and **PRoCD** on the generated synthetic graphs as described in Section 4.2.1 and generalized them to each dataset for online inference. For the rest baselines, we had to run them from scratch on each dataset.

Evaluation Criteria. Following the evaluation protocol described in Section 3, we adopted modularity and inference time (sec) as the quality and efficiency metrics. Moreover, we define that a method encounters the out-of-time (OOT) exception if it fails to derive a feasible CD result within 2×10^4 seconds.

Due to space limits, we elaborate on other details of experiment setup (e.g., parameter settings) in Appendix C.

5.2 Quantitative Evaluation & Discussions

The average evaluation results over five independent runs are reported in Table 3. Besides the total inference time, we also report the time w.r.t. each inference phase of PRoCD in Table 4, where 'Feat', 'FFP', 'Init', and 'Rfn' denote the time of (i) feature extraction, (ii) one FFP, (iii) initial result derivation, and (iv) online refinement. Further analysis about the inference time of GC and embedding baselines is given in Tables 9 and 10 (see Appendix).



(a) DBLP, w/ *LPA* (b) DBLP, w/ *InfoMap* (c) DBLP, w/ *Locale*
(d) Amazon, w/ *LPA* (e) Amazon, w/ *InfoMap* (f) Amazon, w/ *Locale*

Figure 4: Parameter analysis w.r.t. α and λ on DBLP and Amazon in terms of modularity↑.

In Table 3, three variants of PRoCD achieve significant improvement of efficiency w.r.t. the refinement methods while the quality degradation is less than 2%. In some cases, PRoCD can even obtain improvement for both aspects. Compared with all the baselines, PRoCD can ensure the best efficiency on all the datasets and is in groups with the best quality. In summary, we believe that the proposed method provides a possible way to obtain a better trade-off between the quality and efficiency in CD.

In most cases, PRoCD outperforms GC baselines in both aspects, which validates the superiority of offline pre-training and inductive inference beyond heuristic GC. The extensions of some embedding baselines (e.g., *SketchNE* and *ICD* combined with DBSCAN) suffer from low quality and efficiency compared with other end-to-end approaches (e.g., *Louvain* and *Locale*). It implies that the task-specific module (e.g., binary node pair classifier in PRoCD) is essential for the embedding framework to ensure the quality and efficiency in K-agnostic CD, which is seldom considered in related studies.

According to Table 4, online refinement is the major bottleneck in the inference of PRoCD, with the runtime depending on concrete refinement methods. Compared with running a refinement method from scratch, online refinement has much lower runtime. It validates our motivation that constructing the initialization (as described in Fig. 3) can reduce the number of nodes to be partitioned and iterations to update label assignments for refinement methods.

To construct the initialization of online refinement, GC is not as efficient as the online generalization of PRoCD (i.e., feature extraction, one FFP, and initial result derivation) according to Table 9. In Table 10, the downstream clustering (i.e., DBSCAN to derive a feasible result for K-agnostic CD) is time-consuming, which results in low efficiency of the extended embedding baselines, although their embedding derivation phases are efficient.

5.3 Ablation Study

For our PRoCD method, we verified the effectiveness of (i) online refinement, (ii) offline pre-training, (iii) feature extraction module in (3), (iv) embedding encoder in (4), (v) binary node pair classifier in (6), (vi) cross-entropy objective $\mathcal{L}_{\rm BCE}$, and (vii) modularity maximization objective $\mathcal{L}_{\rm MOD}$ by removing corresponding components from the original model. In case (iii), we let $\mathbf{X}$ be the one-hot encodings of node degree, which is a standard feature extraction strategy for GNNs when attributes are unavailable. We directly used $\mathbf{X}$ as the derived embeddings in case (iv), while we adopted the naive binary classifier (5) to replace our design (6) in case (v). The average ablation study results on DBLP and Amazon over five independent runs are reported in Table 5, where there are quality declines in all the cases. In some extreme cases (e.g., the one without pre-training), the model mistakenly partitions all the nodes into one community, resulting in a modularity value of 0.0000. In summary, all the considered procedures and components are essential to ensure the quality of PRoCD. Further analysis about the number of pre-training graphs T is given in Appendix D.

Table 5: Ablation studies on DBLP and Amazon in terms of modularity↑ and inference time↓ (sec).

	DBLP			Amazon		
	K	Mod↑	Time↓	K	Mod↑	Time↓
(i)w/o Rfn	106553	0.4394	7.44	112408	0.5620	7.88
PRoCD w/ *LPA*	38684	0.6336	20.10	34270	0.7240	20.40
(ii)w/o Ptn	1	0.0000	9.75	1	0.0000	8.81
(iii)w/o Feat (3)	537	0.0625	18.01	1	0.0000	6.29
(iv)w/o EmbEnc (4)	58401	0.5406	18.26	53127	0.5090	17.08
(v)w/o BinClf (6)	48	0.0009	6.43	23	0.0008	6.31
(vi)w/o $\mathcal{L}_{\rm BCE}$ (8)	22496	0.5657	17.84	16378	0.6580	20.19
(vii) w/o $\mathcal{L}_{\rm MOD}$ (7)	39792	0.6261	20.12	34853	0.7188	19.31
PRoCD w/ *IM*	298	0.8222	17.23	417	0.9222	16.66
(ii)w/o Ptn	1	0.0000	9.75	1	0.0000	8.81
(iii)w/o Feat (3)	2	0.0001	5.67	1	0.0000	5.13
(iv)w/o EmbEnc (4)	516	0.8182	26.55	6	0.3869	39.27
(v)w/o BinClf (6)	48	0.0009	5.88	23	0.0008	5.68
(vi)w/o $\mathcal{L}_{\rm BCE}$ (8)	156	0.2895	12.93	356	0.4872	9.91
(vii)w/o $\mathcal{L}_{\rm MOD}$ (7)	306	0.8200	19.22	2	0.2638	22.06
PRoCD w/ *Lcl*	147	0.8376	24.15	173	0.9319	18.94
(ii)w/o Ptn	1	0.0000	9.75	1	0.0000	8.81
(iii)w/o Feat (3)	551	0.0647	5.66	1	0.0000	5.15
(iv)w/o EmbEnc (4)	216	0.8368	42.20	307	0.9228	35.97
(v)w/o BinClf (6)	48	0.0009	6.66	23	0.0008	6.43
(vi)w/o $\mathcal{L}_{\rm BCE}$ (8)	305	0.7460	18.04	776	0.7448	19.23
(vii)w/o $\mathcal{L}_{\rm MOD}$ (7)	143	0.8365	25.97	210	0.9228	17.76

5.4 Parameter Analysis

We also tested the effects of $\{\alpha, \lambda\}$ in the pre-training objective (9) by adjusting $\alpha \in \{0.01, 0.1, 1, 10, 100\}$ and $\lambda \in \{0.1, 1, 10, 100, 1000\}$. The example parameter analysis results on DBLP and Amazon in terms of modularity are shown in Fig. 4, where different settings of $\{\alpha, \lambda\}$ would not significantly affect the quality of PRoCD. The influence of $\{\alpha, \lambda\}$ may also differ from refinement methods. For instance, *LPA* is more sensitive to the parameter setting, compared with *InfoMap* and *Locale*. Further analysis about the number of sampled node pairs n_S (in Algorithm 1) is given in Appendix D.

6 CONCLUSION

In this paper, we explored the potential of DGL to achieve a better trade-off between the quality and efficiency of K-agnostic CD. By reformulating this task as the binary node pair classification, a simple yet effective PRoCD method was proposed. It follows a novel *pre-training & refinement* paradigm, including the (i) *offline pre-training* on small synthetic graphs with various topology properties and high-quality ground-truth as well as the (ii) *online generalization* to and (iii) *refinement* on large real graphs without additional model optimization. Extensive experiments demonstrate that PRoCD, combined with different refinement methods, can achieve higher inference efficiency without significant quality degradation on public real graphs with various scales.

REFERENCES

[1] Rosa I Arriaga and Santosh Vempala. 2006. An algorithmic theory of learning: Robust concepts and random projection. *Machine Learning* 63 (2006), 161–182.

[2] Ayan Kumar Bhowmick, Koushik Meneni, Maximilien Danisch, Jean-Loup Guillaume, and Bivas Mitra. 2020. Louvainne: Hierarchical louvain method for high quality and scalable network embedding. In *WSDM*. 43–51.

[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 & Experiment* 2008, 10 (2008), P10008.

[4] Yuxuan Cao, Jiarong Xu, Carl J. Yang, Jiaan Wang, Yunchao Zhang, Chunping Wang, Lei Chen, and Yang Yang. 2023. When to Pre-Train Graph Neural Networks? From Data Generation Perspective!. In *SIGKDD*. 142–153.

[5] Jie Chen, Yousef Saad, and Zechen Zhang. 2022. Graph coarsening: from scientific computing to machine learning. *SeMA Journal* (2022), 1–37.

[6] Zhengdao Chen, Joan Bruna, and Lisha Li. 2019. Supervised community detection with line graph neural networks. In *ICLR*.

[7] Petr Chunaev, Timofey Gradov, and Klavdiya Bochenina. 2020. Community detection in node-attributed social networks: How structure-attributes correlation affects clustering quality. *Procedia Computer Science* 178 (2020), 355–364.

[8] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical Review E* 70, 6 (2004), 066111.

[9] Lin Dai and Bo Bai. 2017. Optimal decomposition for large-scale infrastructure-based wireless networks. *TWC* 16, 8 (2017), 4956–4969.

[10] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. 2007. Weighted graph cuts without eigenvectors a multilevel approach. *TPAMI* 29, 11 (2007), 1944–1957.

[11] Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3-5 (2010), 75–174.

[12] Santo Fortunato and Mark EJ Newman. 2022. 20 years of network community detection. *Nature Physics* 18, 8 (2022), 848–850.

[13] Yu Gao, Meng Qin, Yibin Ding, Li Zeng, Chaorui Zhang, Weixi Zhang, Wei Han, Rongqian Zhao, and Bo Bai. 2023. RaftGP: Random Fast Graph Partitioning. In *HPEC*. 1–7.

[14] Jiashun Jin, Zheng Tracy Ke, and Shengming Luo. 2021. Improvements on score, especially for weak signals. *Sankhya A* (2021), 1–36.

[15] Brian Karrer and Mark EJ Newman. 2011. Stochastic blockmodels and community structure in networks. *Physical Review E* 83, 1 (2011), 016107.

[16] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[17] Kai Lei, Meng Qin, Bo Bai, Gong Zhang, and Min Yang. 2019. GCN-GAN: A non-linear temporal link prediction model for weighted dynamic networks. In *InfoCom*. 388–396.

[18] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. 2009. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics* 6, 1 (2009), 29–123.

[19] Jiongqian Liang, Saket Gurukar, and Srinivasan Parthasarathy. 2021. Mile: A multi-level framework for scalable graph embedding. In *AAAI Conference on Web & Social Media*, Vol. 15. 361–372.

[20] Yue Liu, Wenxuan Tu, Sihang Zhou, Xinwang Liu, Linxuan Song, Xihong Yang, and En Zhu. 2022. Deep graph clustering via dual correlation reduction. In *AAAI*, Vol. 36. 7603–7611.

[21] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. 2023. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Web Conference*. 417–428.

[22] Yuanfu Lu, Xunqiang Jiang, Yuan Fang, and Chuan Shi. 2021. Learning to pre-train graph neural networks. In *AAAI*, Vol. 35. 4276–4284.

[23] Azade Nazi, Will Hang, Anna Goldie, Sujith Ravi, and Azalia Mirhoseini. 2019. Gap: Generalizable approximate graph partitioning framework. *arXiv:1903.00614* (2019).

[24] Mark EJ Newman. 2006. Modularity and Community Structure in Networks. *PNAS* 103, 23 (2006), 8577–8582.

[25] Mark EJ Newman and Aaron Clauset. 2016. Structure and inference in annotated networks. *NC* 7, 1 (2016), 11863.

[26] Tiago P Peixoto. 2014. Efficient Monte Carlo and greedy heuristic for the inference of stochastic block models. *Physical Review E* 89, 1 (2014), 012804.

[27] Chengbin Peng, Zhihua Zhang, Ka-Chun Wong, Xiangliang Zhang, and David Keyes. 2015. A scalable community detection algorithm for large graphs using stochastic block models. In *IJCAI*. 2090–2096.

[28] Meng Qin, Di Jin, Kai Lei, Bogdan Gabrys, and Katarzyna Musial-Gabrys. 2018. Adaptive community detection incorporating topology and content in social networks. *Knowledge-Based Systems* 161 (2018), 342–356.

[29] Meng Qin and Kai Lei. 2021. Dual-channel hybrid community detection in attributed networks. *Information Sciences* 551 (2021), 146–167.

[30] Meng Qin, Kai Lei, Bo Bai, and Gong Zhang. 2019. Towards a profiling view for unsupervised traffic classification by exploring the statistic features and link patterns. In *SIGCOMM NetAI Workshop*. 50–56.

[31] Meng Qin and Dit-Yan Yeung. 2023. Temporal link prediction: A unified framework, taxonomy, and review. *CSUR* 56, 4 (2023), 1–40.

[32] Meng Qin, Chaorui Zhang, Bo Bai, Gong Zhang, and Dit-Yan Yeung. 2023. High-quality temporal link prediction for weighted dynamic graphs via inductive embedding aggregation. *TKDE* (2023).

[33] Meng Qin, Chaorui Zhang, Bo Bai, Gong Zhang, and Dit-Yan Yeung. 2023. Towards a better trade-off between quality and efficiency of community detection: An inductive embedding method across graphs. *TKDD* (2023).

[34] Tai Qin and Karl Rohe. 2013. Regularized spectral clustering under the degree-corrected stochastic blockmodel. *NIPS*, 3120–3128.

[35] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *SIGKDD*. 1150–1160.

[36] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E* 76, 3 (2007), 036106.

[37] Jörg Reichardt and Stefan Bornholdt. 2006. Statistical mechanics of community detection. *Physical Review E* 74, 1 (2006), 016110.

[38] Yu Rong, Tingyang Xu, Junzhou Huang, Wenbing Huang, Hong Cheng, Yao Ma, Yiqi Wang, Tyler Derr, Lingfei Wu, and Tengfei Ma. 2020. Deep graph learning: Foundations, advances and applications. In *SIGKDD*. 3555–3556.

[39] Giulio Rossetti and Rémy Cazabet. 2018. Community discovery in dynamic networks: a survey. *CSUR* 51, 2 (2018), 1–37.

[40] Martin Rosvall and Carl T Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *PNAS* 105, 4 (2008), 1118–1123.

[41] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *TODS* 42, 3 (2017), 1–21.

[42] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. 2006. BioGRID: a general repository for interaction datasets. *Nucleic Acids Research* 34, suppl_1 (2006), D535–D539.

[43] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. 2022. Gppt: Graph pre-training and prompt tuning to generalize graph neural networks. In *SIGKDD*. 1717–1727.

[44] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2023. All in One: Multi-Task Prompting for Graph Neural Networks. In *SIGKDD*. 2120–2131.

[45] Xiangguo Sun, Jiawen Zhang, Xixi Wu, Hong Cheng, Yun Xiong, and Jia Li. 2023. Graph Prompt Learning: A Comprehensive Survey and Beyond. *arXiv:2311.16534* (2023).

[46] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. 2023. Graph clustering with graph neural networks. *JMLR* 24, 127 (2023), 1–21.

[47] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies* 1, 1 (2020), 396–413.

[48] Po-Wei Wang and J Zico Kolter. 2020. Community detection using fast low-cardinality semidefinite programming. *NIPS* 33 (2020), 3374–3385.

[49] Bryan Wilder, Eric Ewing, Bistra Dilkina, and Milind Tambe. 2019. End to end learning and optimization on graphs. *NIPS* 32 (2019).

[50] Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, and Stan Z Li. 2021. Self-supervised learning on graphs: Contrastive, generative, or predictive. *TKDE* (2021).

[51] Yuyang Xie, Yuxiao Dong, Jiezhong Qiu, Wenjian Yu, Xu Feng, and Jie Tang. 2023. SketchNE: Embedding Billion-Scale Networks Accurately in One Hour. *TKDE* (2023).

[52] Su Xing, Xue Shan, Liu Fanzhen, Wu Jia, Yang Jian, Zhou Chuan, Hu Wenbin, Paris Cecile, Nepal Surya, Jin Di, et al. 2022. A comprehensive survey on community detection with deep learning. *TNNLS* (2022), 1–21.

[53] Jaewon Yang and Jure Leskovec. 2012. Defining and evaluating network communities based on ground-truth. In *SIGKDD*. 1–8.

[54] Liang Yang, Xiaochun Cao, Dongxiao He, Chuan Wang, Xiao Wang, and Weixiong Zhang. 2016. Modularity based community detection with deep learning. In *IJCAI*, Vol. 16. 2252–2258.

[55] Liu Yue, Xia Jun, Zhou Sihang, Wang Siwei, Guo Xifeng, Yang Xihong, Liang Ke, Tu Wenxuan, Liu Xin Wang, et al. 2022. A survey of deep graph clustering: Taxonomy, challenge, and application. *arXiv:2211.12875* (2022).

[56] Ziwei Zhang, Peng Cui, Haoyang Li, Xiao Wang, and Wenwu Zhu. 2018. Billion-scale network embedding with iterative random projection. In *ICDM*. 787–796.

[57] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2020. Deep learning on graphs: A survey. *TKDE* 34, 1 (2020), 249–270.

[58] Zhili Zhao, Zhengyou Ke, Zhuoyue Gou, Hao Guo, Kunyuan Jiang, and Ruisheng Zhang. 2022. The trade-off between topology and content in community detection: An adaptive encoder–decoder-based NMF approach. *Expert Systems with Applications* 209 (2022), 118230.

Table 6: Parameter settings of the synthetic graph generator.

T	N	K	deg_{min}	deg_{max}	γ	μ	ρ
1e3	I(2e3, 5e3)	I(2, 1e3)	$\min\{5, \lceil N/(4K) \rceil\}$	$\min\{5e2, \lceil N/K \rceil\}$	F(2, 3.5)	F(2.5, 5)	F(1, 3)

Table 7: Statistics of the generated synthetic pre-training graphs.

Min	Max	Avg N	Min	Max	Avg E	Min	Max	Avg K	Avg deg_{min}	Avg deg_{max}	Min	Max	Avg Density
2,000	5,000	3,495.1	2,049	66,355	12,115.3	2	968	485.3	1.3	20.8	4e-4	1e-2	2e-3

```
Algorithm 4: Generating a Synthetic Graph via DC-SBM
Input: number of nodes $N_t$; number of communities $K_t$; range
       $[\text{deg}_{\text{min}}, \text{deg}_{\text{max}}]$ for node degrees; $\gamma$ for degree distribution;
       $\mu$ for numbers of within-community edges and
       between-community edges; $\rho$ for community size
       distribution
Output: a synthetic graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$
1  generate node set $\mathcal{V}_t = \{v_1, \cdots, v_{N_t}\}$
2  for $r$ from 1 to $K_t$ do
3      get size of the $r$-th community $|C_r^{(t)}| \sim \text{Dirichlet}(10/\rho)$
4      initialize $C_r^{(t)}$'s degree sum $\varphi_r \leftarrow 0$
5  initialize total degree sum $\bar{\varphi} \leftarrow 0$
6  for each node $v_i \in \mathcal{V}_t$ do
7      get $v_i$'s community label $c_i \sim \text{Multi}(|C_1^{(t)}|, \cdots, |C_{K_t}^{(t)}|)$
8      add $v_i$ into corresponding community $C_{c_i}$
9      get $v_i$'s degree
         $\text{deg}(v_i) \sim \{k^\gamma / \sum_{l=\text{deg}_{\text{min}}}^{\text{deg}_{\text{max}}} l^\gamma | \text{deg}_{\text{min}} \le k \le \text{deg}_{\text{max}}\}$
10     update degree sum $\varphi_{c_i} \leftarrow \varphi_{c_i} + \text{deg}(v_i)$
11     update total degree sum $\bar{\varphi} \leftarrow \bar{\varphi} + \text{deg}(v_i)$
12 for each node $v_i \in \mathcal{V}_t$ do
13     get $v_i$'s degree correction value $\theta_i \leftarrow \text{deg}(v_i)/\varphi_{c_i}$
14 for $r$ from 1 to $K_t$ do
15     for $t$ from $r$ to $K_t$ do
16         if $r = t$ then
17             $\Omega_{\text{rr}} \leftarrow \frac{\mu}{1+\mu} \cdot \varphi_r$
18         else
19             $\Omega_{\text{rt}} \leftarrow \frac{1}{1+\mu} \cdot \frac{\varphi_r \varphi_t}{\bar{\varphi}}$
20 for $i$ from 2 to $N_t$ do
21     for $j$ from 1 to $i$ do
22         generate an edge $(v_i, v_j)$ for $\mathcal{E}_t$ via $\text{Poisson}(\theta_i \theta_j \Omega_{c_i c_j})$
```

A GENERATION OF SYNTHETIC PRE-TRAINING GRAPHS

Our settings of the DC-SBM generator are summarized in Table 6, where '$I(a, b)$' and '$F(a, b)$' denote an integer and a float uniformly sampled from the range $[a, b]$; T is the number of synthetic graphs; N and K are the numbers of nodes and communities in each graph; deg_{min} and deg_{max} are the minimum and maximum node degrees of a graph; γ is a parameter to control the power-law distributions of node degrees with $\text{deg}(v_i) \sim \{k^\gamma / \sum_{l=\text{deg}_{\text{min}}}^{\text{deg}_{\text{max}}} l^\gamma\}$ ($\text{deg}_{\text{min}} \le k \le \text{deg}_{\text{max}}$); μ is a ratio between the number of within-community and between-community edges; ρ is a parameter to control the heterogeneity of community size, with the size of each community following $\text{Dirichlet}(10/\rho)$.

Table 8: Parameter settings & layer configurations of PRoCD.

Datasets	Parameter Settings			Layer Configurations			
	(α, λ)	(n_P, η)	n_S	d	L_{Feat}	L_{GNN}	L_{BC}
Protein	(0.1, 1)	(20, 1e-4)	1e3	64	2	4	2
ArXiv	(1, 100)	(100, 1e-4)	1e4			2	6
DBLP	(0.1, 10)	(100, 1e-4)	1e4			4	2
Amazon	(0.1, 10)	(100, 1e-4)	2e4			4	2
Youtube	(1e2, 1e2)	(89, 1e-4)	1e4			4	2
RoadCA	(1e2, 10)	(11, 1e-4)	1e4			2	2

For each synthetic graph $\mathcal{G}_t$, suppose there are N_t nodes partitioned into K_t communities, with N_t and K_t sampled from the corresponding distributions in Table 6. Let $C^{(t)} = \{C_1^{(t)}, \cdots, C_{K_t}^{(t)}\}$ denote the community ground-truth of $\mathcal{G}_t$. We also sample (γ, μ, ρ) from corresponding distributions in Table 6, which define the topology properties of $\mathcal{G}_t$. Algorithm 4 summarizes the procedure of generating a graph $\mathcal{G}_t$ via DC-SBM, based on the parameter settings of $\{N_t, K_t, \text{deg}_{\text{min}}, \text{deg}_{\text{max}}, \gamma, \mu, \rho\}$. Concretely, DC-SBM generates an edge (v_i, v_j) in each graph $\mathcal{G}_t$ via $\mathbf{A}_{ij}^{(t)} \sim \text{Poisson}(\theta_i \theta_j \Omega_{c_i c_j})$, where c_i is the community label of node v_i; $\theta \in \mathbb{R}_+^{N_t}$ and $\Omega \in \mathbb{R}_+^{K_t \times K_t}$ are the degree correction vector and block interaction matrix [15] determined by $\{\gamma, \mu, \rho\}$; $\theta_i \theta_j \Omega_{c_i c_j}$ represents the expected number of edges between v_i and v_j.

Statistics of the generated synthetic graphs (used for *offline pre-training* in our experiments) are reported in Table 7.

B DETAILED COMPLEXITY ANALYSIS

For each graph $\mathcal{G}'$, let N, M, and d be the number of nodes, number of edges, and embedding dimensionality, respectively. By using the efficient sparse-dense matrix multiplication, the complexity of the feature extraction defined in (3) is $O(Md + Nd^2)$. Similarly, the complexity of one FFP of the embedding encoder described by (4) is no more than $O(Md + Nd^2)$.

To derive a feasible CD result via Algorithm 1, the complexities of constructing the node pair set $\mathcal{P}$, one FFP of the *binary classifier* to derive auxiliary graph $\tilde{\mathcal{G}}$, and extracting connected components via DFS/BFS are $O(M + n_S)$, $O((M + n_S)d^2)$, and $O(N + \tilde{M})$, where n_S is the number of randomly sampled node pairs; $\tilde{M} := |\tilde{\mathcal{E}}|$ is the number of edges in the auxiliary graph $\tilde{\mathcal{G}}$.

In summary, the complexity of *online generalization* is $O((Md + Nd^2) + (Md + Nd^2) + (M + n_S) + (M + n_S)d^2 + (N + \tilde{M})) = O((N + M)d^2)$, where we assume that $\tilde{M} \approx M$, and $n_S \ll M$. In particular, the feature extraction and FFP of the model can be significantly speeded up via GPUs.

C DETAILED EXPERIMENT SETUP

Demo code of experiments is anonymously provided[2]. We will make the code, data, and check points public if accepted.

[2] https://anonymous.4open.science/r/PRoCD-DEFE

C.1 Datasets

In the six real datasets (see Table 2), *Protein*[3] [42] was collected based on the protein-protein interactions in the BioGRID repository. *ArXiv*[4] [47] and *DBLP*[5] [53] are public paper citation and collaboration graphs extracted from ArXiv and DBLP, respectively. *Amazon*[6] [53] was collected by crawling the product co-purchasing relations from Amazon, while *Youtube*[7] [53] was constructed based on the friendship relations of Youtube. *RoadCA*[8] [18] describes a road network in California.

To pre-process *Protein* based on the records from BioGRID, we abstracted each protein as a unique node and constructed graph topology based on corresponding protein-protein interactions. Since there are multiple connected components in the extracted graph, we extracted the largest component for evaluation. Furthermore, we directly used the original formats of the remaining datasets provided by their sources.

C.2 Experiment Environments & Implementation Details

All the experiments were conducted on a server with one AMD EPYC 7742 64-Core CPU, 512GB main memory, and one 80GB memory GPU. We used Python 3.7 to implement the proposed PRoCD. In particular, the feature extraction module (3), embedding encoder (4), and binary node pair classifier (6) were implemented via PyTorch 1.10.0. In this setting, the feature extraction and FFP of PRoCD can be speeded up via the GPU. The efficient function 'scipy.sparse.csgraph.connected_components' was used to extract connected components in Algorithm 1 to derive a feasible CD result.

For all the baselines, we adopted their official implementations and tuned their parameters to report the best quality. To ensure the fairness of comparison, the embedding dimensionality of all the embedding-based methods (i.e., *LouvainNE*, *SketchNE*, *RaftGP*, *ICD*, and **PRoCD**) was set to be the same (i.e., 64).

C.3 Parameter Settings & Layer Configurations

The recommended parameter settings and layer configurations of PRoCD are depicted in Table 8, where α and λ are hyper-parameters in the pre-training objective (9); n_P and η are the number of epochs and learning rate of pre-training in Algorithm 2; n_S is the number of randomly sampled node pairs in Algorithm 1 for inference; d is the embedding dimensionality; L_{Feat}, L_{GNN}, and L_{BC} are the numbers of MLP layers in (3), GNN layers in (4), and MLP layers in (6).

Given an MLP, let $\mathbf{u}^{[s-1]}$ and $\mathbf{u}^{[s]}$ be the input and output of the s-th perceptron layer. The MLP in the feature extraction module (3) is defined as

$$\mathbf{u}^{[s]} = \tanh(\mathbf{u}^{[s-1]}\mathbf{W}^{[s]} + \mathbf{b}^{[s]}), \qquad (10)$$

with $\{\mathbf{W}^{[s]} \in \mathbb{R}^{d \times d}, \mathbf{b}^{[s]} \in \mathbb{R}^{d}\}$ as trainable parameters. For the MLP $h_s(\cdot)$ (or $h_d(\cdot)$) in the binary classifier (6), suppose there are L layers. The s-th layer ($s < L$) and the last layer of $h_s(\cdot)$ (or $h_d(\cdot)$) are defined as

$$\mathbf{u}^{[s]} = \tanh(\mathbf{u}^{[s-1]}\mathbf{W}^{[s]} + \mathbf{b}^{[s]}) + \mathbf{u}^{[s-1]}, \qquad (11)$$

$$\mathbf{u}^{[L]} = \mathrm{ReLU}(\mathbf{u}^{[L-1]}\mathbf{W}^{[L]} + \mathbf{b}^{[L]}), \qquad (12)$$

where there is a skip connection in each perceptron layer except the last layer.

[3]https://downloads.thebiogrid.org/BioGRID
[4]https://ogb.stanford.edu/docs/nodeprop/
[5]https://snap.stanford.edu/data/com-DBLP.html
[6]https://snap.stanford.edu/data/com-Amazon.html
[7]https://snap.stanford.edu/data/com-Youtube.html
[8]https://snap.stanford.edu/data/roadNet-CA.html

Table 9: Detailed inference time (sec) of graph coarsening (GC) baselines.

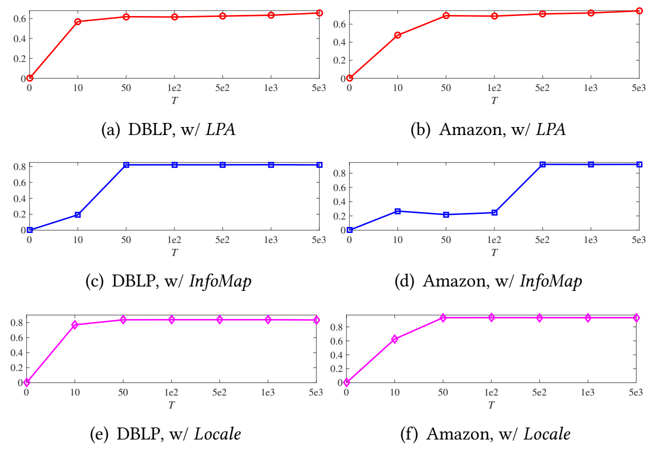
Datasets	Methods	GC	Rfn
Protein	GC+*LPA*	15.20	17.32
	GC+*InfoMap*		18.64
	GC+*Locale*		29.15
ArXiv	GC+*LPA*	10.20	11.37
	GC+*InfoMap*		14.08
	GC+*Locale*		34.97
DBLP	GC+*LPA*	17.84	12.20
	GC+*InfoMap*		10.15
	GC+*Locale*		19.03
Amazon	GC+*LPA*	17.19	12.21
	GC+*InfoMap*		10.82
	GC+*Locale*		12.90
Youtube	GC+*LPA*	31.84	37.32
	GC+*InfoMap*		73.01
	GC+*Locale*		162.65
RoadCA	GC+*LPA*	71.33	35.56
	GC+*InfoMap*		58.41
	GC+*Locale*		59.24

Table 10: Detailed inference time (s) of embedding baselines.

Datasets	Methods	Emb	Clus
Protein	LouvainNE+DBSCAN	4.13	73.07
	SketchNE+DBSCAN	4.30	11.88
	ICD-C+DBSCAN	6.99	82.31
	ICD-M+DBSCAN	12.49	75.97
ArXiv	LouvainNE+DBSCAN	6.86	207.41
	SketchNE+DBSCAN	5.79	537.12
	ICD-C+DBSCAN	21.59	274.43
	ICD-M+DBSCAN	19.57	274.55
DBLP	LouvainNE+DBSCAN	10.52	99.35
	SketchNE+DBSCAN	7.71	37.90
	ICD-C+DBSCAN	5.66	1789.38
	ICD-M+DBSCAN	11.53	973.45
Amazon	LouvainNE+DBSCAN	10.79	53.67
	SketchNE+DBSCAN	8.05	41.79
	ICD-C+DBSCAN	16.26	2011.04
	ICD-M+DBSCAN	23.22	848.89
Youtube	LouvainNE+DBSCAN	34.98	635.55
	SketchNE+DBSCAN	19.40	466.32
	ICD-C+DBSCAN	10.43	OOT
	ICD-M+DBSCAN	7.66	OOT
RoadCA	LouvainNE+DBSCAN	56.30	180.15
	SketchNE+DBSCAN	25.11	1385.80
	ICD-C+DBSCAN	14.49	OOT
	ICD-M+DBSCAN	12.90	OOT

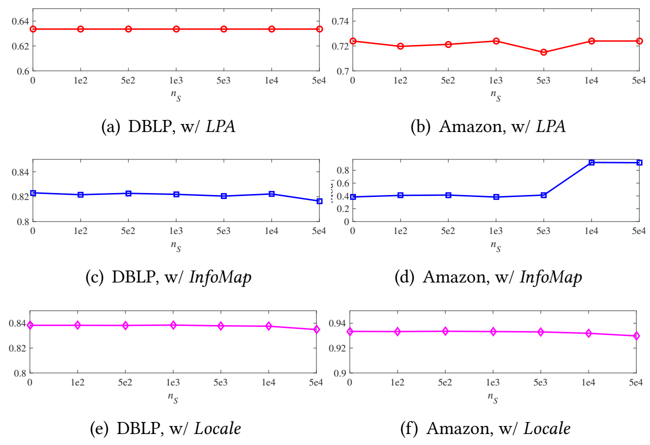
Table 11: Ablation study w.r.t. binary classification threshold on DBLP and Amazon in terms of modularity↑.

			0.1	0.3	**0.5**	0.7	0.9
DBLP	**Eq. (6)**	w/ LPA	0.2512	0.6790	0.6336	0.6003	0.5626
		w/ IM	0.0121	0.8154	0.8222	0.8207	0.8187
		w/ Lcl	0.3220	0.8282	0.8376	0.8383	0.8379
	Eq. (5)	w/ LPA	0.0000	0.0000	0.0009	0.6034	0.0000
		w/ IM	0.0000	0.0000	0.0009	0.8212	0.0000
		w/ Lcl	0.0000	0.0000	0.0009	0.8374	0.0000
Amazon	**Eq. (6)**	w/ LPA	0.4498	0.7787	0.7240	0.6667	0.5793
		w/ IM	0.1778	0.9205	0.9222	0.9216	0.9203
		w/ Lcl	0.4882	0.9283	0.9319	0.9321	0.9315
	Eq. (5)	w/ LPA	0.0000	0.0000	0.0008	0.6506	0.0000
		w/ IM	0.0000	0.0000	0.0008	0.9214	0.0000
		w/ Lcl	0.0000	0.0000	0.0008	0.9318	0.0000



(a) DBLP, w/ *LPA* (b) Amazon, w/ *LPA* (c) DBLP, w/ *InfoMap* (d) Amazon, w/ *InfoMap* (e) DBLP, w/ *Locale* (f) Amazon, w/ *Locale*

Figure 5: Ablation study w.r.t. the number of pre-training graphs T on DBLP and Amazon in terms of modularity↑.



(a) DBLP, w/ *LPA* (b) Amazon, w/ *LPA* (c) DBLP, w/ *InfoMap* (d) Amazon, w/ *InfoMap* (e) DBLP, w/ *Locale* (f) Amazon, w/ *Locale*

Figure 6: Prameter analysis w.r.t. the number of sampled node pairs n_S on DBLP and Amazon in terms of modularity↑.

D FURTHER EXPERIMENT RESULTS

Detailed analysis about the inference time (sec) of (i) graph coarsening (GC) and (ii) embedding baselines is depicted in Tables 9 and 10, where 'GC', 'Rfn', 'Emb', and 'Clus' denote the time of (i) graph coarsening, (ii) online refinement, (iii) embedding derivation, and (iv) downstream clustering (i.e., via DBSCAN).

To further verify the significance of offline pre-training in our PRoCD method, we conducted additional ablation study by setting the number of pre-training graphs $T \in \{0, 10, 50, 1e2, 5e2, 1e3, 5e3\}$. The corresponding results on DBLP and Amazon in terms of modularity are reported in Fig. 5. Compared with our standard setting (i.e., $T = 1e3$), there are significant quality declines for all the variants of PRoCD when there are too few pre-training graphs (e.g., $T < 50$). It implies that *the offline pre-training (with enough synthetic graphs) is essential to ensure the high inference quality of PRoCD*.

We further validated our design of classifier (6) for binary node pair classification by varying the threshold (in line 8 of Algorithm 1) from 0.1 to 0.9 with a step size of 0.2. We compared the quality with that of the naive classifier (5) using the same experiment settings. Ablation study results on DBLP and Amazon are reported in Table 11. For the proposed binary classifier (6), a small classification threshold (e.g., 0.1) may easily result in poor CD quality. 0.5 is an appropriate default value for all the variants of PRoCD to derive high-quality results. In contrast, most settings for the naive classifier (5) lead to poor quality. Even the best setting (i.e., 0.7) cannot outperform that of (6). Therefore, *our binary classifier (6) with a novel adaptive temperature parameter is essential for PRoCD.*

We also conducted further parameter analysis for the number of sampled node pairs (in Algorithm 1) for inference, where we set $n_S \in \{0, 1e2, 5e2, 1e3, 5e3, 1e4, 5e4\}$. Results on DBLP and Amazon in terms of modularity are shown in Fig. 6. In most cases except the variant with *InfoMap* on Amazon, our PRoCD method is not sensitive to the setting of n_S. To ensure the inference quality of the exception case, one needs a large setting of n_S (e.g., $n_S \geq 1e4$).

E FUTURE DIRECTIONS

Some possible future directions are summarized as follows.

Theoretical Analysis. This study empirically verified the potential of PRoCD (with a novel *pre-training & refinement* paradigm) to achieve a better trade-off between the quality and efficiency of K-agnostic. However, most existing graph pre-training techniques lack theoretical guarantee for their transfer ability w.r.t. different pre-training data and downstream tasks. In our future work, we plan to theoretically analyze the transfer ability of PRoCD from small synthetic pre-training graphs to large real graphs, following previous analysis on random graph models (e.g., SBM [14, 34]).

Extension to Dynamic Graphs. In this study, we considered K-agnostic CD on static graphs. CD on dynamic graphs [39] is a more challenging setting, involving the variation of nodes, edges, and community membership over time. Usually, one can formulate a dynamic graph as a sequence of static snapshots with similar underlying properties [17, 31, 32]. PRoCD can be easily extended to handle dynamic CD by directly generalizing the pre-trained model to each snapshot for fast online inference, without any re-training. Further validation of this extension is also our next focus.

Integration of Graph Attributes. As stated in Section 3, we considered CD without available graph attributes. A series of previous studies [7, 25, 28, 29, 58] have demonstrated the complicated correlations between graph topology and attributes, which are inherently heterogeneous information sources, for CD. On the one hand, the integration of attributes may provide complementary characteristics for better CD quality. On the other hand, *it may also incorporate noise or inconsistent features that lead to unexpected quality decline compared with those only considering one source.* In our future work, we will explore the *adaptive integration of attributes* for PRoCD. Concretely, when attributes match well with topology, we expect that PRoCD can fully utilize the complementary information of attributes to improve CD quality. *When the two sources mismatch with one another, we try to adaptively control the contribution of attributes to avoid quality decline.*

Scaling Up to Ultra-large Graphs. In experiments, we demonstrated the scalability of PRoCD on real graphs with various scales, where the number of nodes N increases from 8×10^4 to 10^6. In our future work, we intend to further validate its scalability on real graphs with ultra-large Ns (e.g., billion nodes).