Efficient Unsupervised Community Search with Pre-trained Graph Transformer

Jianwei Wang
University of New South Wales
jianwei.wang1@unsw.edu.au

Kai Wang
ACEM, Shanghai Jiao Tong University
w.kai@sjtu.edu.cn

Xuemin Lin
ACEM, Shanghai Jiao Tong University
xuemin.lin@sjtu.edu.cn

Wenjie Zhang
University of New South Wales
wenjie.zhang@unsw.edu.au

Ying Zhang
Zhejiang Gongshang University
ying.zhang@zjgsu.edu.cn

ABSTRACT

Community search has aroused widespread interest in the past decades. Among existing solutions, the learning-based models exhibit outstanding performance in terms of accuracy by leveraging labels to 1) train the model for community score learning, and 2) select the optimal threshold for community identification. However, labeled data are not always available in real-world scenarios. To address this notable limitation of learning-based models, we propose a pre-trained graph **Trans**former based community search framework that uses **Zero** label (i.e., unsupervised), termed *TransZero*. *TransZero* has two key phases, i.e., the offline pre-training phase and the online search phase. Specifically, in the offline pre-training phase, we design an efficient and effective community search graph transformer (*CSGphormer*) to learn node representation. To pre-train *CSGphormer* without the usage of labels, we introduce two self-supervised losses, i.e., personalization loss and link loss, motivated by the inherent uniqueness of node and graph topology, respectively. In the online search phase, with the representation learned by the pre-trained *CSGphormer*, we compute the community score without using labels by measuring the similarity of representations between the query nodes and the nodes in the graph. To free the framework from the usage of a label-based threshold, we define a new function named expected score gain to guide the community identification process. Furthermore, we propose two efficient and effective algorithms for the community identification process that run without the usage of labels. Extensive experiments over 10 public datasets illustrate the superior performance of *TransZero* regarding both accuracy and efficiency.

PVLDB Reference Format:
Jianwei Wang, Kai Wang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. Efficient Unsupervised Community Search with Pre-trained Graph Transformer. PVLDB, 17(9): 2227 - 2240, 2024.
doi:10.14778/3665844.3665853

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/guaiyoui/TransZero.

Proceedings of the VLDB Endowment, Vol. 17, No. 9 ISSN 2150-8097.
doi:10.14778/3665844.3665853

Table 1: Comparison among CS methods

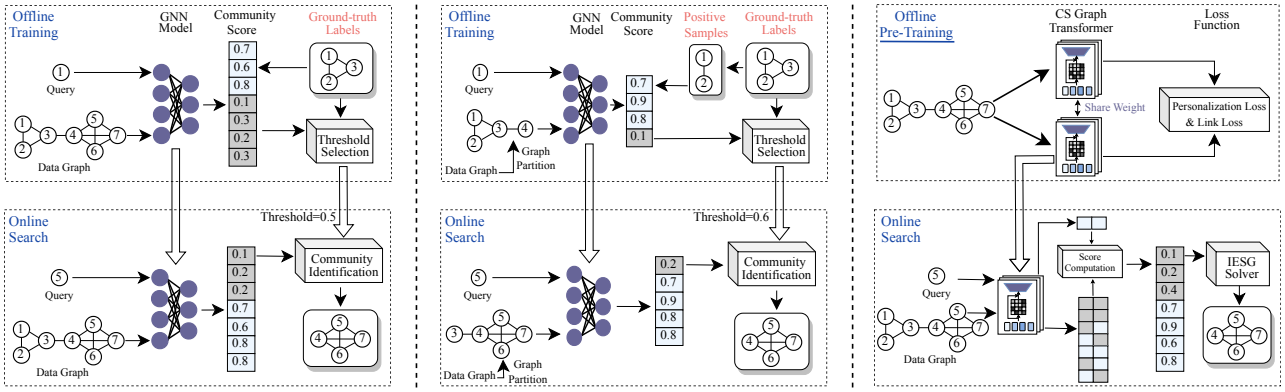
Methods	Label Free?	Structure Flexibility?	Backbone Model	Loss Function
CST [15]	✓	✗	k-core	-
EquiTruss [2]	✓	✗	k-truss	-
MkECS [3]	✓	✗	k-ECC	-
CTC [27]	✓	✗	k-truss	-
QD-GNN [28]	✗	✓	GNN	Binary Cross Entropy
COCLEP [32]	✗	✓	GNN	Contrastive Loss
TransZero (our)	✓	✓	Graph Transformer	Contrastive Loss & Generative Loss

1 INTRODUCTION

Graphs play a prominent role in modeling relationships between entities in a system and are applied across diverse domains such as social networks [9, 34, 45, 52], biology networks [39, 49] and finance networks [8, 13, 58]. As a fundamental problem in graph analytics, community search (CS) [19] has aroused widespread interest in the past decades. Given a set of query nodes, CS aims to find a query-dependent subgraph, with the resultant subgraph, also referred to as a community, manifesting as a densely intra-connected structure. CS is also relevant and widely applied for tasks in real-world applications, such as fraud detection in e-commerce platforms [35, 60] and protein complex identification [19, 37]. Given the importance and widespread applications of CS, a set of algorithms are proposed, which include the traditional CS algorithms [2, 10, 15, 25, 26, 44] and learning-based CS models [21, 28, 32, 33, 43].

As summarized in Table 1 and in the recent survey paper [19], most of existing traditional CS algorithms characterize the community structure by specific subgraph cohesiveness models such as k-core [15, 44], k-truss [2, 26, 27] and k-edge connected component (k-ECC) [3, 10, 25], and thus suffer from the limitation known as *structure inflexibility* [21, 28, 32]. These fixed subgraph models impose rigid constraints on the topological structure of communities, making it difficult for real-world communities to meet such inflexible constraints. For example, methods based on k-core require every node in the found community to have a degree larger than or equal to k, which may not be met by real-world communities, particularly for nodes located at the boundary of the community.

Recently, learning-based approaches such as QD-GNN [28] and COCLEP [32] are emerging in this field due to their outstanding performance in terms of accuracy. As illustrated in Figure 1(a) and



(a) QD-GNN (Supervised) (b) COCLEP (Semi-Supervised) (c) TransZero (Unsupervised)

Figure 1: Framework comparisons of learning-based methods for CS

Figure 1(b), QD-GNN trains a feature-aggregation model in a supervised manner where all nodes from the ground-truth community are used for training, and COCLEP trains a feature-aggregation model in a semi-supervised manner where only a subset nodes from the ground-truth community is used for training. By incorporating meticulous framework design and leveraging ground-truth information, both QD-GNN and COCLEP can effectively alleviate the limitation of *structure inflexibility* encountered by traditional CS algorithms, and thus demonstrate state-of-the-art performance in the supervised and semi-supervised settings, respectively. In a nutshell, they both employ a two-stage framework, consisting of an offline training phase and an online search phase, and rely on the labels to 1) train the neural network for *community score learning* where the community score reflects the membership of the corresponding node w.r.t. the query, and 2) select the optimal threshold from the labeled validation set for *community identification*.

Motivations. While learning-based approaches demonstrate phenomenal performance, a notable limitation of current learning-based methods is their dependence on ground-truth communities which are often unavailable or of low quality in real-world scenarios. Additionally, the excessive dependence on ground-truth communities makes it challenging for QD-GNN to generalize and predict unseen communities, as evaluated in Section 6. On the other hand, traditional CS algorithms operate without labels and thus demonstrate good generalization abilities to discover unseen communities. Therefore, a natural and promising idea is to develop a learning-based method that inherits the favorable properties of learning-based approaches, including flexible community structures, strong expressive capabilities and outstanding performance, and simultaneously combines the advantages of traditional CS algorithms, such as operating without labels and demonstrating good generalization ability. Hence, in this paper, we aim to design an efficient and effective learning-based approach for CS that runs without using labels. The challenges are mainly two-folds: 1) Challenge I: effectively learning the community score for each query without using labels, and 2) Challenge II: adaptively identifying the community without label-based optimal threshold.

One promising direction for community score learning without using labels is to use some unsupervised frameworks. However, the objective of existing unsupervised frameworks is incompatible with the task of community score learning, and existing works of CS cannot be easily extended to support unsupervised community score learning. Moreover, most of the existing unsupervised frameworks utilize message-passing-based Graph Neural Networks (GNNs) as the backbone. This choice inherently introduces challenges of the over-smoothing problem [11] and over-squashing problem [4] with the increment of model depth, consequently limiting their potential capability for graph representation learning as highlighted in [12]. Therefore, designing a community score learning method that operates without using labels presents a considerable challenge.

One direct approach for community identification without using labels is to assign a fixed hyper-parameter as the threshold directly. However, as demonstrated in the evaluation conducted in [28], utilizing different fixed thresholds can result in a maximum decrease of ~40% in the accuracy measured by the F1-score. Furthermore, the optimal threshold may vary across different datasets and different similarity metrics. Another promising approach is to select nodes with the top-K highest community scores. However, the size of real-world communities can differ significantly within one graph and across different graphs. It is hard to utilize one fixed size that suits all the communities. Therefore, it is challenging to design a community identification method that runs without using labels.

Our approaches. Driven by the aforementioned challenges, we propose a new pre-trained graph **Trans**former based community search framework that uses **Zero** label (i.e., unsupervised), termed *TransZero*. The overall illustration is in Figure 1(c). *TransZero* also contains two phases, i.e., the offline pre-training phase and the online search phase.

To address Challenge I, we introduce a two-step methodology that incorporates both the offline pre-training phase and the online community score computation module to obtain the community score without using labels. First of all, we pre-train the *CSGphormer* in the offline pre-training phase. Subsequently, we calculate the community score in the community score computation module by measuring the similarity between the representation of the query and the representation of each node within the graph where the representation is inferred by the learned *CSGphormer*. Specifically, we propose *CSGphormer* motivated by NAGphormer [12],

the existing state-of-the-art graph transformer. NAGphormer with other graph transformers have shown effectiveness for mitigating the over-smoothing and over-squashing problems [12, 59, 64]. To pre-train *CSGphormer* without using labels, we introduce two self-supervised losses designed specifically for CS, i.e., personalization loss and link loss, motivated by the inherent uniqueness of each node and graph topology, respectively. It combines the contrast-based self-supervised learning (i.e., personalization loss) [29, 63] and the generation-based self-supervised learning (i.e., link loss) [36, 53] to achieve better performance.

To solve Challenge II, we introduce a new function named expected score gain and formulate the problem of community identification as the problem of community Identification with Expected Score Gain (*IESG*). Specifically, the community score reflects the likelihood of a node being included in the community, and the objective is to identify a community where nodes exhibit high scores. QD-GNN and COCLEP use a label-based threshold to define high scores. To eliminate reliance on labels, we define expected score gain. Community scores that maximize the expected score gain are considered high. Motivated by modularity [30, 41] which is a classical metric for community cohesiveness, the expected score gain calculates the sum of node scores within the community minus the sum of expected scores if nodes are chosen randomly. A higher expected score gain value indicates a potentially better community. Based on this new function and inspired by the cohesive nature of communities and the query-dependent property of CS, *IESG* aims to find a connected subgraph that includes the query while having the maximum expected score gain value. Furthermore, we prove that the *IESG* is NP-hard, indicating that it cannot be solved in polynomial time. Therefore, we design two heuristic algorithms, i.e., *Local Search* and *Global Search* to effectively and efficiently identify promising communities without using labels.

Contributions. The main contributions are as follows:

- We propose a new learning-based CS framework *TransZero* that runs without using ground-truth communities. It contains the offline pre-training phase and the online search phase.
- In the offline pre-training phase, we design an efficient graph transformer *CSGphormer* for CS. Two self-supervised losses including the personalization loss and link loss are utilized to pre-train *CSGphormer* without using labels.
- In the online search phase, the score computation module first obtains the community score by measuring the similarity of the learned representation. Based on the new proposed expected score gain function, we model community identification as *IESG*, and propose two efficient and effective algorithms, i.e., *Local Search* and *Global Search*, to find promising communities.
- Experiments across 10 public datasets highlight the superiority of *TransZero* regarding both accuracy and efficiency. Under the hybrid training setting, *TransZero* that does not use labels even outperforms COCLEP and QD-GNN which rely on labels with an average F1-score improvement of 10.01% and 5.91%, respectively. Regarding offline training efficiency, *TransZero* achieves an average speedup of 122.39× and 118.22× compared to COCLEP and QD-GNN, respectively. Regarding online search efficiency, *TransZero* achieves an average speedup of 10.02× and 26.77× compared to COCLEP and QD-GNN, respectively.

Table 2: Symbols and Descriptions

Notation	Description
$G(V, E)$	an undirected graph
X, A	feature matrix and adjacency matrix
$q = V_q$	the query with node set V_q
$C_q, \tilde{C}_q$	the ground-truth/predicted community of q
$f^{\theta}(\cdot)$	neural network model with parameters θ
$S \in \mathbb{R}^{\lvert V\rvert}$	community score vector
$V_1 \backslash V_2$	nodes in V_1 but not V_2

2 PRELIMINARIES

In this section, we introduce preliminaries and the state-of-the-art CS models. The frequently used symbols are summarized in Table 2.

2.1 Problem Statement

We follow the typical setting of the general community search problem and focus on the undirected graph $G(V, E)$ where V is the node set and $E \subseteq V \times V$ is the edge set. We use $|V|$ and $|E|$ to denote the cardinality of V and E, respectively. The feature matrix is denoted as $X \in \mathbb{R}^{|V| \times d}$ where d is the dimension of the feature. $A \in \mathbb{R}^{|V| \times |V|}$ is the adjacency matrix where $A_{ij} = 1$ indicates the link between node v_i and node v_j. $S \in \mathbb{R}^{|V|}$ is used to denote the community score vector. We use q and V_q interchangeably to denote the query node set. C_q and $\tilde{C}_q$ are utilized to denote the ground-truth community and the predicted community w.r.t. q. Next, we give the formal definition of community search.

Definition 1. *(Community Search [28, 32]). Given a data graph $G(V, E)$ and query q, the task of Community Search (CS) aims to identify a query-dependent connected subgraph (i.e., community) C_q where nodes in the found community are densely intra-connected.*

2.2 State-of-the-art

The state-of-the-art learning-based CS models employ a framework comprising the offline training phase and the online search phase.

Offline Training. QD-GNN models the problem of CS as binary node classification. Specifically, given a data graph G and a training dataset $\mathcal{D}_{train} = \{q_i, C_{q_i}\}_{i=1}^{|\mathcal{D}_{train}|}$ containing a set of queries and the corresponding ground-truth communities, QD-GNN proposes a neural network model, denoted as $\mathcal{M}$, which takes the query, adjacency matrix and features as inputs and outputs a community score vector $S_q \in \mathbb{R}^{|V|}$. This vector indicates the membership likelihood of each node in the predicted community. And then, the Binary Cross Entropy (BCE) function is used to measure the divergence between S_q and the ground-truth vector Y_q. $Y_{q,j} = 1$ if and only if $v_j \in C_q$. Here, $Y_{q,j}$ is the j-th bit of Y_q.

$$\mathcal{L} = \sum_{q \in \mathcal{D}_{train}} \frac{1}{|V|} \sum_{i=1}^{|V|} -\left(Y_{q,i} log S_{q,i} + \left(1 - Y_{q,i}\right) log\left(1 - S_{q,i}\right)\right)$$

The parameters of the model are updated by gradient descent to minimize the loss between the predicted community score and the ground-truth vector. It is worth noting that the loss function

employed in COCLEP differs from the aforementioned loss. COCLEP employs contrastive learning and focuses on enhancing the prediction performance for the selected positive candidates. However, both QD-GNN and COCLEP employ ground-truth information within their respective loss functions for model training.

Following the model training, both QD-GNN and COCLEP determine the optimal parameters, particularly the community score threshold, by evaluating the validation set. Note that, the validation set also contains the ground-truth information.

Online search. The learned model and the selected optimal threshold from the offline training phase are utilized in the online search phase. Specifically, it first calculates the community score by the model inference. To identify the final community, QD-GNN proposes a Constrained Breadth-First Search algorithm in [28] which requires the label-based threshold as input. It expands outward and selects neighbors with a community score larger than the threshold.

3 OFFLINE PRE-TRAINING PHASE

Motivation. Given a graph with feature matrix X and adjacency matrix A, the objective of pre-training for CS is to learn a generic encoder that can encode the community information and the graph topology into the latent space. As we focus on the unsupervised CS, we resort to self-supervised learning which is an important category of unsupervised learning. Specifically, we consider both the generative self-supervised learning and the contrastive self-supervised learning to achieve a better performance [38].

3.1 Overview

The overall architecture of the offline pre-training phase is illustrated in Figure 2. Given a data graph, an augmented subgraph sampler is applied to generate the corresponding community-level subgraph for each select node. It aims to generate new data with maximum consistent features from different views w.r.t. a node in the graph as positive samples. Then, the augmented subgraph is sent into a graph encoder (*CSGphormer* as in our proposed *TransZero*) to extract the latent features that encode both the community information and graph topology. The graph encoder outputs both the node-level representation and community-level representation. The learned representations are used for loss computation which includes the personalization loss (i.e., contrastive loss) and the link loss (i.e., generative loss). The obtained loss is back-propagated to update the parameters in *CSGphormer*.

3.2 Augmented Subgraph Sampler

Motivation. The contrast-based self-supervised learning is based on the augmented subgraphs to design the loss function. COCLEP, an existing state-of-the-art semi-supervised CS model, constructs the augmented subgraphs by incorporating the ground-truth positive samples and their K-hop neighbors, where K is a fixed value. However, it is intuitive that various central nodes should have different neighborhood distances, necessitating a personalized selection of different hops for different nodes. To address this, we employ conductance [5, 57], a well-established measure of community cohesiveness, to enhance the choice of hops. The definition of conductance is as follows:

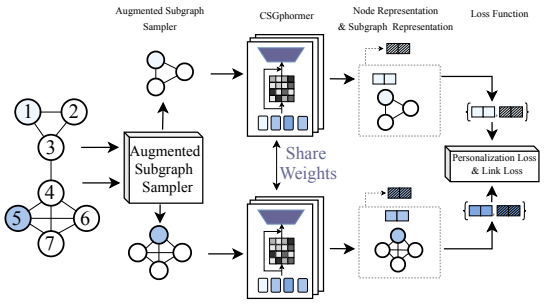


Figure 2: Illustration of the offline pre-training phase

Definition 2. *(Conductance [5, 57]). Given a graph $G(V, E)$ and a community C, the conductance of C is defined as:*

$$\Phi(G, C) = \frac{|e(C, \overline{C})|}{min(d_C, d_{\overline{C}})} \tag{1}$$

where $\overline{C} = V \backslash C$ is complement of C. $e(C, \overline{C})$ is the edges between nodes in C and nodes in $\overline{C}$. d_C is the sum of degrees of the nodes in C.

Conductance measures the fraction of the total edge volume that points outside the community. A smaller conductance means a higher ratio of information can be used for pre-training. Based on conductance, we choose the subgraph induced by the K-hop neighbors of the query nodes that has the lowest conductance value as the augmented subgraph. It's important to note that this approach allows us to obtain the augmented subgraph adaptively, without requiring a pre-set value for K. To strike a balance between search space and personalization, we set the upper limit for K as 5 as suggested by the experimental results.

Example 1. *Given the data graph as in Figure 1 and node 1, its 1-hop induced subgraph has nodes 1, 2 and 3. Thus, the conductance of the 1-hop induced subgraph is $\frac{1}{7} = 0.143$. Similarly, the conductance of the 2-hop induced subgraph is $\frac{3}{9} = 0.333$. Thus, 1-hop induced subgraph is selected as the augmented subgraph.*

3.3 *CSGphormer* Architecture

Motivation. The graph encoder used in the pre-training phase inputs the augmented subgraph and outputs both the node-level and the community-level representations. A direct way is to use GNNs to learn the node-level representation [17, 54, 56] and use a graph pooling operator to aggregate the node-level representations into community-level representations. However, GNNs possess inherent limitations such as over-smoothing [11] and over-squashing [4] issues, which hinder the full potential of GNNs for representation learning. On the other hand, transformers have been recently introduced for graph analytics due to their effectiveness in addressing over-smoothing and over-squashing issues [12], resulting in a strong representation learning capacity. Many graph transformer models are proposed such as Graphormer [59] and Gophormer [64].

Here, we follow the state-of-the-art graph transformer NAGphormer [12] and propose *CSGphormer*. NAGphormer is the state-of-the-art graph transformer with high efficiency. NAGphormer treats each hop as one token in a sequence and uses a transformer

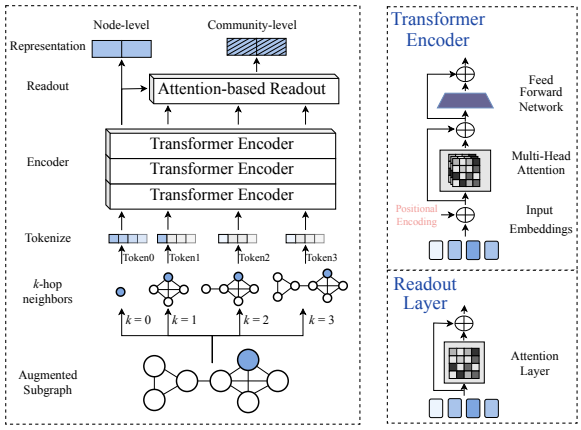


Figure 3: Architecture of *CSGphormer*

to model the correlation among different hops and learn the node representation. *CSGphormer* has three key distinctions from NAGphormer. Firstly, NAGphormer primarily targets supervised learning, while *CSGphormer* focuses on unsupervised learning. Secondly, unlike the fixed value of K used for K-hop neighbors in NAGphormer, we employ the conductance to dynamically determine the value of K. Thirdly, *CSGphormer* outputs both the node-level and the community-level representations while NAGphormer just outputs the node-level representation.

The architecture of *CSGphormer* is illustrated in Figure 3, and the forward propagation of *CSGphormer* is summarized in Algorithm 1. Specifically, we propagate the feature matrix X from 1 to K times to obtain the token sequence $\mathcal{X} = \{{}^0X, {}^1X, \cdots, {}^KX\}$. Here, ${}^0X = X \in \mathbb{R}^{n\times d}$ is the original feature matrix. ${}^kX \in \mathbb{R}^{n\times d}$ is the k-hop neighborhood matrix and is computed by ${}^kX = \hat{A}^kX$. Here, $\hat{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the normalized adjacency matrix. D is the degree matrix of A where $D(i,i) = \sum_{j=1}^{n} A(i,j)$. $\mathcal{X}_v = \{{}^0x_v, \cdots, {}^Kx_v\}$ is the aggregated neighborhood sequence of node v.

Then, the obtained sequence matrix $\mathcal{X}_v \in \mathbb{R}^{(K+1)\times d}$ is sent into a learnable linear projection $W \in \mathbb{R}^{d\times d_m^{(0)}}$: $H_v^{(0)} = \mathcal{X}_vW$, where $H_v^{(0)} \in \mathbb{R}^{(K+1)\times d_m^{(0)}}$. Next, $H_v^{(0)}$ is sent for L layers of the transformer encoder. The transformer encoder contains three important sub-structures, i.e. Positional Encoding, Multi-Head Attention and Feed Forward Network. Specifically, given an input embedding $H_v^{(l)}$ in layer l, the position encoding of the sequence P is added to the input embedding first $H_v^{(l)} = H_v^{(l)} + P$, as in [12].

After that, $H_v^{(l)}$ is sent to the multi-head attention layer (MHA):

$$\begin{aligned} \mathrm{MHA}(H_v^{(l)}) &= \mathrm{Concat}(\mathrm{head}_1, \cdots, \mathrm{head}_h)W^o \\ \text{where } \mathrm{head}_i &= \mathrm{Attention}(H_v^{(l)}W_i^q, H_v^{(l)}W_i^k, H_v^{(l)}W_i^v) \\ \text{and } \mathrm{Attention}(Q,K,V) &= \mathrm{softmax}(\frac{QK^T}{\sqrt{d_m^{(l+1)}}})V \end{aligned} \tag{2}$$

Here, $W_i^q \in \mathbb{R}^{d_m^{(l)}\times d_m^{(l+1)}}$, $W_i^k \in \mathbb{R}^{d_m^{(l)}\times d_m^{(l+1)}}$, $W_i^v \in \mathbb{R}^{d_m^{(l)}\times d_m^{(l+1)}}$, $W^o \in \mathbb{R}^{hd_m^{(l+1)}\times d_m^{(l+1)}}$ are all learnable weights to project $H_v^{(l)}$ into different matrices.

Algorithm 1: Forward Propagation of *CSGphormer*.

Input: center node v, feature matrix X, adjacent matrix A, transformer layers L.
Output: The node representation Z_v^{node} and community-level representation Z_v^{com}.

$\mathcal{X}_v \leftarrow \{{}^0x_v, {}^1x_v, \cdots, {}^Kx_v\}$
$H_v^{(0)} \leftarrow \mathcal{X}_vW$
// L-layers transformer encoder.
for $l = 0, \cdots, L-1$ **do**
  $P \leftarrow$ Position Encoding Construction
  $H_v^{(l)} \leftarrow H_v^{(l)} + P$
  $H_v^{(l+1)} = \mathrm{MHA}(\mathrm{LN}(H_v^{(l)})) + H_v^{(l)}$
  $H_v^{(l+1)} = \mathrm{FFN}(\mathrm{LN}(H_v^{(l+1)})) + H_v^{(l+1)}$
// Readout layer.
$Z_v^{node} \leftarrow {}^0H_v^{(L)}$; $Z_v^{com} \leftarrow$ Zero Tensor
for $k = 1, \cdots, K$ **do**
  $\alpha_k = \frac{\exp(({}^0H_v^{(L)}||{}^kH_v^{(L)})W_a^T)}{\sum_{i=1}^{K}\exp(({}^0H_v^{(L)}||{}^iH_v^{(L)})W_a^T)}$
  $Z_v^{com} \leftarrow Z_v^{com} + \alpha_k\ {}^kH_v^{(L)}$
return Z_v^{node}, Z_v^{com}

The output of the multi-head self-attention layer is added to the original input embedding followed by a layer normalization (LN) [7]. A position-wise feed-forward network (FFN) is applied to each position separately and identically. The FFN consists of two linear layers with a GELU [24] non-linearity:

$$\begin{aligned} H_v^{(l+1)} &= \mathrm{MHA}(\mathrm{LN}(H_v^{(l)})) + H_v^{(l)} \\ H_v^{(l+1)} &= \mathrm{FFN}(\mathrm{LN}(H_v^{(l+1)})) + H_v^{(l+1)} \end{aligned} \tag{3}$$

After the L layers of the transformer encoder, we can obtain the latent representation $H_v^{(L)} \in \mathbb{R}^{(K+1)\times d_m^{(L)}}$ of node v which contain the center node token $Z_v^{node} = {}^0H_v^{(L)} \in \mathbb{R}^{d_m^{(L)}}$ and the latent representation of the neighborhood tokens $\{{}^1H_v^{(L)}, \cdots, {}^KH_v^{(L)}\}$. Then, an attention-based readout function is utilized to weight the neighborhood tokens to obtain the community-level representation:

$$\begin{aligned} \alpha_k &= \frac{\exp(({}^0H_v^{(L)}||{}^kH_v^{(L)})W_a)}{\sum_{i=1}^{K}\exp(({}^0H_v^{(L)}||{}^iH_v^{(L)})W_a)} \\ Z_v^{com} &= \sum_{k=1}^{K}\alpha_k\ {}^kH_v^{(L)}. \end{aligned} \tag{4}$$

Here, $||$ is the concatenation operator [46], and $W_a \in \mathbb{R}^{2d_m^{(L)}\times 1}$ is the learnable weight matrix.

3.4 Training Objectives

Motivation. Intuitively, nodes are dependent on their communities to learn the representation and each node is unique in the graph. We consider the strong correlation between central nodes and their communities to design the contrast-based loss, i.e., personalization loss. Moreover, we design the generation-based loss, i.e., link loss,

based on the idea that *nodes that have a link should be close in the latent space and vice versa.* The generation-based loss would benefit the preservation of local graph topology, and the contrast-based loss would benefit the preservation of the global information and capture the long-distance relationship, as illustrated in [38].

We model the personalization loss by the margin triplet loss [42] to bring the representation of a selected node and its corresponding community closer together and push away the representation of the selected node from the communities of other nodes.

$$\mathcal{L}_p = \frac{1}{|V|^2} \sum_{v \in V} \sum_{u \in V} \left(-\max\left(\sigma(Z_v^{node} Z_v^{com}) - \sigma(Z_v^{node} Z_u^{com}) + \epsilon, 0 \right) \right) \tag{5}$$

where ϵ is the margin value and $\sigma(\cdot)$ is the sigmoid function.

The link loss is formulated as follows to enhance the similarity of neighboring nodes while discriminating non-adjacent nodes.

$$\begin{aligned} \mathcal{L}_k = \frac{1}{|V|^2} \sum_{v \in V} \sum_{u \in V} & -A(u,v)(Z_u^{node} Z_v^{node}) \\ & +(1 - A(u,v))(Z_u^{node} Z_v^{node}) \end{aligned} \tag{6}$$

CSGphormer takes the above two losses into account together. The overall loss function is defined as:

$$\mathcal{L} = \mathcal{L}_p + \alpha \mathcal{L}_k \tag{7}$$

where $\alpha \in [0, 1]$ is the coefficient to balance the two losses. Note that both $\mathcal{L}_p$ and $\mathcal{L}_k$ do not contain label information.

With the loss defined in Equation 7, the overall offline pre-training procedure is summarized in Algorithm 2. It first initializes the parameters in the optimizer and divides all nodes into several batches (lines 1 to 2). Next, it samples all the augmented subgraphs (lines 3 to 4). It trains the model batch by batch. In the training of each batch, it obtains the node-level representation and community-level representation of all nodes in the batch by propagating the *CSGphormer* network (lines 6 to 7). The loss containing both the personalization loss and link loss is computed (lines 8 to 11) and is used to update the parameters in *CSGphormer*(line 12).

Algorithm 2: Offline Pre-training Procedure (One Epoch)

Input: The data graph G, batch size n_{batch}, layer number L, *CSGphormer* $f^{\theta}(\cdot)$, learning rate φ, coefficient α.

1 Initialize optimizer opt_{θ} with learning rate φ
2 Separate V into batches $\{V_b\}$ with the batch size n_{batch}
3 **for** *each* $v \in V$ **do**
4 $G_v \leftarrow$ augmented subgraph sampler
5 **for** $\{V_b\} \in V$ **do**
6 **for** *each* $v \in V_b$ **do**
7 $Z_v^{node}, Z_v^{com} \leftarrow f^{\theta}(v, X(G_v), A(G_v), L)$
8 **for** *each* $u, v \in V_b$ **do**
9 $\mathcal{L}_p = -\max\left(\sigma(Z_v^{node} Z_v^{com}) - \sigma(Z_v^{node} Z_u^{com}) + \epsilon, 0 \right)$
10 $\mathcal{L}_k = -A(u,v)(Z_u^{node} Z_v^{node}) + (1 - A(u,v))(Z_u^{node} Z_v^{node})$
11 $\mathcal{L} += \mathcal{L}_p + \alpha \mathcal{L}_k$
12 Update θ by opt_{θ} with loss $\frac{\mathcal{L}}{|V_b|^2}$.

Algorithm 3: Community Score Computation

Input: The query V_q, graph G, pre-trained network $f^{\theta}(\cdot)$.
Output: The community score S.

1 Initialize $S \leftarrow \{s_v = 0 \text{ for } v \in V\}$
2 **for** $\{v\} \in V$ **do**
3 **for** $\{u\} \in V_q$ **do**
4 $s_v \leftarrow s_v + \frac{\sum_{i=0}^{d_m^{(L)}} f_i^{\theta}(v) f_i^{\theta}(u)}{\sqrt{\sum_{i=0}^{d_m^{(L)}} f_i^{\theta}(v) f_i^{\theta}(v)} \times \sqrt{\sum_{i=0}^{d_m^{(L)}} f_i^{\theta}(u) f_i^{\theta}(u)}}$
5 $s_v \leftarrow \frac{s_v}{|V_q|}$;
6 **return** S

4 ONLINE SEARCH PHASE

With the pre-trained *CSGphormer* in Section 3, we now introduce the details of the online search phase devised for unsupervised CS. We first introduce the community score computation module, and then we introduce the problem of identification with expected score gain (*IESG*), followed by two search algorithms, i.e., *Local Search* and *Global Search* to find promising communities.

4.1 Community Score Computation

Motivation. After pre-training, the community-level information and the graph topology are encoded into the latent representation. Nodes with similar latent representations should have similar community-level information and should be close to each other in the original graph. Therefore, we can compute the community score by evaluating the similarity between the representation of the query and the representations of nodes within the graph. A higher similarity suggests a greater likelihood of the node being part of the resulting community.

The overall community score computation algorithm is shown in Algorithm 3. It inputs the query nodes, graph and the pre-trained graph transformer and outputs the community score w.r.t. the query. The community score is first initialized as all zeros (line 1). It then computes the pairwise similarity between the representation of each node in the query and the representation of each node in the graph (lines 2 to 5). We use the cosine similarity here. More other similarity functions are evaluated in the experiments of Section 6. We ensure that the obtained score is adjusted to fall within the range of 0 to 1 by normalizing it with the cardinality of the query node set (line 5). At last, the community score is returned.

4.2 Identification with Expected Score Gain

Motivation. The community score quantifies the likelihood of a node being included in the community. An ideal community is one where all nodes exhibit high community scores w.r.t. the query nodes. In order to measure the degree to which the score is large, QD-GNN and COCLEP use the label-based threshold, and nodes having a community score larger than the label-based threshold are included in the resulting communities. Under the setting of

unsupervised community search, as analyzed in Section 1, a naive approach like a fixed threshold or a fixed number of nodes would potentially harm the accuracy of community identification. Therefore, we introduce the function of expected score gain (*ESG*), and community scores that maximize the *ESG* are considered high. The *ESG* function, which centers around utilizing community scores, computes the sum of node scores within the community, subtracted by the sum of expected scores under random node selection. A higher *ESG* suggests the potential for a superior community. This idea is inspired by the concept of modularity, a well-established metric of community cohesiveness. The modularity measures the number of edges in the community minus the expected number of edges in the community if the edges are randomly distributed. The higher the modularity, the more cohesive the community [30].

Definition 3. *(Expected Score Gain). Given a graph $G(V, E)$, a community $C = (V_C, E_C)$ and the community score S, the expected score gain of C is defined as:*

$$ESG(S, C, G) = \frac{1}{|V_C|^\tau}\left(\sum_{v \in V_C} s_v - \frac{\sum_{u \in V} s_u}{|V|}|V_C|\right) \tag{8}$$

where $\tau \in [0, 1]$ is a hyper-parameter to control the granularity of the subgraph, and a higher τ value leads to a more fine-grained subgraph.

The first term $\sum_{v \in V_C} s_v$ is the sum of the community score of nodes in the selected community. $\frac{\sum_{u \in V} s_u}{|V|}|V_C|$ is the expected community score where nodes randomly selected have an expected score the same as the average score of the graph. We set $\tau = 0.5$ as suggested by our experiments in Section 6.

Example 2. *Given the data graph as in Figure 1 with the community containing nodes 4 5, 6 and 7, and supposing the community scores are 0.1, 0.2, 0.4, 0.7, 0.9, 0.6, 0.8 for nodes 1 to 7 respectively, the expected score gain can be calculated as $\frac{1}{4^{0.5}}(3.0 - \frac{3.7}{7} \times 4) = 0.443$.*

Besides the preference for nodes in the resulting community to exhibit high community scores, we aim to search for a cohesive subgraph, and connected components are more cohesive than unconnected subgraphs. Furthermore, the identified community is query-dependent and should therefore include the query nodes. Based on these considerations, we formally define the problem of identification with expected score gain (*IESG*).

Definition 4. *(Identification with Expected Score Gain). Given a graph $G(V, E)$, the query V_q, the community score S and a profit function $ESG(\cdot)$, IESG aims to select a community C of G, such that:*

(1) V_C contains nodes in V_q, and C is connected;

(2) $ESG(S, C, G)$ is maximized among all feasible choices for C.

We give the hardness of *IESG* in Lemma 1, and the proof can be found in Section 5.

Lemma 1. *The problem of IESG is NP-hard.*

4.3 Heuristic Algorithms

As *IESG* is NP-hard which means it cannot be solved in polynomial time, heuristic algorithms are proposed to effectively and efficiently find promising communities without labels. A direct approach (*Local Search*) starts from the query nodes and greedily incorporates the node which is of the highest community score and is in the neighborhood of the intermediate subgraph. The subgraph with the largest *ESG* encountered during the search process is returned. In this way, we do not need the label-based threshold, and thus *Local Search* does not require labels for community identification.

Algorithm 4: *Local Search* Algirithm

Input: The community score S, graph G and query V_q.
Output: The identified community $\tilde{C}_q$.

1 $\tilde{C}_q, Q \leftarrow V_q$; $max\_esg \leftarrow -inf$
2 **while** $|Q| < |V|$ **do**
3 $u \leftarrow \text{argmax}_{v \in \zeta\overline{Q}} s_v$
4 $Q = Q \cup u$;
5 **if** $ESG(S, \tilde{C}_q \cup \{u\}, G) > max\_esg$ **then**
6 $max\_esg \leftarrow \text{ESG}(S, \tilde{C}_q \cup \{u\}, G)$
7 $\tilde{C}_q = \tilde{C}_q \cup \{u\}$
8 **else**
9 Terminate
10 **return** $\tilde{C}_q$

The overall algorithm of *Local Search* is summarized in Algorithm 4. It inputs the community score, graph and query nodes, and outputs the identified community. We use Q to store the nodes that have been traversed and designate the maximum expected score gain value as negative infinity during initialization (line 1). The algorithm terminates until all the nodes have been traversed or early stops when there are no promising candidates (lines 2 to 9). In each loop, we first select the node with the highest community score that has not been traversed and is located at the boundary of the node set that has been traversed (line 3). $\zeta\overline{Q} = \{v \in \overline{Q} | \exists i \in N^{(1)}(v) \cap Q\}$ is the boundary of $\overline{Q}$ and $\overline{Q} = V \backslash Q$. If the selected nodes can increase the expected score gain of the previous intermediate subgraph, we incorporate it into the community. Otherwise, the algorithm early stops (lines 5 to 9). At last, the community returns.

Motivations for *Global Search*. The motivations behind *Global Search* can be outlined in three aspects. Firstly, by incorporating link loss, the learned representation effectively preserves graph topology information. Nodes with high similarities to the query nodes exhibit high community scores and are likely connected to the query nodes. This suggests that optimizing the expected score gain first can also provide a favorable priority for connected nodes. Secondly, the time complexity of *Local Search* is $O(|V|^2 \log(|V|))$, as detailed in Section 5. This quadratic logarithmic time complexity presents challenges when applying *Local Search* to large graphs. Thirdly, as detailed in Lemma 2, the expected score gain of the first p nodes in the queue sorted by community score initially increases and then decreases with the increase of p.

Lemma 2. *Given sorted scores $\hat{S}$ from large to small, size $p > 0, C_p = \{v_i | \hat{s}_i \geq \hat{s}_p\}, S_p = \{\hat{s}_i | i \leq p\}$, assuming $\sum_{\hat{s}_i \in S_p} \hat{s}_i = \mu |S_p|^{\sigma\tau}$ where μ, σ are hyperparmeters and $\mu, \sigma > 0$ and, $\sigma\tau < 1$, $ESG(\hat{S}, C_p, G)$ first increases and then decreases as p increases.*

As S is learned, there lacks a functional expression for S. Since S_p gets the first p scores from a sorted queue $\hat{S}$, we assume that the sum of S_p exhibits a decreasing growth rate as size increases, i.e.

Algorithm 5: *Global Search* Algorithm

Input: The community score S, graph G and query V_q.
Output: The identified community $\tilde{C}_q$.

1 $\tilde{C}_q \leftarrow V_q$;$t_s = 0$; $t_e = |S|$
2 $\hat{S} \leftarrow$ sort S from large to small
3 **while** $t_s < t_e$ **do**
4 $\quad C_{mid} = \{v_i | \hat{s}_i \geq \hat{s}_{\frac{t_s+t_e}{2}}\}$
5 $\quad C_{left} = \{v_i | \hat{s}_i \geq \hat{s}_{\frac{t_s+t_e}{2}-1}\}$
6 $\quad$ **if** $ESG(\hat{S}, C_{mid}, G) > ESG(\hat{S}, C_{left}, G)$ **then**
7 $\quad\quad t_s \leftarrow \frac{t_s+t_e}{2}$
8 $\quad$ **else**
9 $\quad\quad t_e \leftarrow \frac{t_s+t_e}{2}$
10 **return** $\tilde{C}_q = \tilde{C}_q \cup \{v_i | \hat{s}_i \geq \hat{s}_{t_e}\}$

$\sum_{\hat{s}_i \in S_p} \hat{s}_i = \mu |S_p|^{\sigma\tau}$. Lemma 2 provides theoretical support for the binary search optimization, and the proof is in Section 5.

Given these motivations and the theoretical foundation, we introduce *Global Search* in Algorithm 5, which prioritizes candidates that enhance the *ESG* from a global perspective. It inputs the learned community score, graph and query nodes, and outputs the found community. The community is initialed as the query nodes, and we designate the search start point t_s as 0 and the search end point t_e as the maximum number of nodes during initialization (line 1). The algorithm sorts all the community scores (line 2). It loops until the start point equals the end point (lines 3 to 9). In each loop, it selects the candidate community C_{mid} from 0 to $\frac{t_s+t_e}{2}$ and selects the candidate community C_{left} from 0 to $\frac{t_s+t_e}{2}-1$ which lies at the left of C_{mid} (lines 4 and 5). If the *ESG* of C_{mid} is larger than that of C_{left} which means there may be promising candidates in the index range of $[\frac{t_s+t_e}{2}, t_e]$, we set the new start point as $\frac{t_s+t_e}{2}$ (lines 6 and 7). Otherwise, the end point is set as $\frac{t_s+t_e}{2}$ (lines 8 and 9). At last, the identified community returns (line 10).

While the community found by *Global Search* may not always be guaranteed to be connected, it prioritizes connected nodes since link loss is used for pre-training, as shown in our earlier motivation. Additionally, *Global Search* has a time complexity of $O(2\times|V|\log(|V|))$, as analyzed in Section 5, making it well-suited for large datasets.

Lemma 3. *Global Search runs at most* $log_2(|V|)$ *iterations.*

The proof of Lemma 3 is immediate as *Global Search* reduces the search space by half each iteration.

5 ANALYSIS

5.1 Theoretical Analysis

Proof of Lemma 1. We reduce the problem of *IESG* from the set cover problem which is a well-known NP-hard problem [6]. The following gives the formal definition of the set cover problem.

Definition 5 (Set Cover Problem). *Given a finite set* $M = \{m_1, m_2, \ldots, m_{|M|}\}$ *and a collection* $\mathcal{N} = \{N_1, N_2, \ldots, N_{|\mathcal{N}|}\}$ *of subsets of* M*, the Set Cover Problem aims to find a minimum-size subcollection* $\mathcal{N}_{opt}$ *such that the union of all sets in* $\mathcal{N}_{opt}$ *covers all elements in* M*, i.e.,* $\bigcup_{N_i \in \mathcal{N}_{opt}} N_i = M$.

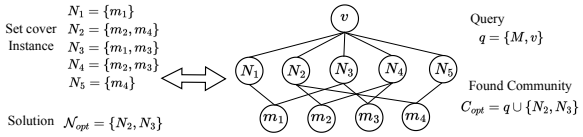


Figure 4: Graph construction for the set cover problem

We then follow [55] to construct a graph. We create one node m_i for each element in $m_i \in M$ and a node N_i for each set in $N_i \in \mathcal{N}$. One edge is added between m_i and N_i if $m_i \in N_i$. Another node v is added with edges between v and each set in $\mathcal{N}$. Figure 4 presents an example of graph construction.

We use $w(C) = \sum_{v \in C} s_v$ to denote the sum of scores in C and set $\tau = 1$ in $ESG(S, C, G)$. Therefore, we have $ESG(S, C, G) = \frac{w(C)}{|V_C|} - \frac{w(G)}{|V|}$. We omit the second term as it is a fixed value for communities in one graph. We use $g(\cdot)$ to denote the simplified function, i.e., $g(S, C, G) = \frac{w(C)}{|V_C|}$ and assume the community score of nodes in M and $\{v\}$ is $\frac{1}{|M|+1}$, and the community score of nodes in $\mathcal{N}$ is $\frac{1}{|M||\mathcal{N}|}$.

We set the query nodes as $q = M \cup \{v\}$. To make the return community connected, we need to select some nodes in $\mathcal{N}$. Let $\mathcal{N}_* \subseteq \mathcal{N}$ be a feasible solution for the set cover problem given the element collection M and the set collection $\mathcal{N}$. Let $C = q \cup \mathcal{N}_* = M \cup \{v\} \cup N_*$. Then, C is connected, contains the query nodes and has the expected weight gain $g(S, q \cup \mathcal{N}_*, G) = \frac{w(C)}{|V_C|} = \frac{1+\frac{|\mathcal{N}_*|}{|M||\mathcal{N}|}}{|M|+1+|\mathcal{N}_*|}$. The derivative of $g(S, q \cup \mathcal{N}_*, G)$ is $g'(S, q \cup \mathcal{N}_*, G) = (\frac{|M||\mathcal{N}|+|\mathcal{N}_*|}{|M|^2|\mathcal{N}|+|M||\mathcal{N}|+|M||\mathcal{N}||\mathcal{N}_*|})' = \frac{1}{|M|^2|\mathcal{N}|+|M||\mathcal{N}|+|M||\mathcal{N}||\mathcal{N}_*|} - \frac{(|M||\mathcal{N}|+|\mathcal{N}_*|)|M||\mathcal{N}|}{(|M|^2|\mathcal{N}|+|M||\mathcal{N}|+|M||\mathcal{N}||\mathcal{N}_*|)^2}$. Thus, $g'(S, q \cup \mathcal{N}_*, G) = \frac{|M||\mathcal{N}|(|M|+1-|M||\mathcal{N}|)}{(|M|^2|\mathcal{N}|+|M||\mathcal{N}|+|M||\mathcal{N}||\mathcal{N}_*|)^2}$. As $|M|, |\mathcal{N}| > 1$, then $|M| + 1 - |M||\mathcal{N}| < 0$ and $g'(S, q \cup \mathcal{N}_*, G) < 0$. Thus $g(S, q \cup \mathcal{N}_*, G)$ is monotonically decreasing with regard to $|\mathcal{N}_*|$. Since $|\mathcal{N}_*| \geq |\mathcal{N}_{opt}|$, the subgraph $q \cup \mathcal{N}_{opt}$ contains the query nodes, is connected, and has the highest expected score gain. Note that $\mathcal{N}_{opt}$ is the optimal solution to the set cover problem. Therefore, we can reduce the problem of *IESG* from the set cover problem.

Given a community C, $g(S, C, G)$ is monotonically decreasing as demonstrated above. With an optimal community C_{opt}, its *ESG* is highest. Hence, $|C_{opt} \backslash q|$ is the minimum among feasible solutions. According to the graph constructed above, $C_{opt} \backslash q$ is connected to all elements in M and has the minimum size. Thus, the corresponding sets of $C_{opt} \backslash q$ is the optimal solution for the set cover. Moreover, according to the map, the time complexity of the reduction is linear.

Proof of Lemma 2. $ESG(\hat{S}, C_p, G) = \frac{\mu|S_p|^{\sigma\tau}}{|V_{C_p}|^{\tau}} - \frac{|V_{C_p}|}{|V_{C_p}|^{\tau}}\lambda = \frac{\mu|V_{C_p}|^{\sigma\tau}}{|V_{C_p}|^{\tau}} - \frac{|V_{C_p}|}{|V_{C_p}|^{\tau}}\lambda$ where $\lambda = \frac{\sum_{u \in V} s_u}{|V|}$. Then the derivative is $ESG'(\hat{S}, C_p, G) = \mu(\sigma\tau - \tau)|V_{C_p}|^{\sigma\tau-\tau-1} - (1-\tau)\lambda|V_{C_p}|^{-\tau}$. By setting $ESG'(\hat{S}, C_p, G) = 0$, we can prove that $ESG(\hat{S}, C_p, G)$ increases in $[0, (\frac{(1-\tau)\lambda}{\mu(\sigma\tau-\tau)})^{\frac{1}{\sigma\tau-1}}]$ and decreases in $[(\frac{(1-\tau)\lambda}{\mu(\sigma\tau-\tau)})^{\frac{1}{\sigma\tau-1}}, |V|]$.

5.2 Time Complexity And Extension

Time complexity of pre-training. The time complexity of the projection of three matrices is $O(3\times(K+1)\times d^2)$. The dot product of

Table 3: Statistics of the datasets

Datasets	$\|V\|$	$\|E\|$	$\|C\|$	d
Texas	183	325	5	1,703
Cornell	183	298	5	1,703
Wisconsin	251	515	5	1,703
Cora	2,708	10,556	7	1,433
Citeseer	3,327	9,104	6	3,703
Photo	7,650	238,162	8	745
DBLP	17,716	105,734	4	1,639
CoCS	18,333	163,788	15	6,805
Physics	34,493	495,924	5	8,415
Reddit	232,965	114,615,892	41	602

query and key takes $O((K+1)^2 \times d)$ and the dot product of attention and value also takes $O((K+1)^2 \times d)$. Therefore, the self-attention has a time complexity of $O(3 \times (K+1) \times d^2 + 2 \times (K+1)^2 \times d)$. There are $|V|$ nodes in the graph and the number of transformer encoders is L, thus the time complexity of *CSGphormer* is $O(L \times |V| \times (3 \times (K+1) \times d^2 + 2 \times (K+1)^2 \times d))$. It is trained for t epochs, therefore, the total time complexity of the offline pre-training is $O(t \times L \times |V| \times (3 \times (K+1) \times d^2 + 2 \times (K+1)^2 \times d))$.

Time complexity of community score computation. The time complexity of pair-wise similarity computation is $O(d_m)$ where d_m is the dimension of the latent representation. We need to compute for $|V_q| \times |V|$. Therefore, the overall time complexity of community score computation is $O(|V_q| \times |V| \times d_m)$.

Time complexity of *Local Search*. The computation of nodes with the highest score needs $O(|V|log|V|)$ times. We need to compute at most by $|V|$ times. Thus, the overall time complexity of *Local Search* is $O(|V|^2 log|V|)$.

Time complexity of *Global Search*. Sorting needs $O(|V|log|V|)$ [1, 14]. It iterates at most $log|V|$ iterations, and each iteration needs $O(|V|)$ operations. Therefore, the total time complexity of *Global Search* is $O(2 \times |V|log|V|)$.

Extension and future works. *TransZero* is designed for general CS without labels, and it can be extended to support other related settings of CS. With a specific score computation module to hand additional query input, *TransZero* can be extended to support attributed CS [31, 47, 51]. With a specific pretraining model, *TransZero* can be extended to support other types of graph, e.g., temporal graph [23, 61]. We leave these promising fields in the future works.

6 EXPERIMENTAL EVALUATION

6.1 Dataset Description

We use 10 public datasets from Pytorch Geometric [20] following existing work [28] to comprehensively evaluate the performance. The statistics information of the datasets is summarized in Table 3. Datasets are characterized by varying numbers of nodes (i.e., $|V|$), numbers of edges (i.e., $|E|$), numbers of communities (i.e., $|C|$) and dimensionalities of features (i.e., d).

6.2 Experimental Setup

Baselines: We focus on the general CS task. Following [28, 32], we use the existing learning-based models including 1) QD-GNN [28], which is a supervised state-of-the-art for CS; 2) COCLEP [32], which is a semi-supervised state-of-the-art for CS, and the traditional CS methods including 3) CST [15] that uses k-core to model the community; 4) EquiTruss [2] that uses k-truss to model the community; 5) MkECS [3] that uses k-ECC to model the community; and 6) CTC [27] that aims to find the closest truss community.

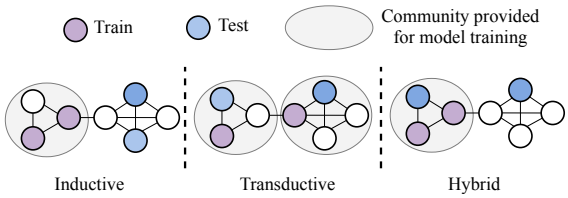


Figure 5: Illustration for the query generation settings

Query generation: We use the following three generation mechanisms to generate queries for training, validation and testing.

- **Inductive Setting.** We randomly partition all ground-truth communities into two groups including training communities and testing communities with a ratio of about 1: 1. And we generate training and validation queries from the training communities and generate test queries from the testing communities. This setting aims to test the ability to predict unseen communities.
- **Transductive Setting.** We generate all the queries randomly from all the ground-truth communities.
- **Hybrid Setting.** We randomly divide ground-truth communities into training and testing groups (∼1:1 ratio). Training and validation queries are generated from the training communities, while test queries are generated from all ground-truth communities. This setting closely simulates real-world scenarios by training on a subset of known ground-truth communities and evaluating across all the ground-truth communities.

An illustration of the above three settings is shown in Figure 5. Note that QD-GNN is evaluated in a transductive manner in the original paper, and COCLEP is evaluated in a transductive (*resp.* inductive) manner when the number of ground-truth communities is small (*resp.* large) as in the original paper. Consistent with [28], the number of training queries, validation queries and testing queries are 150, 100 and 100, respectively. Following [28], we randomly select 1 to 3 nodes from the ground-truth community as the query nodes. As in the original paper of COCLEP [32], we generate 3 positive samples beside the query node for COCLEP.

Metrics: In this paper, we mainly focus on F1-score [40] that is commonly used by existing works [28, 32] to evaluate the quality of the found community. Besides the F1-score, we also utilize Normalized Mutual Information (NMI) [16] and Jaccard similarity (JAC) [62] aligned with COCLEP [32] for evaluation. We follow the calculation of F1-score used in [28]. For all the F1-score, NMI and Jaccard, a higher value indicates a better found community.

Implementation Details: We run *TransZero* for 100 epoches with early stopping. The maximum number of hops used in the augmented subgraph sampler is 5. The value of τ is set as 0.5 for all datasets. The value of α is set as 0.1 for all datasets. The batch size is set as the number of nodes in the graph or 4000 if it runs out of memory. We limit our search to a maximum of 50% of the total

Table 4: F1-score results under different settings

Settings	Models	Texas	Cornell	Wisconsin	Cora	Citeseer	Photo	DBLP	CoCS	Physics	Reddit	Average +/-
Inductive	CST	0.1986	0.1975	0.2251	0.2111	0.1423	0.2019	0.2854	0.1252	0.2276	0.1463	-27.12%
	EquiTruss	0.3120	0.3168	0.3079	0.2384	0.2240	0.2166	0.3252	0.1225	0.2471	0.2163	-21.46%
	M*k*ECS	0.3581	0.3177	0.3404	0.2364	0.2015	0.1975	0.2768	0.1152	0.2193	0.2068	-22.03%
	CTC	0.3211	0.3482	0.3327	0.2558	0.2418	0.2626	0.3417	0.1059	0.2511	0.2431	-19.69%
	QD-GNN	0.0821	0.0669	0.0683	0.0322	0.0536	0.0018	0.0372	0.0145	OOM	OOM	-41.50%
	COCLEP	0.4044	0.2960	0.1804	0.3094	0.3058	0.4413	0.3066	0.4253	0.3389	0.2696	-13.95%
	TransZero-LS	0.1801	0.1583	0.2074	0.5467	0.3906	0.5725	**0.4407**	0.4292	0.5075	**0.4879**	-7.52%
	TransZero-GS	**0.4283**	**0.3716**	**0.3755**	**0.5764**	**0.4535**	**0.6018**	0.4326	**0.4374**	**0.5113**	0.4848	-
Transductive	QD-GNN	0.6703	0.8408	0.6247	0.5062	0.4726	0.2205	0.4918	0.6356	OOM	OOM	+9.81%
	COCLEP	0.4020	0.3167	0.3206	0.3685	0.3331	0.5060	0.3763	0.3549	0.4388	0.3270	-9.29%
Hybrid	QD-GNN	0.3852	0.3644	0.5956	0.4789	0.4097	0.0833	0.3902	0.4969	OOM	OOM	-5.91%
	COCLEP	0.3883	0.3313	0.2938	0.3615	0.3067	0.4388	0.3733	0.4027	0.4693	0.3071	-10.01%

∗ CST, EquiTruss, M*k*ECS, CTC and *TransZero* have consistent results under three settings as they are label-free. *TransZero* with *Local Search* is denoted as *TransZero*-LS, and *TransZero* with *Global Search* is denoted as *TransZero*-GS. OOM indicates out-of-memory. The last column presents the average margin compared to *TransZero*-GS.

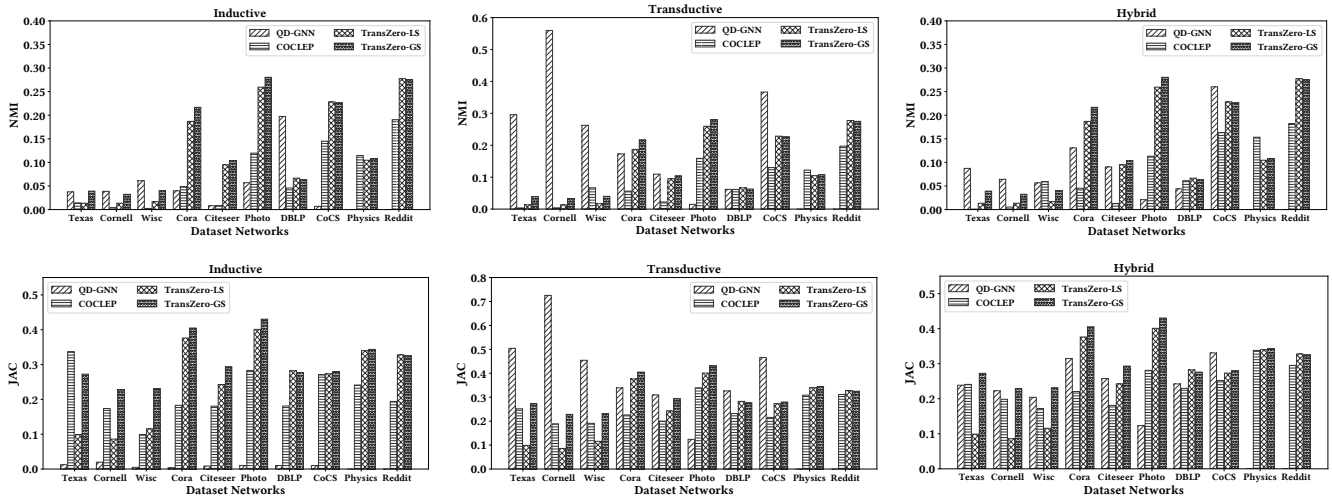


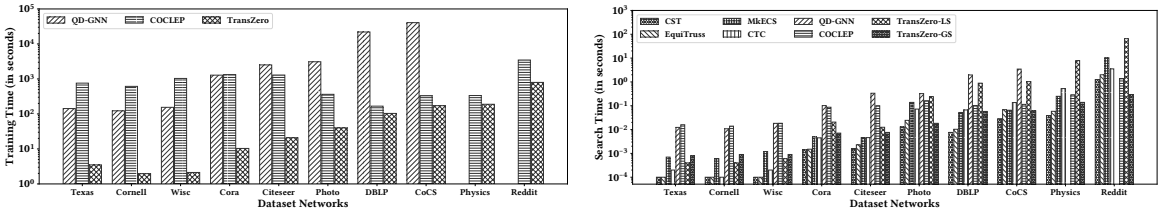
Figure 6: NMI and JAC results under different settings

nodes and do not exceed 10,000 nodes. The number of transformer layer is the same with [12]. *Global Search* is used as the default online search method. The hyper-parameters of QD-GNN and COCLEP are the same as in their original paper. For the new graphs, the number of clusters in COCLEP is set to 10 (*resp.* 2) for large (*resp.* small) graphs. Experiments are conducted on a server with Intel(R) Xeon(R) Gold 6342 CPU and Nvidia RTX 4090 (GPU).

6.3 Effectiveness Evaluation

Exp-1: F1-score results. We first present the F1-score results across three settings in Table 4. Note that traditional CS methods and *TransZero* have consistent results across three settings as they are label-free, and we solely present their performance in the inductive setting to avoid redundancy. *TransZero* with *Local Search* (*resp. Global Search*) is denoted as *TransZero*-LS (*resp. TransZero*-GS). OOM indicates out-of-memory. Among the traditional methods, both EquiTruss and CTC demonstrate competitive performances, and *TransZero*-GS outperforms EquiTruss by 21.46% and surpasses CTC by 19.69%. Among the learning-based models, the performance of QD-GNN varies significantly across settings. QD-GNN needs all nodes in the ground-truth communities for training and thus good at the transductive setting as it can memorize all the communities. However, it is hard to generalize its performance to settings with unseen communities (i.e., inductive setting and hybrid setting). In the inductive setting, *TransZero*-GS significantly outperforms QD-GNN and COCLEP with an average F1-score enhancement of 41.50% and 13.95%, respectively. In the hybrid setting, *TransZero*-GS outperforms QD-GNN and COCLEP by an average F1-score of 5.91% and 10.01%, respectively.

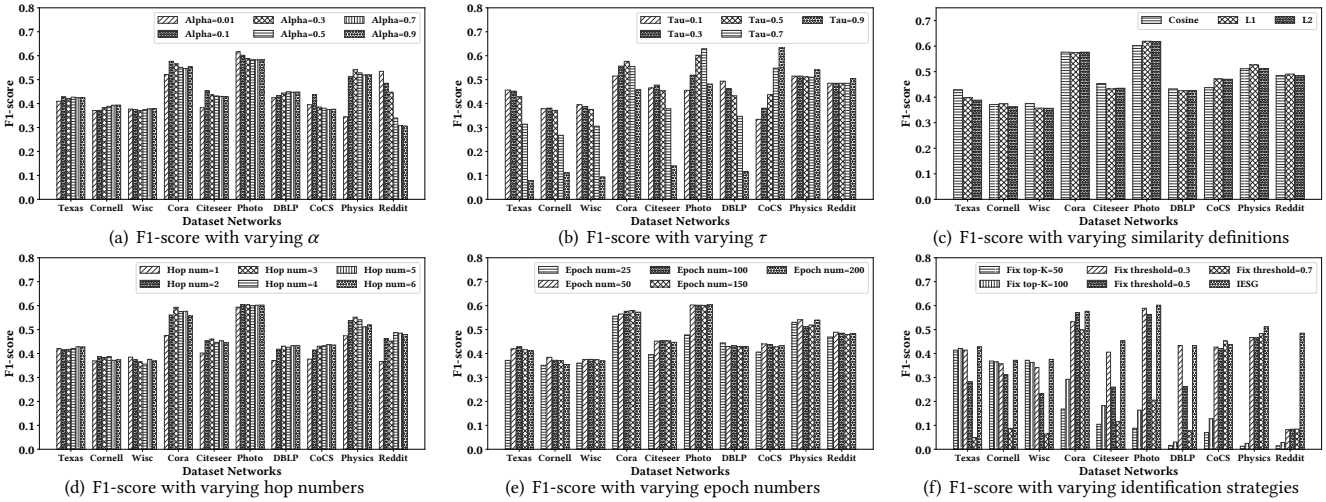
Exp-2: NMI and JAC results under different settings. In this part, we use the NMI and JAC to measure the learning-based methods as their high performance demonstrated in Exp-1. The results



(a) Efficiency results of the training phase

(b) Efficiency results of the search phase

Figure 7: Efficiency results



(a) F1-score with varying α

(b) F1-score with varying τ

(c) F1-score with varying similarity definitions

(d) F1-score with varying hop numbers

(e) F1-score with varying epoch numbers

(f) F1-score with varying identification strategies

Figure 8: Hyper-parameter analysis results

are presented in Figure 6. The figure shows that QD-GNN still has the best performance under the transductive setting, and the performance is hard to generalize to the inductive setting and the hybrid setting. In terms of NMI, *TransZero*-GS outperforms QD-GNN with an average enhancement of 9.75% and surpasses COCLEP by 5.86% under the hybrid setting. In terms of JAC, *TransZero*-GS outperforms QD-GNN by an average of 3.27% and surpasses COCLEP by 6.75% under the hybrid setting.

6.4 Efficiency Evaluation

Exp-3: Efficiency evaluation. In Figure 7, we report the efficiency results of both the training phase and the search phase. Note that there is no training phase for the traditional CS methods, and thus we only report their efficiency of the search phase. In terms of the training phase, *TransZero* significantly outperforms the existing learning-based methods. It achieves an average speedup of 118.22× and up to 235.83× in dataset CoCS compared to QD-GNN. When compared to COCLEP, *TransZero* achieves an average speedup of 122.39× and up to 486.07× in Wisconsin. Regarding the search phase, *TransZero*-GS achieves an average speedup of 26.77× and reaches up to 56.48× in the CoCS dataset when compared to QD-GNN. When compared to COCLEP, *TransZero* achieves an average speedup of 10.02× and up to 20.41× in Wisconsin. Compared to the traditional CS methods, *TransZero*-GS also shows a competitive performance, particularly on large datasets.

6.5 Hyper-parameter Analysis

Exp-4: Varying α. In Figure 8(a), we evaluate *TransZero* with varying values of α. Note that α is utilized to balance the personalization loss and the link loss in Equation 7. We set the value equal to 0.01, 0.1, 0.3, 0.5, 0.7 and 0.9 respectively. The figure shows that the value of α has a different impact on different datasets. In Cornell and DBLP, the performance of *TransZero* improves with the increase of α. On the datasets like Reddit and Photo, the performance decreases with the increase of α. In general, α with a value of 0.1 could effectively balance the two losses and achieve a good performance.

Exp-5: Varying τ. In Figure 8(b), we evaluate *TransZero* across various values of τ. τ is utilized to regulate the granularity of the subgraph, as defined in Definition3, with higher τ resulting in a more fine-grained subgraph. We consider τ values of 0.1, 0.3, 0.5, 0.7, and 0.9. The results show that the performance of *TransZero* experiences a decrease with the increase of τ for smaller datasets such as Texas. Conversely, for datasets like CoCS and Reddit, the performance of *TransZero* improves with higher τ values. In general, τ with a value of 0.5 consistently demonstrates excellent performance.

Exp-6: Varying similarity definitions. In this part, we evaluate *TransZero* using different similarity definitions. We replace line 4 in Algorithm 3 with L1-similarity and L2-similarity [48]. The results are in Figure 8(c). The results show that the performance using different similarities achieves a close performance. The results validate the robustness of *TransZero* to different similarity definitions.

Table 5: Ablation study

Models	Texas	Cornell	Wisconsin	Cora	Citeseer	Photo	DBLP	CoCS	Physics	Reddit	Average +/−
Full model	0.4283	0.3716	0.3755	0.5764	0.4535	0.6018	0.4326	0.4374	0.5113	0.4848	-
w/o $\mathcal{L}_p$	0.4215	0.3749	0.3773	0.5462	0.4259	0.5716	0.4501	0.3502	0.5183	0.2981	-3.19%
w/o $\mathcal{L}_k$	0.3894	0.3576	0.3579	0.4203	0.3044	0.6116	0.4087	0.4532	0.3506	0.5076	-5.12%
w/o Conductance Aug	0.4212	0.3692	0.3848	0.4755	0.4019	0.5935	0.3708	0.3766	0.4738	0.4167	-3.89%
w/o *CSGphormer*	0.3317	0.2421	0.2169	0.4048	0.2780	0.4473	0.2708	0.3074	0.3435	0.3649	-14.65%

Exp-7: Varying hop numbers. In Figure 8(d), we evaluate *TransZero* across varying numbers of hops, which are employed in the augmented subgraph sampler. We consider the number of hops as 1, 2, 3, 4, 5, and 6, respectively. The results show that 1 hop information proves sufficient for small graphs such as Wisconsin. Conversely, for medium and large graphs, a larger number of hops is necessary. For instance, in the case of Reddit, *TransZero* achieves optimal performance with 4 hops and 5 hops. In summary, 5 hops are generally sufficient for *TransZero* to achieve optimal performance.

Exp-8: Varying epoch numbers. In Figure 8(e), we present the performance of *TransZero* across varying numbers of training epochs during the pre-training phase. We consider the number of epochs as 25, 50, 100, 150, and 200, respectively. As depicted in the figure, the performance exhibits similarity after 50 epochs, suggesting that the model has reached convergence within the first 50 epochs, and 100 epochs prove to be sufficient for effective model training.

Exp-9: Varying identification strategies. In Figure 8(f), we present the results of various identification strategies discussed in Section 1. We compare the fixed number strategy and the fixed threshold strategy. The fixed number is set as 50 and 100, and the fixed threshold is set as 0.3, 0.5, and 0.7, respectively. The figure indicates that neither the fixed-number-based strategy nor the fixed-threshold-based strategy generalizes well across all evaluated datasets. In contrast, our proposed *IESG* consistently demonstrates strong performance. These findings highlight the effectiveness of our proposed *IESG*.

6.6 Ablation Study

Exp-10: Ablation study. In this section, we investigate the effectiveness of components employed in *TransZero*, including the personalized loss $\mathcal{L}_p$, the link loss $\mathcal{L}_k$, the conductance-based subgraph sampler and the *CSGphormer*. The results are presented in Table 5. Regarding the personalization loss, its impact becomes apparent in scenarios involving medium and large graphs. Especially, when applied to the Reddit dataset, $\mathcal{L}_p$ exhibits a remarkable enhancement in the F1-score, enabling an F1-score increase of 18.67. In general, it delivers an average F1-score improvement of 3.19%. In terms of the link loss, it delivers an average F1-score improvement of 5.12%. For the Physics dataset, it can enhance the F1-score by 16.07%. To evaluate the effectiveness of our proposed conductance-based augmented subgraph sampler, we replace it with the sampler in COCLEP [32], the previous state-of-the-art CS model. The results show that the conductance-based sampler can enhance the F1-score with an average of 3.89%. Furthermore, to evaluate the effectiveness of the *CSGphormer* architecture, we replace it with the Subg-Con model [29] which is a classical contrast-based self-supervised approach to pre-train the node representation. The results show that our *CSGphormer* significantly improves the F1-score, with an average improvement of 14.65%. These results collectively demonstrate the effectiveness of the modules designed in *TransZero*.

7 RELATED WORK

Existing methods for CS can be classified into two categories: traditional CS methods and learning-based techniques. Traditional CS methods aim to identify a cohesively connected subgraph within a given graph that contains specific query nodes and satisfies given constraints. They model the community by pre-defined cohesive subgraph models such as k-core [15, 44], k-truss [2, 26] and k-edge connected component (k-ECC) [10, 25]. Nevertheless, these approaches encounter a limitation known as *structure inflexibility*. Recently, there is a growing interest towards learning-based CS methods. ICS-GNN [21] proposes a lightweight interactive community search model via graph neural network. QD-GNN and AQD-GNN are proposed in [28] for CS and attributed community search in a supervised manner. COCLEP is proposed in [32] for CS in a semi-supervised manner that only needs a few labels of nodes rather than all labels of nodes in the ground-truth community. Meta-learning is used for CS in [18]. One parallel work is designed in [22]. Although it does not use the ground-truth community information, it uses the K-core information as labels for pretraining and predicts the K-core community. Moreover, it selects fixed-size nodes as the prediction. Another parallel work [50] uses graph transformer for CS. However, it is designed for supervised learning.

8 CONCLUSION

In this paper, we study the problem of general community search and propose an efficient and learning-based community search framework *TransZero* that runs without using labels. It contains the offline pre-training phase and the online search phase. In the offline pre-training phase, we pre-train *CSGphormer*. We compute the community score without using labels by measuring the similarity of the learned representations. In the online search phase, we model the task of community identification as the task of *IESG*. We prove that the problem of *IESG* is NP-hard, and propose two heuristic algorithms including the *Local Search* and *Global Search* to effectively and efficiently find communities. Experiments over 10 public datasets highlight the effectiveness and efficiency of *TransZero*.

ACKNOWLEDGMENTS

Kai Wang is the corresponding author. Kai Wang is supported by NSFC 62302294 and NSFC U2241211. Xuemin Lin is supported by NSFC U20B2046 and NSFC U2241211. Wenjie Zhang is supported by ARC DP230101445 and ARC FT210100303.

REFERENCES

[1] Miklós Ajtai, János Komlós, and Endre Szemerédi. 1983. An 0 (n log n) sorting network. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*. 1–9.

[2] Esra Akbas and Peixiang Zhao. 2017. Truss-based community search: a truss-equivalence based indexing approach. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1298–1309.

[3] Takuya Akiba, Yoichi Iwata, and Yuichi Yoshida. 2013. Linear-time enumeration of maximal K-edge-connected subgraphs in large networks by random contraction. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, Qi He, Arun Iyengar, Wolfgang Nejdl, Jian Pei, and Rajeev Rastogi (Eds.). ACM, 909–918. https://doi.org/10.1145/2505515.2505751

[4] Uri Alon and Eran Yahav. 2021. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *International Conference on Learning Representations*. https://openreview.net/forum?id=i80OPhOCVH2

[5] Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE, 475–486.

[6] Sanjeev Arora. 1998. The approximability of NP-hard problems. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. 337–348.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[8] Wendong Bi, Bingbing Xu, Xiaoqian Sun, Zidong Wang, Huawei Shen, and Xueqi Cheng. 2022. Company-as-tribe: Company financial risk assessment on tribe-style graph with hierarchical graph neural networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2712–2720.

[9] Ciro Cattuto, Marco Quaggiotto, André Panisson, and Alex Averbuch. 2013. Time-varying social networks in a graph database: a Neo4j use case. In *First international workshop on graph data management experiences and systems*. 1–6.

[10] Lijun Chang, Xuemin Lin, Lu Qin, Jeffrey Xu Yu, and Wenjie Zhang. 2015. Index-based optimal algorithms for computing steiner components with maximum connectivity. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 459–474.

[11] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 3438–3445.

[12] Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. 2022. NAGphormer: A tokenized graph transformer for node classification in large graphs. In *The Eleventh International Conference on Learning Representations*.

[13] Wei Chen, Manrui Jiang, Wei-Guo Zhang, and Zhensong Chen. 2021. A novel graph convolutional feature based convolutional neural network for stock trend prediction. *Information Sciences* 556 (2021), 67–94.

[14] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2022. *Introduction to algorithms*. MIT press.

[15] Wanyun Cui, Yanghua Xiao, Haixun Wang, and Wei Wang. 2014. Local search of communities in large graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 991–1002.

[16] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. 2005. Comparing community structure identification. *Journal of statistical mechanics: Theory and experiment* 2005, 09 (2005), P09008.

[17] Peng Fang, Arijit Khan, Siqiang Luo, Fang Wang, Dan Feng, Zhenli Li, Wei Yin, and Yuchao Cao. 2023. Distributed Graph Embedding with Information-Oriented Random Walks. *Proc. VLDB Endow.* 16, 7 (2023), 1643–1656. https://doi.org/10.14778/3587136.3587140

[18] Shuheng Fang, Kangfei Zhao, Guanghua Li, and Jeffrey Xu Yu. 2023. Community Search: A Meta-Learning Approach. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 2358–2371. https://doi.org/10.1109/ICDE55515.2023.00182

[19] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. 2020. A survey of community search over big graphs. *The VLDB Journal* 29 (2020), 353–392.

[20] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[21] Jun Gao, Jiazun Chen, Zhao Li, and Ji Zhang. 2021. ICS-GNN: lightweight interactive community search via graph neural network. *Proceedings of the VLDB Endowment* 14, 6 (2021), 1006–1018.

[22] Xiaoxuan Gou, Xiaoliang Xu, Xiangying Wu, Runhuai Chen, Yuxiang Wang, Tianxing Wu, and Xiangyu Ke. 2023. Effective and Efficient Community Search with Graph Embeddings. In *ECAI 2023*. IOS Press, 891–898.

[23] Farnoosh Hashemi, Ali Behrouz, and Milad Rezaei Hajidehi. 2023. CS-TGN: Community Search via Temporal Graph Neural Networks. In *Companion Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben (Eds.). ACM, 1196–1203. https://doi.org/10.1145/3543873.3587654

[24] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).

[25] Jiafeng Hu, Xiaowei Wu, Reynold Cheng, Siqiang Luo, and Yixiang Fang. 2016. Querying minimal steiner maximum-connected subgraphs in large graphs. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 1241–1250.

[26] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. 2014. Querying k-truss community in large and dynamic graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1311–1322.

[27] Xin Huang, Laks VS Lakshmanan, Jeffrey Xu Yu, and Hong Cheng. 2015. Approximate closest community search in networks. *Proceedings of the VLDB Endowment* 9, 4 (2015), 276–287.

[28] Yuli Jiang, Yu Rong, Hong Cheng, Xin Huang, Kangfei Zhao, and Junzhou Huang. 2022. Query driven-graph neural networks for community search: from non-attributed, attributed, to interactive attributed. *Proceedings of the VLDB Endowment* 15, 6 (2022), 1243–1255.

[29] Yizhu Jiao, Yun Xiong, Jiawei Zhang, Yao Zhang, Tianqi Zhang, and Yangyong Zhu. 2020. Sub-graph contrast for scalable self-supervised graph representation learning. In *2020 IEEE international conference on data mining (ICDM)*. IEEE, 222–231.

[30] Junghoon Kim, Siqiang Luo, Gao Cong, and Wenyuan Yu. 2022. DMCS: Density modularity based community search. In *Proceedings of the 2022 International Conference on Management of Data*. 889–903.

[31] Longhai Li, Lei Duan, Junchen Wang, Chengxin He, Zihao Chen, Guicai Xie, Song Deng, and Zhaohang Luo. 2023. Memory-enhanced transformer for representation learning on temporal heterogeneous graphs. *Data Science and Engineering* 8, 2 (2023), 98–111.

[32] Ling Li, Siqiang Luo, Yuhai Zhao, Caihua Shan, Zhengkui Wang, and Lu Qin. 2023. COCLEP: Contrastive Learning-based Semi-Supervised Community Search. *IEEE 39th ICDE* (2023).

[33] Ling Li, Yuhai Zhao, Siqiang Luo, Guoren Wang, and Zhengkui Wang. 2023. Efficient Community Search in Edge-Attributed Graphs. *IEEE Trans. Knowl. Data Eng.* 35, 10 (2023), 10790–10806. https://doi.org/10.1109/TKDE.2023.3267550

[34] Shunyang Li, Kai Wang, Xuemin Lin, Wenjie Zhang, Yizhang He, and Yuan Long. 2024. Querying Historical Cohesive Subgraphs over Temporal Bipartite Graphs. *2024 IEEE 40th International Conference on Data Engineering (ICDE)* (2024).

[35] Zhao Li, Pengrui Hui, Peng Zhang, Jiaming Huang, Biao Wang, Ling Tian, Ji Zhang, Jianliang Gao, and Xing Tang. 2021. What happens behind the scene? Towards fraud community detection in e-commerce from online to offline. In *Companion Proceedings of the Web Conference 2021*. 105–113.

[36] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and S Yu Philip. 2022. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering* 35, 6 (2022), 5879–5900.

[37] Yanjun Qi, Fernanda Balem, Christos Faloutsos, Judith Klein-Seetharaman, and Ziv Bar-Joseph. 2008. Protein complex identification by supervised graph local clustering. *Bioinformatics* 24, 13 (2008), i250–i268.

[38] Zekun Qi, Runpei Dong, Guofan Fan, Zheng Ge, Xiangyu Zhang, Kaisheng Ma, and Li Yi. 2023. Contrast with Reconstruct: Contrastive 3D Representation Learning Guided by Generative Pretraining. In *International Conference on Machine Learning (ICML)*.

[39] Santhosh Sankar and Nagasuma Chandra. 2022. SiteMotif: A graph-based algorithm for deriving structural motifs in Protein Ligand binding sites. *PLoS Computational Biology* 18, 2 (2022), e1009901.

[40] Yutaka Sasaki et al. 2007. The truth of the F-measure. *Teach tutor mater* 1, 5 (2007), 1–5.

[41] Gerhard Schlosser and Günter P Wagner. 2004. Modularity in development and evolution. (2004).

[42] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 815–823.

[43] Qing Sima, Jianke Yu, Xiaoyang Wang, Wenjie Zhang, Ying Zhang, and Xuemin Lin. 2024. Deep Overlapping Community Search via Subspace Embedding. *arXiv preprint arXiv:2404.14692* (2024).

[44] Mauro Sozio and Aristides Gionis. 2010. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 939–948.

[45] Lei Tang and Huan Liu. 2010. Graph mining applications to social network analysis. *Managing and mining graph data* (2010), 487–513.

[46] Hanchen Wang, Rong Hu, Ying Zhang, Lu Qin, Wei Wang, and Wenjie Zhang. 2022. Neural Subgraph Counting with Wasserstein Estimator. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 160–175. https://doi.org/10.1145/3514221.3526163

[47] Jianwei Wang, Kai Wang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. 2024. Neural Attributed Community Search at Billion Scale. *Proceedings of the ACM on Management of Data* 1, 4 (2024), 1–25.

[48] Jianwei Wang, Ying Zhang, Kai Wang, Xuemin Lin, and Wenjie Zhang. 2024. Missing Data Imputation with Uncertainty-Driven Network. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–25.

[49] Lei Wang, Zhu-Hong You, Yang-Ming Li, Kai Zheng, and Yu-An Huang. 2020. GCNCDA: a new method for predicting circRNA-disease associations based on graph convolutional network algorithm. *PLOS Computational Biology* 16, 5 (2020), e1007568.

[50] Yuxiang Wang, Xiaoxuan Gou, Xiaoliang Xu, Yuxia Geng, Xiangyu Ke, Tianxing Wu, Zhiyuan Yu, Runhuai Chen, and Xiangying Wu. 2024. Scalable Community Search over Large-scale Graphs based on Graph Transformer. In *SIGIR*.

[51] Yuxiang Wang, Shuzhan Ye, Xiaoliang Xu, Yuxia Geng, Zhenghe Zhao, Xiangyu Ke, and Tianxing Wu. 2024. Scalable Community Search with Accuracy Guarantee on Attributed Graphs. In *ICDE*.

[52] Yiqi Wang, Long Yuan, Zi Chen, Wenjie Zhang, Xuemin Lin, and Qing Liu. 2023. Towards efficient shortest path counting on billion-scale graphs. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2579–2592.

[53] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 726–735.

[54] Xueyi Wu, Yuanyuan Xu, Wenjie Zhang, and Ying Zhang. 2023. Billion-Scale Bipartite Graph Embedding: A Global-Local Induced Approach. *Proc. VLDB Endow.* 17, 2 (oct 2023), 175–183.

[55] Yubao Wu, Ruoming Jin, Jing Li, and Xiang Zhang. 2015. Robust local community detection: on free rider effect and its elimination. *Proceedings of the VLDB Endowment* 8, 7 (2015), 798–809.

[56] Yuanyuan Xu, Wenjie Zhang, Xiwei Xu, Binghao Li, and Ying Zhang. 2024. Scalable and Effective Temporal Graph Representation Learning With Hyperbolic Geometry. *IEEE Transactions on Neural Networks and Learning Systems* (2024).

[57] Jaewon Yang and Jure Leskovec. 2012. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*. 1–8.

[58] Shuo Yang, Zhiqiang Zhang, Jun Zhou, Yang Wang, Wang Sun, Xingyu Zhong, Yanming Fang, Quan Yu, and Yuan Qi. 2021. Financial risk analysis for SMEs with graph-based supply chain mining. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*. 4661–4667.

[59] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems* 34 (2021), 28877–28888.

[60] Si Zhang, Dawei Zhou, Mehmet Yigit Yildirim, Scott Alcorn, Jingrui He, Hasan Davulcu, and Hanghang Tong. 2017. Hidden: hierarchical dense subgraph detection with application to financial fraud detection. In *Proceedings of the 2017 SIAM international conference on data mining*. SIAM, 570–578.

[61] Tianming Zhang, Yunjun Gao, Jie Zhao, Lu Chen, Lu Jin, Zhengyi Yang, Bin Cao, and Jing Fan. 2024. Efficient Exact and Approximate Betweenness Centrality Computation for Temporal Graphs. In *Proceedings of the ACM on Web Conference 2024*. 2395–2406.

[62] Yao Zhang, Yun Xiong, Yun Ye, Tengfei Liu, Weiqiang Wang, Yangyong Zhu, and Philip S Yu. 2020. SEAL: Learning heuristics for community detection with generative adversarial networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1103–1113.

[63] Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Chee-Kong Lee. 2021. Motif-based graph self-supervised learning for molecular property prediction. *Advances in Neural Information Processing Systems* 34 (2021), 15870–15882.

[64] Jianan Zhao, Chaozhuo Li, Qianlong Wen, Yiqi Wang, Yuming Liu, Hao Sun, Xing Xie, and Yanfang Ye. 2021. Gophormer: Ego-graph transformer for node classification. *arXiv preprint arXiv:2110.13094* (2021).