A Comprehensive Survey and Experimental Study of Learning-based Community Search

Xiaoxuan Gou
Fudan University
Shanghai, China
xxgou24@m.fudan.edu.cn

Weiguo Zheng*
Fudan University
Shanghai, China
zhengweiguo@fudan.edu.cn

Yuxiang Wang
Hangzhou Dianzi University
Zhejiang, China
lsswyx@hdu.edu.cn

Xiaoliang Xu
Hangzhou Dianzi University
Zhejiang, China
xxl@hdu.edu.cn

Zhiyuan Yu
Hangzhou Dianzi University
Zhejiang, China
zy_yu@hdu.edu.cn

ABSTRACT

Given a graph G and a query node q, the goal of community search (CS) is to find a structurally cohesive subgraph from G that contains q. Significant progress has been made in community search using deep learning in recent years. To the best of our knowledge, no existing work has provided a comprehensive review of learning-based community search methods. Additionally, we find that: (1) Existing methods offer diverse definitions or descriptions of communities, which require systematic summarization. (2) The methods rely on distinct metrics for limited community assessment. (3) Overhead evaluations of the methods vary and exhibit certain biases.

Therefore, a comprehensive survey and experimental study are essential to achieve four key objectives: designing a unified pipeline, clarifying community definitions, enriching community evaluation, and establishing overhead assessment. In this paper, we first propose a unified pipeline for these methods, highlighting techniques. We categorize community definitions and analyze the relationships between identified communities. Beyond that, we proposed several community metrics to evaluate the communities comprehensively. Moreover, we introduce a more detailed overhead evaluation approach that considers resource consumption during both the training and search phases. Finally, we employ the proposed community evaluation metrics and overhead assessment framework to evaluate and analyze the methods, examine correlations among metrics, and explore the effects of several commonly used techniques.

PVLDB Reference Format:
Xiaoxuan Gou, Weiguo Zheng, Yuxiang Wang, Xiaoliang Xu, and Zhiyuan Yu. A Comprehensive Survey and Experimental Study of Learning-based Community Search. PVLDB, 18(9): 2941 - 2954, 2025.
doi:10.14778/3746405.3746419

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://anonymous.4open.science/r/CS_survey-864D/.

*Corresponding author

Proceedings of the VLDB Endowment, Vol. 18, No. 9 ISSN 2150-8097.
doi:10.14778/3746405.3746419

1 INTRODUCTION

Since graphs are widely used to represent entities (nodes) and their interconnections (edges) in various real-world applications [24, 44, 56, 70, 78, 88], graph mining has attracted extensive research interests [62], enabling the discovery of particular patterns and valuable insights within graphs. As a fundamental problem in graph mining, *Community Search* (CS) aims to identify a cohesive subgraph from a given graph G that contains the user-specified query node q [9, 36, 84, 85]. Community search can be applied across a variety of tasks, e.g., online recommendations [13, 16, 22, 29, 39, 54, 59, 82], expert finding [77, 81], and biological data analysis [20, 40, 42, 57, 86]. For example, in a movie database (e.g., IMDb), a recommendation system can perform CS using one of her favorite movies as a query and recommend movies based on the returned community of movies. In collaboration networks (e.g., DBLP), researchers can leverage CS to structure academic workshops based on the identified communities.

1.1 Background

The methods for community search can be roughly categorized into two groups, including non-learning-based methods and learning-based methods. Non-learning-based methods tackle the community search problem without relying on learning techniques. For more details, please refer to two survey papers [26, 27]. Significant progress in deep learning, particularly in Computer Vision (CV) with models like Vision Transformer (ViT) [19], and in Natural Language Processing (NLP) with models such as BERT [18] and GPT [64], has led to the development of numerous learning-based methods [14, 21, 23, 37, 41, 52, 53, 74–76], aimed at addressing the challenges of community search. Despite of the rapid advancement of such methods, there is no systematic analysis or survey of learning-based community search methods yet.

In this paper, we present a systematic analysis of learning-based community search methods. Given the crucial role of community search, a comprehensive survey is warranted that goes beyond merely listing existing methods, providing an overview of the development roadmap, summarizing common techniques, and conducting systematic experimental analysis. Our goal is to systematically evaluate the learning-based community search methods, providing system designers with insights to make more informed choices.

To achieve this, we need to address the following challenges:

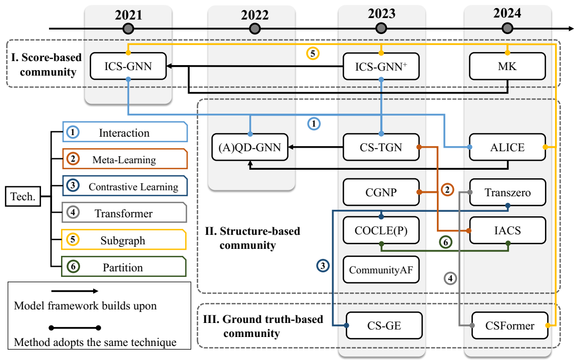
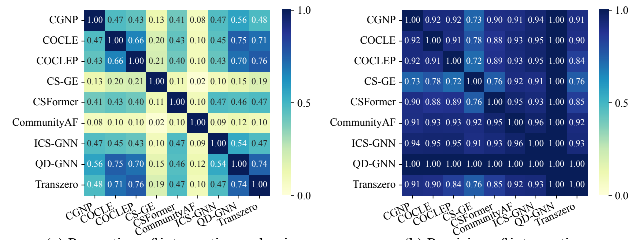


Figure 1: Roadmap of learning-based CS methods

(1) **Diverse definitions of communities**. Existing methods offer diverse definitions or descriptions of communities. Is it possible to integrate these definitions into a unified framework? What, if any, are the relationships or connections that link these different community definitions?
(2) **Variability in community evaluation metrics**. Various metrics have been introduced to evaluate community quality. However, different communities often rely on distinct metrics in assessments, making comprehensive comparisons difficult.
(3) **Incompletion in overhead evaluation**. Evaluating learning-based CS methods requires not only measuring their effectiveness but also analyzing the associated costs, including both time and space overhead. However, most existing approaches report only partial cost metrics, typically focusing on either time or space overhead during the training and search phases.

1.2 Contributions

Hightlighting Trends & Pipeline. We first outline the development paths of recent methods, highlighting the technical connections between them. Figure 1 illustrates these connections, with colored lines linking methods that share similar techniques. The evolution of these methods is driven by various factors, including application requirements, data resources, model architectures, and computational resources. From an application perspective, interactive search strategies (①) can significantly enhance user experience in real-world applications. For training data, contrastive learning (③) and meta-learning (②) are effective in addressing few-shot and cross-task challenges. In terms of model architecture, Graph Transformers (④) help alleviate issues like over smoothing [10] and over squashing [2] often encountered in Graph Neural Networks (GNNs). Furthermore, candidate subgraphs (⑤) and graph partitioning strategies (⑥) are employed to reduce computational overhead and improve search efficiency. We also analyze the workflow of these methods and summarize a unified pipeline consisting of three stages (preparation, training, and search), presenting guidance and suggestions for subsequent researchers in method design. This pipeline provides a structured guide for future researchers in method design, identifying where each technique is applied.
Exploring Connections Among Communities. We review diverse definitions and descriptions of communities from various methods and find it challenging to generalize all communities under a single definition, as community characteristics vary depending



(a) Proportion of intersection and union between each methods' community

(b) Precision of intersection

Figure 2: Heatmap of different methods' community intersection on Cora dataset

Table 1: Metrics for evaluating communities

Methods	F1	Pre	Rec	NMI	ARI	JAC	Community Quality* (Table 7)
ICS-GNN [33]	✔	✔	✔	✘	✘	✘	✘
(A)QD-GNN [46]	✔	✘	✘	✘	✘	✘	✘
ICS-GNN$^+$ [11]	✔	✔	✔	✘	✘	✘	✘
COCLE(P) [52]	✔	✘	✘	✔	✘	✔	✘
CS-TGN [41]	✔	✘	✘	✘	✘	✘	✘
CommunityAF [14]	✔	✘	✘	✘	✘	✘	✘
CS-GE [37]	✔	✘	✘	✘	✘	✘	✘
CGNP [21]	✔	✔	✔	✘	✘	✘	✘
Transzero [74]	✔	✘	✘	✔	✘	✔	✘
ALICE [75]	✔	✘	✘	✘	✘	✘	✔ (average degree [65])
CSFormer [76]	✔	✘	✘	✘	✘	✘	✘
MK [53]	✔	✘	✘	✘	✘	✘	✘
IACS [23]	✔	✔	✔	✘	✘	✘	✘

*Detailed community quality metrics are listed in Table 7.

on particular requirements and application contexts. Therefore, we distinguish between explicitly defined and implicitly described communities, categorizing the existing communities into three types: score-based, structure-based, and ground truth-based communities.

Furthermore, we investigate the connections among communities discovered by different methods. Figure 2(a) shows the ratio of community intersection to community union across methods on the Cora dataset. The low ratios observed between most methods suggest that communities identified by different approaches share few common nodes. However, as illustrated in Figure 2(b), the precision of nodes within these intersections is relatively high, characterizing the significance of these shared nodes.
Comprehensive Evaluation of Identified Communities. Evaluating discovered communities reflects the accuracy performance of different methods. We summarize the metrics for community evaluation used by various methods in Table 1. Notably, while all methods employ the F1-score to measure how closely the identified communities align with the ground truth, not all report metrics related to community quality, such as cohesiveness-related metrics (e.g., TPR [83] and FOMD [83]), despite their significance in evaluating community structures. This raises an important question: do methods that perform well on accuracy metrics also excel in community quality metrics? To address this, we analyze the ranking shifts of various methods across multiple metrics on four datasets, as illustrated in Figure 3. The results demonstrate that high F1-scores do not necessarily translate to strong performance on cohesiveness-specific metrics. This finding underscores the limitations of relying solely on accuracy metrics like F1-score to evaluate overall community quality.
Overhead Evaluation of Different Methods. Besides evaluating the effectiveness of a method, it is essential to consider the resources required to achieve this performance by analyzing each method's computational costs, e.g., GPU usage and processing time. Specifically, we report computational resource demands and time

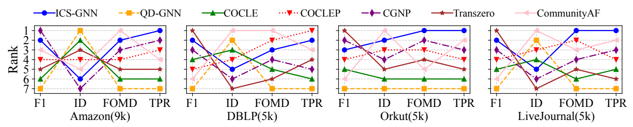


Figure 3: Method rankings across different evaluation metrics

Table 2: Metrics for evaluating community search methods

Methods	Train Phase			Search Phase	
	Parameters	GPU	Time	GPU	Time
ICS-GNN [33]	✗	✗	✓	✗	✓
(A)QD-GNN [46]	✗	✗	✓	✗	✓
ICS-GNN^{+} [11]	✗	✗	✓	✗	✓
COCLE(P) [52]	✗	✗	✓	✗	✓
CS-TGN [41]	✗	✗	✗	✗	✓
CommunityAF [14]	✗	✗	✓	✗	✗
CS-GE [37]	✓	✗	✓	✗	✓
CGNP [21]	✗	✗	✓	✗	✓
Transzero [74]	✗	✗	✓	✗	✓
ALICE [75]	✗	✗	✓	✗	✓
CSFormer [76]	✓	✗	✓	✗	✓
MK [53]	✗	✓	✓	✗	✓
IACS [23]	✗	✗	✓	✗	✓

consumption across both the training and search phases. For the training phase, key factors include model parameters, GPU usage, and training time per epoch; for the search phase, we focus primarily on GPU usage and search time. However, as depicted in Table 2, the evaluation of existing methods remains incomplete, highlighting the need for comprehensive cost evaluations to enable a more in-depth comparison of method performance.

In summary, we make the following contributions in this paper.

- To the best of our knowledge, we are the first to review the evolution of learning-based community search methods and highlight the technical connections among them. We also propose a unified, three-stage pipeline to summarize the existing methods.
- We classify existing community definitions into three categories and investigate their connections by analyzing shared characteristics across various methods. We observe that nodes within community intersections often demonstrate high precision.
- We conduct a comprehensive evaluation of communities identified by various methods, examining community assessment metrics across multiple datasets. Our findings highlight that accuracy metrics alone do not fully capture community quality.
- In addition to the effectiveness evaluation of different methods, we systematically compare the computational overhead of these methods, offering a detailed performance comparison.

2 DEFINITION AND PRELIMINARY

We consider an undirected graph $G = (V, E)$ with a node set $V = \{v_1, v_2, ..., v_{|V|}\}$ and an edge set $E = \{e_{v_i v_j} | v_i, v_j \in V\}$, where $e_{v_i v_j}$ represents the edge between v_i and v_j. Let $N(v)$ denote the neighbors of node v, i.e., $N(v) = \{u \in V | e_{u,v} \in E\}$. Thus, the degree of node v is denoted as $deg(v) = |N(v)|$. The graph can be represented by using an adjacency matrix $A \in \mathbb{R}^{|V| \times |V|}$, where $A_{ij} = 1$ if $e_{ij} \in E$ and $A_{ij} = 0$ if $e_{ij} \notin E$. A graph may have an initial node feature matrix $X \in \mathbb{R}^{|V| \times F}$, where each row $x_{v_i} \in \mathbb{R}^F$ corresponds to the feature vector of node v_i, x_{v_i} is the i-th row of the matrix X.

Definition 1 (**Community Search (CS)**). *For a graph $G = (V, E)$, given a query node $v_q \in V$, the problem of Community Search (CS) is to find a query-dependent subgraph (i.e., a community), denoted as C_q, that contains v_q.*

Table 3: Community definition

Explicit		Implicit	
Score-based	Structure-based	Ground Truth-based	
ICS-GNN [33]	CS-GE [37]	(A)QD-GNN [46]	CGNP [21]
ICS-GNN^{+} [11]	CSFormer [76]	COCLEP [52]	CommunityAF [14]
MK [53]		CS-TGN [41]	IACS [23]
		ALICE [75]	Transzero [74]

The differences in definitions of the CS problem across various methods primarily stem from how each method defines a community. As shown in Table 3, existing communities can be roughly categorized into three groups: (1) score-based community, (2) structure-based community, and (3) ground truth-based community.

Definition 2 (**Score-based Community**). *Given a community of size m and a score function $f(\cdot)$ that is used to evaluate the community score of individual nodes, the score-based community, denoted as $G_C = (V_C, E_C)$, is a subgraph with the requirements: (1) The query vertex $q \in V_C$, and G_C is connected, (2) $|V_C| = m$, and (3) The community score $\sum_{i=1}^{m} f(v_i)$ is maximized.*

ICS-GNN [33] and ICS-GNN^{+} [11] utilize a GNN to score nodes, returning an m-sized community that includes the query node and has the highest GNN scores. Similarly, MK [53] employs GNN-based scoring but introduces an additional constraint on node types, providing an m-sized community that includes the query node, maximizes GNN scores, and ensures uniform node types throughout.

Definition 3 (**k-core** [3–5, 28]). *Given an undirected graph $G = (V, E)$ and a non-negative integer k, a k-core of G is the largest subgraph $H_k \subseteq G$, such that $\forall v \in H_k$ has degree at least k, i.e., $deg(v, H_k) \geq k$, where $deg(v, H_k)$ denotes the degree of v in H_k.*

Definition 4 (**Structure-based Community**). *Structure-based communities impose specific structural constraints to characterize the structural cohesiveness of the community, where cohesiveness means that nodes in the community are intensively linked to each other according to a particular subgraph metric, e.g., k-core (k quantifies the cohesiveness of the community).*

CS-GE [37] defines a community as a k-core [3–5, 28], returning a k-core community that includes the query node with a cohesiveness parameter k. Similarly, CSFormer [76] also identifies k-core communities but returns the k-core community containing the query node with the highest cohesiveness.

Score-based and structure-based communities offer explicit definitions of what constitutes a community. In contrast, another research direction focuses on communities defined implicitly, where the community structure is not clearly specified but is instead learned from ground-truth communities provided in the dataset. We refer to this type as the ground truth-based community.

Definition 5 (**Ground truth-based community**). *Ground truth-based community is a connected subgraph in which nodes are densely intra-connected and share similar attributes. For training and evaluation purposes, the ground-truth communities are manually annotated within the datasets.*

Based on different supporting graphs, we can categorize community search problems as follows:

Table 4: Learning-based community search methods

Methods	Graph Sample	Learning Paradigm	Backbone	Contrastive	Interactive
ICS-GNN	Candidate Subgraph	supervised	GCN	✘	✔
(A)QD-GNN	-	supervised	GCN	✘	✔
ICS-GNN$^+$	Candidate Subgraph	supervised	GCN	✘	✔
COCLE	-	semi-supervised	GCN	✔	✘
COCLEP	Graph Partitioning	semi-supervised	GCN	✔	✘
CS-TGN	-	meta-learning	GCN	✘	✔
CommunityAF	-	supervised	iGPN	✘	✘
CS-GE	-	supervised	GCN	✔	✘
CGNP	-	meta-learning	GCN	✘	✘
Transzero	-	unsupervised	GT	✔	✘
ALICE	Candidate Subgraph	supervised	GIN	✘	✔
CSFormer	Candidate Subgraph	supervised	GT	✘	✘
MK	Candidate Subgraph	supervised	HGNN	✘	✘
IACS	Graph Partitioning	meta-learning	GAE	✘	✘

(1) Basic Community Search (BCS): Methods like QD-GNN [46], ICS-GNN [33], CommunityAF [14], COCLEP [52], CGNP [21], Transzero [74], and IACS [23] aim to identify a query-dependent, densely connected subgraph C_q.
(2) Attributed Community Search (ACS): ACS finds communities that possess both structural cohesiveness and attribute homogeneity, meaning that nodes within the community are densely intra-connected and share similar attributes. For example, ALICE [75], AQD-GNN [46], and the attributed version of IACS.
(3) Community Search in Temporal Networks (CST): CST aims to identify a query-dependent community that maintains connectivity and structural cohesiveness across a temporal network $G = \{G^1, \ldots, G^t\}$, such as CS-TGN [41].

This paper primarily focuses on a comparative study of learning-based methods for solving the basic community search problem.

3 OVERVIEW OF LEARNING-BASED COMMUNITY SEARCH

We categorize and summarize learning-based community search methods according to the three community definitions presented in Section 2. We also provide development roadmaps (Figure 1) and present a summary of key properties for these methods in Table 4.

3.1 Methods for Score-based Community

These methods generally aim to identify a k-sized maximum-GNN-score (kMG) community, where scores are derived from the predicted GNN scores of nodes, indicating the likelihood of each node belonging to a community. They focus on optimizing three key aspects: (1) Candidate Update: refining the candidate subgraph based on user feedback; (2) Model Training: enhancing the model's accuracy in predicting community membership; and (3) Search Phase: efficiently locating the kMG community, particularly when the query node lies at the community boundary.

ICS-GNN [33] is the first method to leverage GNN models for solving the interactive community search problem. The ICS-GNN process consists of the following steps: first, a candidate subgraph is constructed using breadth-first search (BFS) and edge enhancement strategies, then trained with a GNN model to infer the probability of nodes belonging to a community. Next, the GNN model predicts node scores, allowing for the identification of a k-sized Maximum-GNN-score (kMG) community. Finally, users can provide feedback on the retrieved community to refine subsequent model training.

ICS-GNN$^+$ [11] is an enhanced version of ICS-GNN. It introduces an adaptive algorithm for candidate subgraph maintenance to effectively control the size of the candidate subgraph. To further enhance GNN performance, ICS-GNN$^+$ incorporates an unsupervised clustering loss during the training phase, ensuring node embeddings closely align with their respective cluster centers.

MK [53] extends the ICS-GNN approach to heterogeneous attributed graphs by introducing the MK framework which is a comprehensive solution with modules for local subgraph construction, HGNN training and inference, and community rewriting. To enhance training and query efficiency, the framework incorporates various optimization techniques, including a self-training algorithm that reduces the need for user-provided labels.

3.2 Methods for Structure-based Community

These methods generally aim to identify communities using structural community models, such as k-core [3–5, 28] and k-truss [1, 43, 45, 55]. They often need to address two primary challenges: (1) efficiently handling searches for arbitrary cohesiveness values k across various community models, and (2) managing large-scale, complex scenarios, such as dynamic graphs.

CS-GE [37] is a community search framework leveraging graph embeddings and operates in two stages: offline learning and online search. In the offline stage, CS-GE introduces a k-core injected graph embedding technique to capture cohesiveness features for any specified k. During the online stage, it constructs a Proximity Graph (PG) to facilitate rapid community retrieval.

CSFormer [76] introduces a novel neighborhood community vector, leveraging the n-th order h-index to effectively capture structural features related to community cohesiveness, which are beneficial for community search (CS) tasks. To handle large-scale graphs, CSFormer employs a transformer encoder model with an attention-based readout function to process neighborhood community vectors, enabling efficient online CS through lightweight coreness prediction. Additionally, CSFormer extends the solution to more complex scenarios, including CS on dynamic graphs and CS with various community models.

3.3 Methods for Ground Truth-based Community

These methods leverage ground-truth communities as supervised signals to identify cohesive communities. They primarily focus on: (1) handling large-scale scenarios with limited computational resources, (2) operating effectively with limited or unlabeled data, and (3) enabling inductive inference.

QD-GNN [46] is the first query-driven GNN model designed specifically for community search. It integrates local query-dependent structures with global node embeddings to effectively return community results based on any given query using trained models. AQD-GNN [46] enhances the capabilities of QD-GNN by incorporating node attribute bipartite graphs into its learning framework.

ALICE [75] builds upon the idea of QD-GNN and introduces a novel learning-based approach to improve the performance of ACS. It utilizes a two-step methodology: first, it identifies promising candidate subgraphs to narrow the search scope, then it performs community searches using ConNet.

CS-TGN [41] extends the framework of QD-GNN to temporal graphs and designs a query-driven temporal graph convolutional

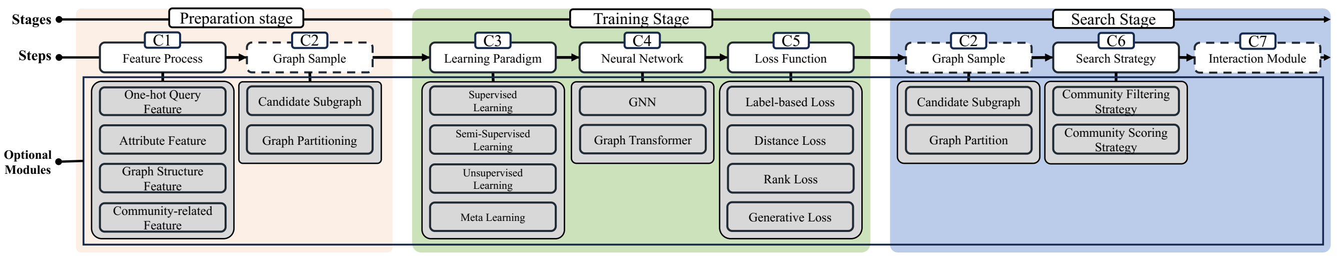


Figure 4: Process of CS methods. Dotted lines represent an optional step, and grey regions indicate the available modules.

neural network that integrates the local query-dependent structure and global node embeddings at each timestamp.

CommunityAF [14] is a community generation framework consisting of two key components: (1) SEAL [87] integrates community-aware structural features to learn node embeddings for large-scale graphs while minimizing computational resource requirements and enabling incremental embedding updates; and (2) An autoregressive flow-based generation (AF) component, which is designed to select the next node to be added to the current community.

COCLE [52] is a semi-supervised model utilizing contrastive learning. It employs a hypergraph-based augmentation method along with a contrastive learning strategy that integrates low-order intra-view learning, high-order intra-view learning, and low-high-order inter-view learning. Moreover, COCLEP [52], an extension of COCLE, further reduces training resource costs via graph partitioning.

CGNP [21] is a novel framework built on the conceptual Conditional Neural Process [34] (CNP), designed to learn the meta model from an efficient, metric-based learning perspective. It is the first approach to investigate the application of meta-learning techniques to address community search (CS) problems with limited data.

Transzero [74] is a learning-based CS framework that runs without using labeled data. In the offline pre-training phase, two self-supervised losses including the personalization loss and link loss are utilized to pre-train the model without using labels. In the online search phase, Transzero finds promising communities based on the new proposed expected score gain function.

IACS [23] utilizes a GNN-based encoder-decoder model tailored for heterogeneous attributed ACS tasks and can generalize across diverse datasets. The IACS framework follows a structured workflow: it begins by training a shared model using a meta-algorithm across multiple ACS tasks, then fine-tunes the model for new tasks with limited data, and finally deploys it for online queries.

4 COMPONENT ANALYSIS

Despite the diversity of learning-based CS methods, they all adhere to a unified processing pipeline. As illustrated in Figure 4, methods can be broadly categorized into three parts: preparation (left), training (middle), and search (right). We further decompose these three parts into seven fine-grained components (C1–C7 in Figure 4).

4.1 Preparation

As Figure 4 (left) shows, the preparation part can be divided into two components: Feature Processing (C1) and Graph Sample (C2).

4.1.1 C1: Feature Process. In graph learning, we need to represent node or edge features using matrices (or vectors). The main ways of representing node features in learning-based community search methods are summarized as follows:

One-hot Query Feature [46]. Given a graph $G = \{V, E\}$, each query node v_q is encoded as a one-hot vector $x_q = \{0, 1\}^{|V|}$, where the i-th bit equals to one if v_i is a query node.

Attribute Feature [46, 75]. If nodes have numerical or categorical attributes, these attribute values can be directly used as node features. Given an attribute set $attr = \{a_1, a_2, ...\}$, each node attribute feature can be encoded as $x_a \in \{0, 1\}^{|attr|}$, where the i-th bit equals to one if there exists an attribute a_i while $a_i \subseteq attr$.

Graph Structural Feature. Structural features are derived from the graph's structural information, leveraging learning algorithms such as DeepWalk [63] and Node2Vec [38]. These algorithms capture both local and global structural information of the nodes.

Community-related Feature [76]. Community-related features can be designed and extracted using community-specific information. For example, COCLEP constructs the feature matrix by utilizing the normalized core number of each node. The core number of a node u is the maximum k that a k-core contains u [25].

4.1.2 C2: Graph Sample. In many real-world applications, graphs can contain millions or even billions of nodes and edges, making direct computation on the entire graph both time-consuming and resource-intensive. To mitigate this, graph sampling techniques are employed to create smaller, more manageable subgraphs, thus simplifying the computation process. In learning-based community search methods, sampling strategies are typically applied either prior to training or during the search phase. These methods can generally be classified into two main categories (as shown in Table 4): (1) Candidate subgraph sampling and (2) Graph partitioning.

Candidate Subgraph Sampling [33]. Given a query node, the sampling process is to extract a subgraph containing the query node, typically via graph traversal from the query node, for instance, by employing a breadth-first search (BFS) from the query node. According to small-world theory [51, 58], community members typically exhibit strong access locality [79], meaning that two nodes are more likely to belong to the same community if they are located within each other's localized space. Several methods have developed candidate subgraph strategies tailored to their particular needs. For example, ICS-GNN [11] utilizes an edge enhancement strategy to capture as many nodes related to the target community as possible, while ALICE [75] applies a strategy based on modified modularity to ensure the quality of the candidate subgraph.

Graph Partitioning. Graph partitioning aims to decompose the original graph into multiple smaller, mutually exclusive subgraphs, each constructed from distinct node groups [6]. Graph partitioning may separate nodes that originally belong to the same community into different subgraphs. If a community spans multiple partition boundaries, training based solely on a single subgraph may result in the loss of critical community information. COCLEP [52] tackles the issue of out-of-memory when training on large graphs by employing graph partitioning methods such as Metis [47] and mitigating community information loss through the boundary nodes.

4.2 Training Stage

As Figure 4 (middle) shows, the training part can be divided into three components: Learning Paradigm (C3), Neural Network (C4), and Loss Function (C5). Generally, we need to select the appropriate learning paradigm based on the practical scenario, followed by determining the model architecture, and finally selecting the suitable loss function for model training.

4.2.1 *C3: Learning Paradigm.*

The choice of an appropriate learning paradigm depends on the specific characteristics of the problem, the availability of data, and the objectives of the task. Based on their reliance on labeled data, learning-based community search methods can be categorized into the following types:

Supervised Learning. Most learning-based CS methods depend on a substantial amount of labeled data because of supervised learning techniques. (A)QD-GNN [46], a representative method based on supervised learning, constructs the training set using query nodes and ground truth communities. With a sufficient number of training samples, the model is capable of accurately predicting communities.

Semi-Supervised Learning. In many scenarios, obtaining labeled data is often difficult, resulting in only a small amount of labeled data being available. COCLE(P) [52] employs a contrastive learning-based multi-view strategy. By constructing the original graph and an augmented hypergraph, it captures node representations from different perspectives. This approach allows the model to learn node features from multiple angles, enhancing its understanding of the community structure.

Unsupervised Learning. Self-supervised learning is a good strategy as it does not rely on any labeled data but utilizes the structure and attributes of the data itself to generate training signals. For example, Transzero [74] utilizes a link prediction loss to enable the model to learn the original structure of the graph, treating the edges in the graph as supervised signals.

Meta Learning. Meta-learning [71] enables a model to extract useful information from a few samples by sharing knowledge across different tasks. The rapid development of meta-learning has given rise to numerous meta-learning frameworks, such as MAML [30], Reptile [61], Prototypical Networks [69], and LEO [67]. This approach allows the model to quickly adapt to new tasks with minimal data. In CS tasks, the model can swiftly adjust to new query nodes or new graph structures without needing to learn from scratch.

4.2.2 *C4: Neural Network.*

The rise of deep learning has led to the development of numerous models capable of learning and performing a wide range of tasks. In particular, Graph Neural Networks (GNNs) have significantly advanced graph-based learning approaches. In this section, we will introduce the commonly used model architectures for learning-based community search methods.

Table 5: Loss function of learning-based community search

Methods	Loss Function
ICS-GNN	BCE loss (Eq. (2)), Rank Loss (Eq. (9))
(A)QD-GNN	BCE loss (Eq. (2))
ICS-GNN$^+$	BCE loss (Eq. (2)), Rank Loss (Eq. (9)), MSE Loss (Eq. (5))
COCLE(P)	Distance Loss (Eq. (7))
CS-TGN	BCE loss (Eq. (2))
CommunityAF	Generate Loss, Rank Loss (Eq. (9))
CS-GE	Distance Loss (Eq. (8))
CGNP	BCE loss (Eq. (2))
Transzero	Distance Loss (Eq. (8) and Eq. (6))
ALICE	BCE loss (Eq. (2)), Distance Loss (Eq. (5) and Eq. (6))
CSFormer	Cross-Entropy Loss (Eq. 3)
MK	BCE loss (Eq. (2))
IACS	BCE loss (Eq. (2))

Graph Neural Network (GNN). GNNs have emerged as a powerful technique for modeling graph-structured data, enabling the solution of a wide range of tasks on graph data, including node classification, link prediction, and graph classification, among others. The most widely used GNN models, such as GCN [50], GAT [73], and GIN [80], are based on the message-passing mechanism. In each GNN layer, node representations are updated by aggregating information from neighboring nodes, allowing the model to capture the graph's structural properties. Below, we outline the general process for updating a node's representation at layer l:

$$h_v^{(l)} = \mathcal{AGG}^{(l)}(h_v^{(l-1)}, \{h_u^{(l-1)}, u \in N(v)\}), \tag{1}$$

where $h_v^{(l)}$ is the node v's representation at layer l and $\mathcal{AGG}^{(l)}$ denotes the aggregation function.

Some methods incorporate variants of GNNs to meet specific requirements. For instance, CommunityAF [14] employs iGPN, a GNN designed for incremental scenarios, to facilitate generative community search. CS-TGN [41] utilizes temporal GNNs to capture temporal graph information, enabling CS on temporal graphs.

Graph Transformer (GT). Graph transformer [12] can handle serialized graphs, with memory consumption during computation remaining unaffected by the graph's scale, while mitigating the common issues of over-smoothing and over-squashing seen in GNNs. In CSFormer [76], the graph transformer learns the relationships between neighborhood tokens in the neighborhood community vector sequence, generating node representations that incorporate neighborhood community information for downstream tasks. Similarly, Transzero [74] leverages graph transformers to obtain both community-level and node-level representations, using contrastive learning for node representation training.

4.2.3 *C5: Loss Function.*

In learning-based CS methods, the choice of loss function is typically dictated by the type of downstream task. In this section, we will analyze various categories of downstream tasks based on their corresponding loss functions. Table 5 provides a summary of the loss functions used by various methods.

Label-based Loss. Label-based loss is primarily used in classification tasks with labeled data. It measures the difference between the model's predictions and the actual labels, guiding training by using the difference to iteratively optimize the model's parameters.

Binary Cross Entropy Loss. The binary cross entropy (BCE) loss function is commonly used for binary classification tasks. Some methods primarily use the BCE loss function to enable the model to determine whether candidate nodes belong to the community of the query node, which also means that during the search phase, the model can filter out community nodes from the candidate nodes. The BCE loss function is defined as follows:

$$\mathcal{L}_{bce} = -\sum_{v \in V} (y_v \cdot \log(\hat{y}_v) + (1 - y_v) \cdot \log(1 - \hat{y}_v)), \quad (2)$$

where $y_v \in \{0, 1\}$ is the true label of node v and $\hat{y}_v \in [0, 1]$ is the predicted probability that node v belongs to the query's community.
Cross Entropy Loss. When more complex classification tasks are required beyond simple binary classification, the cross-entropy loss function is suitable for handling multi-class classification tasks. The cross-entropy loss function is defined as follows:

$$\mathcal{L}_{ce} = -\boldsymbol{y} \log \boldsymbol{y}_v^*, \quad (3)$$

where $\boldsymbol{y}_v$ represents the actual label (usually one-hot encoded) and $\boldsymbol{y}_v^*$ represents the predicted probability distribution for node v.
Link Loss. Link Loss is a loss function measuring the connections between nodes in a graph and is commonly used in link prediction tasks. It's also widely used in self-supervised tasks within graphs, where the edges in the graph serve as supervised signals to learn structural information. The link loss function is defined as follows:

$$\mathcal{L}_{link} = \sum_{u \in V} \sum_{v \in N(u)} -e_{uv}\sigma(h_u^T h_v) + (1 - e_{uv})\sigma(h_u^T h_v), \quad (4)$$

where $e_{uv} = \{0, 1\}$ represents the edge between node u and v (1 and 0 indicate the edge's existence or not) and $\sigma(x) = \frac{1}{1+e^{-x}}$ is a sigmoid function mapping the result to a value between 0 and 1.
Distance Loss. Distance loss is a type of loss function used to optimize the distance between samples, particularly in representation learning and metric learning tasks. It plays a crucial role in contrastive learning, where the primary objective is to minimize the distance between similar samples while maximizing the distance between dissimilar ones, thereby enabling the model to learn an appropriate feature space.
Simple Distance Loss. These loss functions are typically derived from distance metrics such as Euclidean distance or cosine similarity. Minimizing the objective function corresponds to reducing the distance between target samples, thereby improving the model's ability to capture meaningful relationships in the feature space.

(1) Mean Squared Error Loss. The mean squared error (MSE) loss function calculates the average of the squared differences between the predicted values and the true values. Besides, MSE loss can also measure the squared Euclidean distance between the predicted values and the true values. In ICS-GNN$^+$ [11], the MSE loss is used to minimize the distance between community membership and the community centroid in the vector space:

$$\mathcal{L}_{mse} = \frac{1}{|V_C|} \sum_{u \in V_C} (h_u - h_C)^2, \quad (5)$$

where h_u is the community membership vector and h_C represents the community centroid vector.

(2) Inner Product Loss: this loss function is based on the inner product calculation between two node vectors (h_u and h_v). The loss function is defined as follows:

$$\mathcal{L}_{inner} = -\sigma(h_u \cdot h_v), \quad (6)$$

Contrastive Loss. In the contrastive loss, distance loss plays a central role as it defines the similarity or dissimilarity between pairs of samples. A typical way for constructing a contrastive loss function involves leveraging the softmax function to distinguish between positive and negative samples, as seen in methods like NT-Xent [72] and InfoNCE [15], which can be formalized as follows:

$$\mathcal{L}_{contrast} = -\log\left(\frac{\exp(sim(h_q, h_u)/\tau)}{\sum_{v \in V} \exp(sim(h_q, h_v)/\tau)}\right), \quad (7)$$

where h_q represents query's embedding and h_u is the positive sample's embedding. τ denotes the temperature parameter that controls the sensitivity of penalties on hard negative samples. The cosine distance function $sim(\cdot)$ measures the embeddings' similarity.
Triplet Loss. Triplet Margin (TM) optimization strategy directly enforces an increase in the relative distance between positive and negative example pairs. Given an anchor v_s, a corresponding positive sample v_+, and a negative sample v_-, the loss is formulated as:

$$\mathcal{L}_{tm} = \max(d(\vec{v_s}, \vec{v_+}) - d(\vec{v_s}, \vec{v_-}) + c, 0), \quad (8)$$

where c is a margin value and $d(\cdot)$ is distance function measuring similarity between samples.
Rank Loss. Ranking loss functions play a crucial role in tasks such as information retrieval [7], where optimizing the performance of the ranking model enhances the relevance of results. Ranking loss can be used for: (1) interactive approaches: optimizing the returned results based on user feedback, and (2) generative strategies: ensuring that the results generated at a current snapshot are superior to those generated at a previous snapshot. Pairwise ranking loss focuses on the ranking relationship between pairs of samples. Given a pair of sample nodes (v, u), the goal is to ensure that v has a higher ranking score than u. The rank loss is formulated as:

$$\mathcal{L}_{rank} = \sum_{v,u \in V} \max(0, S_v - S_u - \gamma), \quad (9)$$

where S_v (S_u) represents the score of node v (u) and γ is a threshold ensuring a minimum difference between the scores of v and u.
Generative Loss. Generative models (e.g., GAN [35], Flow [66], and VAE [49]) are learned to estimate the likelihood of data to be close to the true data distribution. In CS, the generative loss function ensures that the generated communities are as close as possible to the true community distribution.

4.3 Search Stage

As Figure 4 (right) shows, the search part involves three components: Graph Sample (C2), Search Strategy (C6), and Interaction (C7). In this stage, we can also apply graph sampling (mentioned in Section 4.1.2) to reduce the search space, then use a search strategy with a trained model to return the identified community. Finally, an interactive approach can be employed to manually evaluate the communities and generate updated training samples for fine-tuning.

4.3.1 C6: Search Strategy. Depending on the search strategies employed, the methods can be classified into two categories: (1) community filtering strategy and (2) community scoring strategy.
Community Filtering Strategy. A filter-based strategy compares the model-predicted scores (label) with a preset threshold (label) in the search to determine whether the candidate nodes belong to the

Table 6: Dataset statistics

Datasets	Abbr.	# Nodes	# Edges	$feature_{\text{dim}}$	d_{avg}	# Commmunity
EMAIL-EU	Emai	1,005	16,064	-	31.97	42
Cora	Cora	2,708	5,429	1433	3.9	7
Citeseer	Cite	3,312	4,732	3703	2.7	6
DBLP	DBLP	317080	1049866	-	6.62	13,477
Amazon	Amaz	334863	925872	-	5.53	75,149
Orkut	Orku	3,072,441	117,185,083	-	76.28	15,301,901
LiveJournal	Live	3,997,962	34,681,189	-	17.35	664,414

communities of query nodes. The search expands outward from the query node, employing the strategy to filter out community nodes. Based on the different types of scores, filter-based methods can be classified into three categories: (1) Probability score-based, (2) Similarity score-based, and (3) Category-based.

Probability Score-based. (A)QD-GNN [46], CS-TGN [41] and AL-ICE [75] use the trained model to directly predict candidate nodes' probability score (denoted as $S_{prob} \in [0, 1]$) indicating the likelihood that the candidate node belongs to query node's community. If the predicted probability score exceeds the threshold, the candidate node is considered a community membership, and vice versa.

Similarity Score-based. These methods generate node embeddings with the trained model, compute similarities between query and candidate node embeddings, and apply a threshold or top-k filter to identify the query node's community. CS-GE [37] calculates the cosine similarity between the query node and candidate nodes, and then returns the identified community using a top-k strategy.

Category-based. These methods use the trained model for node classification and apply predefined rules during the search phase to filter candidates. CSFormer [76] filters candidate nodes during the search by predicting their coreness and selecting those with a coreness greater than that of the query node.

Community Scoring Strategy. The community score-based strategy calculates node scores during the search process to identify the highest-score community containing the query node. The method traverses the graph from a query node, maintaining the max score community, and terminates when a stopping condition is met (e.g., all nodes are searched or scores no longer improve). Based on the method of score calculation, the strategies can be classified into two categories: (1) Probability score and (2) Predefined score.

Probability Score. In the community score-based strategy, the probability score is used to assess which candidate nodes are more likely to be members of the query node's community. Based on the probability scores predicted by the GNN, ICS-GNN and ICS-GNN$^+$ maintain a k-sized Maximum-GNN-score (kMG) community using a vertex swapping strategy.

Predefined Score. Predefined scores are calculated using predefined formulas or learnable functions to quantify community properties (e.g., cohesiveness). For example, in Transzero [74], the ESG score indicates that higher values correspond to stronger cohesiveness and greater relevance to the query node. Similarly, CommunityAF [14] uses a scoring component (MLP) to predict community scores, allowing the model to decide whether to continue or stop the generation process based on score changes.

4.3.2 *C7: Interactive module.*

To better adapt to real-world scenarios, some methods incorporate interactive functionality, allowing users to provide feedback and update data labels for retraining. ICS-GNN and ICS-GNN$^+$ fine-tune the model based on rank loss (Equation (9)) and updated data to ensure improved results that better align with user preferences in future searches.

Table 7: Community quality metrics

Type	Metrics	Abbr.	Function
Internal Connectivity	Internal Density [65]	ID	$\frac{m_C}{n_C(n_C-1)/2}$
	Triangle Participation Ratio [83]	TPR	$\frac{\lvert\{u \mid u \in C, T \neq \emptyset\}\rvert}{n_C}$
	Fraction over Median Degree [83]	FOMD	$\frac{\lvert\{u \mid u \in C, deg(u) > d_m\}\rvert}{n_C}$
	Average Degree [65]	AD	$\frac{2m_C}{n_C}$
External Connectivity	Expansion [65]	EP	$\frac{n_{BC}}{n_C}$
	Cut Ratio [32]	CR	$\frac{n_{BC}}{n_C(n-n_C)}$
Internal & External	Conductance [68]	CD	$\frac{n_{BC}}{2m_C+n_{BC}}$
	Normalized Cut [68]	NC	$\frac{n_{BC}}{2m_C+n_{BC}} + \frac{n_{BC}}{2(m-m_C)-n_{BC}}$
	Average Out Degree Fraction [31]	ADF	$\frac{1}{n_C}\sum_{u \in C}\frac{\lvert\{(u,v) \in E \mid v \notin C\}\rvert}{deg(u)}$
	Flake Out Degree Fraction [31]	FDF	$\frac{\lvert\{u \mid u \in C, \lvert\{(u,v) \in E \mid v \in C\}\rvert < deg(u)/2\}\rvert}{n_C}$
Network Model	Modularity [60]	MD	$\frac{1}{2n}(2m_C - \frac{(\sum_{u \in V_C} deg(u))^2}{2m})$
	Density Modularity [48]	DM	$\frac{1}{2n_C}(2m_C - \frac{(\sum_{u \in V_C} deg(u))^2}{2m})$

5 EXPERIMENTS

This section presents a series of experiments aiming at addressing the following questions:

Q1: How do the methods perform on different evaluation metrics? Is it possible for a method to achieve optimal performance in both accuracy and community metrics simultaneously? (Section 5.2)

Q2: What are the relationships between the communities identified by different methods? (Section 5.2)

Q3: If a method achieves high search effectiveness, does it also offer advantages in computational resource efficiency? (Section 5.3)

Q4: How do the techniques employed impact the overall performance of the methods? (Section 5.4)

5.1 Experimental Setting

5.1.1 *Datasets.*

The experiments are conducted on seven widely used real-world datasets, which cover a variety of graphs, including citation networks (Cora and Citeseer), collaboration networks (DBLP), social networks (LiveJournal and Orkut), product networks (Amazon), and an email network (Email-Eu). The main characteristics of these datasets are summarized in Table 6. Some methods may encounter out-of-memory (OOM) issues when processing large-scale datasets. To ensure a fair and comprehensive comparison, we perform experiments on subgraphs when the dataset contains more than 10,000 nodes (e.g., DBLP, Amazon, etc.).

5.1.2 *Compared methods.*

We selected 7 representative learning-based methods for our experiments, each with its unique characteristics: (1) ICS-GNN: subgraph candidate strategy; (2) QD-GNN: query-driven framework; (3) COCLE: contrastive learning framework for few-label scenarios; (4) COCLEP: graph partition strategy; (5) CGNP: meta-learning framework; (6) CommunityAF: generative framework; and (7) Transzero: free-label learning strategy.

5.1.3 *Evaluation metrics.*

To measure overall methods performance, we employ various metrics related to community and search performance. For evaluating communities, we use both accuracy metrics and quality metrics to assess how closely they resemble the ground-truth communities and to evaluate their community quality.

The accuracy metrics primarily include F1, Recall, Precision, NMI, ARI, and JAC. We use twelve metrics to characterize the

Table 8: Community evaluation under different metrics

Datasets	Methods	F1	Pre	Rec	NMI	ARI	JAC	ID	AD	FOMD	TPR	EP	CR	NC	CD	ADF	FDF	MD	DM
Email-Eu	ICS-GNN	0.479	0.729	0.357	0.270	0.405	0.331	0.312	15.294	0.239	0.967	6.770	2.717×10^{-2}	0.659	0.616	0.529	0.584	0.020	6.268
	QD-GNN	0.179	0.110	0.480	0.001	0.002	0.099	0.037	16.709	0.260	0.839	**1.071**	3.198×10^{-2}	0.996	0.514	0.469	0.466	0.001	0.036
	COCLE	0.585	0.726	0.567	0.382	**0.511**	0.462	0.195	15.764	0.273	0.902	4.311	1.473×10^{-2}	0.581	0.479	0.401	0.388	0.027	5.810
	COCLEP	0.432	0.404	0.587	0.190	0.301	0.302	0.145	21.581	0.394	0.951	4.181	3.147×10^{-2}	0.752	0.518	0.431	0.436	0.037	4.021
	CGNP	0.322	0.461	0.281	0.134	0.236	0.209	0.355	21.836	0.430	0.993	11.035	5.993×10^{-2}	0.863	0.717	0.619	0.718	0.019	4.294
	CommunityAF	0.337	**0.807**	0.217	0.196	0.280	0.211	**0.510**	13.185	0.098	0.970	14.962	4.993×10^{-2}	0.802	0.760	0.674	0.814	0.009	4.857
	Transzero	**0.616**	0.520	**0.829**	**0.393**	0.486	**0.488**	0.124	20.987	0.411	0.903	2.174	**1.457×10^{-2}**	**0.459**	**0.350**	**0.256**	**0.138**	**0.080**	**6.758**
	k-core	0.118	0.077	0.344	0.023	0.037	0.064	0.373	50.811	**0.974**	**1.000**	5.326	4.875×10^{-2}	0.840	0.406	0.382	0.347	0.034	4.230
	k-truss	0.116	0.074	0.369	0.020	0.033	0.063	0.332	**51.572**	0.970	**1.000**	4.573	4.409×10^{-2}	0.841	0.373	0.353	0.274	0.034	3.797
Cora	ICS-GNN	0.681	0.909	0.544	0.421	0.575	0.517	0.017	4.173	0.400	0.677	1.013	5.864×10^{-4}	0.294	0.254	0.180	0.089	0.081	**1.711**
	QD-GNN	**0.972**	**1.000**	0.945	**0.905**	**0.956**	**0.945**	0.011	4.142	0.410	0.658	0.271	1.707×10^{-4}	0.105	0.087	0.066	0.020	0.126	1.686
	COCLE	0.864	0.777	**0.976**	0.659	0.780	0.762	0.009	4.154	0.420	0.625	0.366	2.460×10^{-4}	0.148	0.116	0.068	0.043	0.143	1.574
	COCLEP	0.805	0.715	0.931	0.545	0.687	0.678	0.007	4.017	0.425	0.605	**0.149**	**9.765×10^{-5}**	**0.064**	**0.049**	**0.044**	**0.002**	**0.160**	1.537
	CGNP	0.659	0.724	0.606	0.410	0.555	0.537	0.013	3.963	0.363	0.649	1.126	6.660×10^{-4}	0.331	0.274	0.219	0.141	0.094	1.500
	CommunityAF	0.188	0.912	0.114	0.104	0.139	0.112	**0.099**	3.495	0.318	0.541	5.734	2.437×10^{-3}	0.638	0.615	0.269	0.225	0.016	1.565
	Transzero	0.835	0.757	0.931	0.586	0.738	0.717	0.008	3.947	0.397	0.591	0.265	1.597×10^{-4}	0.103	0.082	0.082	0.040	0.148	1.524
	k-core	0.265	0.197	0.621	0.007	-0.002	0.155	0.021	**4.829**	**0.575**	0.751	0.550	4.833×10^{-4}	0.432	0.110	0.075	0.014	0.074	0.367
	k-truss	0.277	0.340	0.631	0.029	0.029	0.166	0.084	4.314	0.474	**0.799**	0.558	3.511×10^{-4}	0.204	0.124	0.095	0.029	0.074	0.701
Citeseer	ICS-GNN	0.459	0.816	0.320	0.204	0.319	0.299	0.018	4.378	0.538	0.497	0.714	3.475×10^{-4}	0.230	0.195	0.163	0.084	0.097	**1.757**
	QD-GNN	**0.766**	**1.000**	0.633	**0.567**	**0.662**	**0.633**	0.009	3.821	0.480	0.430	0.342	2.355×10^{-4}	0.190	0.149	0.139	0.040	0.131	1.416
	COCLE	0.678	0.591	**0.864**	0.343	0.430	0.519	0.006	3.962	0.518	0.439	0.128	6.598×10^{-5}	0.058	0.041	0.038	0.008	0.195	1.365
	COCLEP	0.629	0.611	0.652	0.245	0.418	0.459	0.006	3.916	0.511	0.448	**0.062**	**2.750×10^{-5}**	**0.026**	**0.019**	**0.014**	**0.000**	**0.198**	1.365
	CGNP	0.537	0.775	0.414	0.249	0.386	0.374	0.015	4.505	0.560	0.489	0.643	3.341×10^{-4}	0.207	0.169	0.150	0.091	0.120	1.708
	CommunityAF	0.255	0.811	0.162	0.109	0.169	0.154	**0.051**	3.759	0.522	0.473	1.600	7.792×10^{-4}	0.396	0.371	0.204	0.139	0.046	1.644
	Transzero	0.656	0.725	0.603	0.304	0.486	0.491	0.007	3.925	0.501	0.424	0.151	7.708×10^{-5}	0.065	0.048	0.048	0.016	0.168	1.448
	k-core	0.267	0.327	0.419	0.027	0.033	0.159	0.043	**5.181**	**0.731**	**0.675**	0.889	4.501×10^{-4}	0.222	0.125	0.098	0.032	0.142	1.234
	k-truss	0.230	0.209	0.475	0.018	0.006	0.138	0.029	3.746	0.532	0.499	0.371	1.457×10^{-4}	0.056	0.052	0.042	0.024	0.142	0.612
Amazon(9k)	ICS-GNN	0.705	0.982	0.551	0.553	0.679	0.546	0.016	3.979	0.400	0.644	0.405	7.092×10^{-5}	0.148	0.143	0.100	0.060	0.025	1.917
	QD-GNN	0.012	**1.000**	0.006	0.009	0.011	0.006	**0.451**	1.677	0.017	0.155	2.283	3.233×10^{-4}	0.621	0.620	0.537	0.541	0.000	0.837
	COCLE	0.147	0.997	0.083	0.108	0.136	0.082	0.149	3.814	0.367	0.663	1.083	1.540×10^{-4}	0.272	0.271	0.186	0.144	0.004	1.889
	COCLEP	**0.852**	0.815	**0.898**	**0.724**	**0.833**	**0.767**	0.009	3.907	0.391	0.629	**0.017**	**2.127×10^{-6}**	**0.005**	**0.005**	**0.004**	**0.000**	**0.044**	1.857
	CGNP	0.762	0.863	0.716	0.646	0.745	0.676	0.015	3.958	0.384	0.642	0.263	3.983×10^{-5}	0.078	0.076	0.067	0.039	0.032	1.901
	CommunityAF	0.460	0.994	0.306	0.343	0.434	0.306	0.036	4.135	0.409	0.638	0.500	7.096×10^{-5}	0.123	0.121	0.061	0.034	0.014	**2.026**
	Transzero	0.295	0.996	0.202	0.225	0.279	0.201	0.109	3.822	0.354	0.620	0.301	3.786×10^{-5}	0.091	0.091	0.092	0.045	0.011	1.881
	k-core	0.168	0.280	0.641	0.062	0.103	0.099	0.059	**4.760**	**0.604**	**0.847**	0.485	1.024×10^{-4}	0.309	0.100	0.072	0.029	0.029	0.998
	k-truss	0.145	0.400	0.668	0.060	0.084	0.085	0.114	4.525	0.525	0.791	0.322	4.747×10^{-5}	0.066	0.065	0.049	0.018	0.006	1.033
DBLP(5K)	ICS-GNN	0.325	**0.616**	0.221	0.148	0.271	0.198	0.042	6.213	0.453	0.851	1.617	4.476×10^{-4}	0.286	0.274	0.207	0.175	0.030	2.920
	QD-GNN	0.086	0.467	0.106	0.012	0.023	0.046	0.103	3.392	0.219	0.640	1.658	9.903×10^{-4}	0.679	0.568	0.509	0.516	0.033	1.101
	COCLE	0.262	0.279	0.420	0.080	0.152	0.160	0.090	4.990	0.360	0.796	1.623	6.292×10^{-4}	0.440	0.360	0.274	0.215	0.039	1.943
	COCLEP	0.384	0.378	0.425	0.128	0.283	0.242	0.014	6.089	0.455	0.890	**0.264**	**8.129×10^{-5}**	**0.066**	**0.059**	**0.046**	**0.006**	0.091	2.675
	CGNP	0.311	0.330	0.306	0.104	0.226	0.196	0.020	5.837	0.429	0.850	1.733	5.383×10^{-4}	0.341	0.308	0.245	0.211	0.061	2.483
	CommunityAF	0.117	0.445	0.070	0.057	0.085	0.066	0.100	5.728	0.484	0.884	1.018	2.611×10^{-4}	0.197	0.193	0.121	0.091	0.016	2.787
	Transzero	**0.428**	0.330	**0.614**	**0.170**	**0.306**	**0.282**	0.007	5.648	0.397	0.863	0.280	9.635×10^{-5}	0.082	0.068	0.066	0.024	**0.125**	2.324
	k-core	0.141	0.229	0.428	0.019	0.031	0.076	0.115	**10.095**	**0.791**	**0.990**	1.067	4.372×10^{-4}	0.405	0.117	0.096	0.027	0.075	2.640
	k-truss	0.159	0.417	0.407	0.043	0.067	0.089	**0.209**	9.273	0.719	0.976	0.912	2.852×10^{-4}	0.143	0.102	0.080	0.028	0.062	**3.283**
Orkut(5K)	ICS-GNN	0.625	0.969	0.462	0.436	0.561	0.456	0.080	20.028	0.690	0.954	1.545	1.170×10^{-3}	0.246	0.225	0.213	0.091	0.057	**8.778**
	QD-GNN	0.161	0.126	0.378	0.001	0.004	0.088	0.004	6.904	0.198	0.757	1.400	2.810×10^{-3}	0.971	0.573	0.543	0.623	0.007	0.139
	COCLE	0.467	0.372	0.779	0.200	0.291	0.314	0.017	12.353	0.427	0.865	1.051	8.660×10^{-4}	0.284	0.188	0.193	0.125	0.102	4.261
	COCLEP	**0.914**	0.892	**0.963**	**0.814**	0.882	**0.862**	0.031	17.435	0.550	0.893	**0.260**	**8.742×10^{-5}**	**0.031**	**0.027**	**0.036**	**0.009**	**0.113**	7.310
	CGNP	0.662	0.747	0.612	0.450	0.599	0.539	0.050	17.457	0.605	0.922	1.789	1.098×10^{-3}	0.247	0.217	0.204	0.146	0.081	7.177
	CommunityAF	0.203	**0.989**	0.115	0.135	0.168	0.115	**0.241**	12.978	0.492	0.939	5.196	2.622×10^{-3}	0.465	0.456	0.370	0.291	0.011	6.192
	Transzero	0.911	0.894	0.948	0.792	**0.885**	0.849	0.032	16.560	0.529	0.869	0.417	1.362×10^{-4}	0.047	0.042	0.068	0.037	0.104	7.046
	k-core	0.208	0.200	0.466	0.042	0.061	0.120	0.038	**25.204**	**0.858**	**0.996**	1.054	1.155×10^{-3}	0.412	0.125	0.125	0.020	0.103	4.471
	k-truss	0.246	0.269	0.512	0.071	0.095	0.150	0.052	24.246	0.833	0.993	1.002	9.813×10^{-4}	0.384	0.118	0.119	0.023	0.096	4.492
LiveJournal(5K)	ICS-GNN	0.687	0.952	0.538	0.486	0.630	0.524	0.034	8.366	0.522	0.767	1.000	4.552×10^{-4}	0.211	0.202	0.181	0.083	0.033	3.926
	QD-GNN	0.163	0.468	0.144	0.041	0.094	0.094	0.059	3.334	0.172	0.457	2.328	1.515×10^{-3}	0.709	0.666	0.607	0.708	0.016	1.351
	COCLE	0.315	0.307	0.330	0.080	0.209	0.193	0.042	5.868	0.352	0.675	1.439	6.523×10^{-4}	0.345	0.337	0.288	0.217	0.019	2.800
	COCLEP	0.865	0.810	0.952	0.710	0.820	0.779	0.014	7.446	0.430	0.702	**0.147**	**4.419×10^{-5}**	**0.028**	**0.026**	**0.026**	**0.003**	0.063	3.423
	CGNP	0.521	0.589	0.493	0.328	0.448	0.406	0.025	7.674	0.472	0.740	1.296	6.620×10^{-4}	0.292	0.273	0.234	0.161	0.043	3.470
	CommunityAF	0.223	**0.958**	0.135	0.148	0.191	0.134	0.121	5.853	0.379	0.706	3.759	1.642×10^{-3}	0.557	0.550	0.469	0.478	0.007	2.818
	Transzero	**0.939**	0.912	**0.967**	**0.827**	**0.919**	**0.885**	0.016	7.392	0.429	0.685	0.170	4.935×10^{-5}	0.032	0.030	0.037	0.011	0.054	3.438
	k-core	0.292	0.345	0.566	0.143	0.183	0.203	0.079	**27.286**	**0.807**	**0.938**	0.828	1.029×10^{-3}	0.394	0.092	0.109	0.016	**0.066**	**5.305**
	k-truss	0.279	0.430	0.599	0.137	0.171	0.193	**0.173**	23.391	0.724	0.914	1.366	1.089×10^{-3}	0.292	0.133	0.138	0.081	0.057	4.832

community quality of C from four aspects as listed in Table 7: quantify community quality, such as internal connectivity, external connectivity, both internal and external connectivity, and network model [8, 83]. Given a graph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges, a community $C = (V_C, E_C)$ is a subgraph of G with $n_C = |V_C|$ nodes and $m_C = |V_E|$ where $E_C = \{(u, v) \in G | u \in V_C, v \in V_C\}$. The number of edges on the boundary of community C can be represented as $n_{BC} = |E_{GC}| = |\{(u, v) \in E | u \in V_C, v \notin V_C\}|$. Note that $T = \{(v, w) | v, w \in C, (u, v) \in E, (u, w) \in E, (v, w) \in E\}$, d_m is the median value of $d(u)$ in V.

For search evaluation, we focus on three aspects: search effectiveness and search cost. Search effectiveness is measured using the aforementioned metrics, search cost is assessed along two aspects: (1) Training phase: The time required for model training, GPU resources (memory overhead), and (2) Search phase: the time consumption to complete a single query, and the GPU resources.

5.2 Community Evaluation

We evaluated the identified communities using various metrics. For all methods, we employed 50 query nodes to assess the communities identified by these approaches.

5.2.1 Community Evaluation Across Multiple Metrics. We employ 18 metrics, including accuracy and community quality metrics, to measure the communities identified by 9 methods (including 2 non-learning methods k-core [4] and k-truss [17]). The results are shown in Table 8. Learning-based methods demonstrate a significant advantage over non-learning-based methods in accuracy metrics (e.g., F1 and NMI). However, non-learning-based methods achieve better performance in community quality metrics (e.g., AD, FOMD, and TPR) due to predefined subgraph metrics. Moreover, for these learning-based methods, we observe that COCLE, COCLEP, and Transzero perform well on F1 and several quality metrics

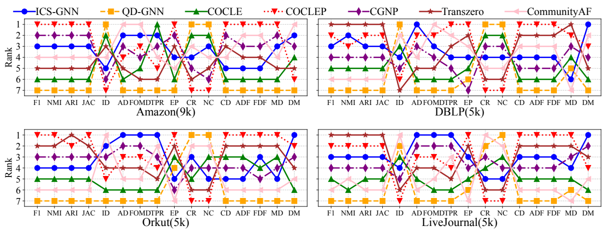


Figure 5: Ranking changing across metrics

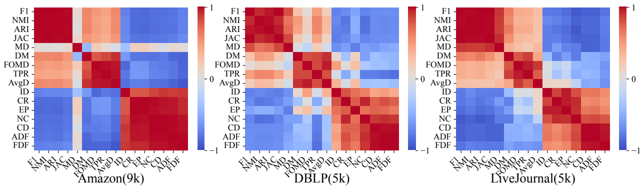


Figure 6: Metric correlation analysis

across most datasets. Despite using fewer labeled samples or none, the introduction of contrastive learning allows the models to learn additional information, thereby improving method performance.

We visually compare the changes in rankings of learning-based methods across various metrics, as illustrated in Figure 5. Notably, methods that rank highly on accuracy metrics (e.g., F1-score) do not achieve high rankings on community metrics, particularly those measuring internal connectivity, such as TPR. We also observe that most methods exhibit consistent rankings across metrics AD, FOMD, and TPR (or CD, ADF, and FDF), suggesting that these metrics capture similar community properties. It is interesting to note that ID, TPR, FOMD, and AD are all quality metrics measuring internal connectivity, but ID does not exhibit ranking consistency. A similar phenomenon is observed with EP and CR as well.

5.2.2 Metric Correlation Analysis. To better analyze the relationships between metrics and select appropriate metrics for evaluation, we explored the correlations among different metrics on four datasets using Pearson correlation coefficients. The results are presented in the heatmaps as shown in Figure 6.

Figure 6 presents the correlation calculations among all metrics. It is evident that there is a strong correlation among the accuracy metrics, while the correlation between the accuracy metrics and community quality metrics is relatively weak. We can observe that accuracy metrics and community metrics evaluate different aspects of communities. This suggests that incorporating community metrics into the evaluation process is reasonable, thereby enriching the methods for measuring communities.

For community quality metrics, we observed that metrics based on internal connectivity exhibit strong positive correlations with each other on most datasets, while those related to external connectivity also show relatively strong positive correlations within their group. We can select representative metrics from those with strong positive correlations to evaluate specific community quality (e.g., internal cohesiveness). In this paper, we recommend TPR, EP, DM, and CD as the representative community quality metrics.

5.2.3 Relationships between Identified Communities. We explore the relationships between the communities identified by different methods for the same query. Figure 7 (first row) shows the ratio of the intersection of methods' communities to their union, there is relatively little overlap between the communities identified by different methods. Additionally, we analyze the proportion of nodes in the intersection that belong to the ground truth. Figure 7 (last row) shows a significant proportion of the intersection nodes that belong to the ground truth, suggesting the existence of key community nodes that are more likely to be identified by different methods.

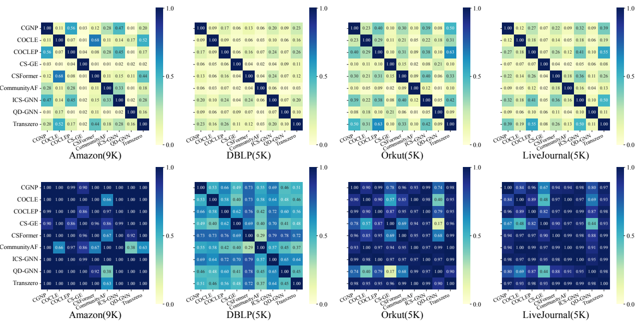


Figure 7: Connection between identified communities

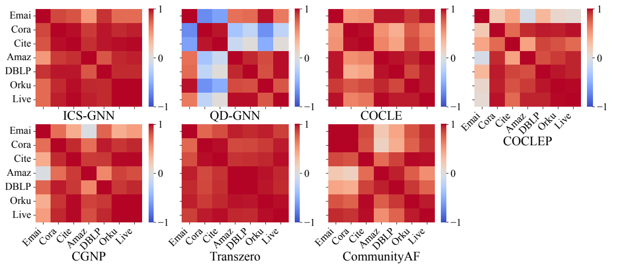


Figure 8: Correlation between methods' effectiveness across different datasets

5.2.4 Relationships between Methods and Datasets. We analyze the correlations between methods' effectiveness (including F1, NMI, ARI, JAC, TPR, EP, CD, and DM) across different datasets, as illustrated in Figure 8. ICS-GNN and Transzero exhibit consistently strong positive correlations across nearly all datasets (highlighted in red), indicating their robustness and insensitivity to dataset variations. In contrast, QD-GNN and COCLE show strong correlations between the Cora and Citeseer datasets, demonstrating better performance on these datasets and indicating a preference for specific datasets. Notably, Cora and Citeseer have lower average degrees than other datasets and are commonly used for node classification.

5.3 Overhead Evaluation of Methods

This section will conduct a comprehensive analysis of the methods' overhead, considering both the training phase and the search phase.

5.3.1 Training Phase Analysis. During the training phase, we focus on model's initialization parameters, GPU usage, and epoch time. **Model Parameters**. The number of model parameters generally determines the model's expressive capacity. We statistically analyze the learnable parameters of the models, as illustrated in Figure 9. The hyperparameters for the models are set to their respective default values, primarily regarding the number of layers, input

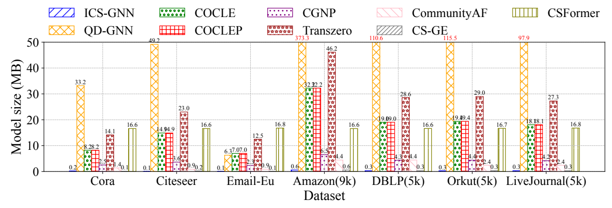


Figure 9: Model initialization parameters

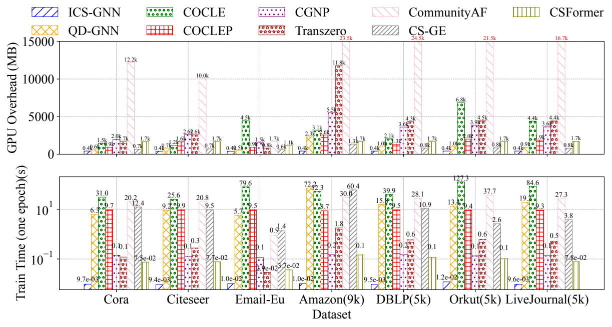


Figure 10: GPU consumption and training time

dimensions, and hidden dimensions. The number of layers and hidden dimensions are often configured manually and are typically positively correlated with the model parameters. In Figure 9, the significantly lower parameter of ICS-GNN and GS-GE compared to other models can be attributed to their smaller hidden dimension setting. The input dimension is determined by the initial features of the nodes, and it is positively correlated with the model size.

Training GPU Usage. Figure 10 (top) illustrates the time required for different methods to train for one epoch on seven datasets during the training process. As evident from the propagation formula of GNNs, both the adjacency matrix and input features occupy a significant amount of space. Consequently, the scale of the dataset (determines the size of the adjacency matrix and the number of input features) and the input dimension can greatly influence GPU memory usage. The figure illustrates that in the largest dataset, Amazon (9k), most methods exhibit the highest GPU usage.

Training Time. Figure 10 (bottom) illustrates the GPU memory usage of different methods during the training process on 7 datasets. ICS-GNN achieves a very short training time for each epoch due to its minimal model parameters and candidate subgraph strategy. This efficiency allows it to perform online training and searching.

Comparison of Training Costs. We visually compare the effectiveness of different methods and the associated costs during the training phase through ranking shifts, as shown in Figure 12 (top). Except for ICS-GNN, no other method demonstrates consistency across Model, GPU, and Time. Moreover, very few methods can achieve good effectiveness while gaining advantages in the training.

5.3.2 *Search Phase Analysis.* During the search phase, we primarily analyze GPU usage and the search time required for a single query.

Search GPU Usage. Figure 11 (top) illustrates the GPU memory usage of different methods during the training process on 7 datasets. GPU usage during the search phase mainly depends on model parameters and input feature size. CGNP shows higher GPU consumption than other methods, likely due to its meta-learning strategy,

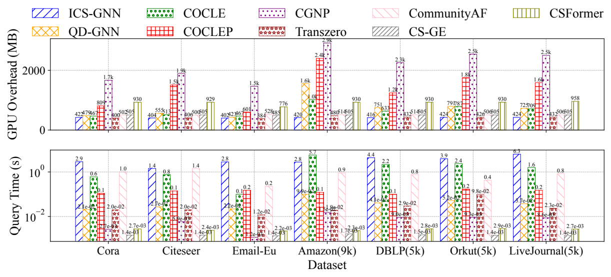


Figure 11: Query time and GPU consumption

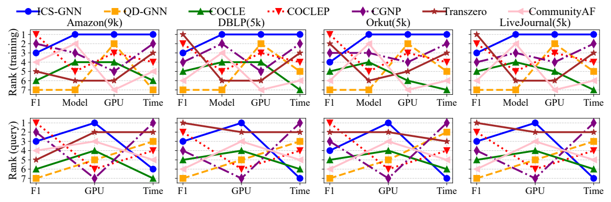


Figure 12: Ranking changing at training and query phases

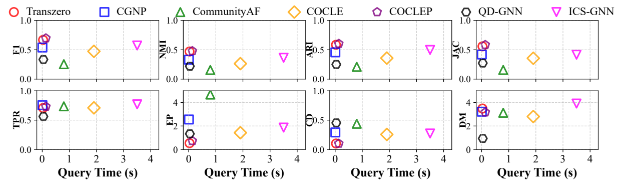


Figure 13: The tradeoff between query time and effectiveness

which requires preliminary training before searching. Moreover, ICS-GNN benefits from the candidate subgraph strategy, maintaining stable memory consumption across different datasets.

Search Time. Figure 11 (bottom) illustrates the time required for different methods to train for one epoch on seven datasets during the training process. COCLEP achieves faster search speeds than COCLE on most datasets, indicating that graph partitioning enhances search efficiency. On the other hand, GS-GE benefits from the proximity graph, resulting in the least time consumption among all methods. In contrast, ICS-GNN consumes more search time on most datasets, likely due to the iterative vertex swapping required during the search process to optimize the identified community.

Comparison of Search Costs. We visually compare the effectiveness of different methods and the associated costs during the search phase using ranking shifts, as shown in Figure 12 (bottom). It is observed that there is no consistency between GPU usage and search time; most methods that demonstrate good search performance tend to require more search time.

Tradeoff between Query Time and Effectiveness. We visually illustrate the tradeoff between query time and effectiveness, as shown in Figure 13. It shows that Transzero achieves an impressive balance between effectiveness and efficiency, whereas ICS-GNN demands more time due to its online training.

5.4 Techniques Influencing Performance

The techniques of graph sampling methods and training paradigms play a critical role in designing learning-based community search

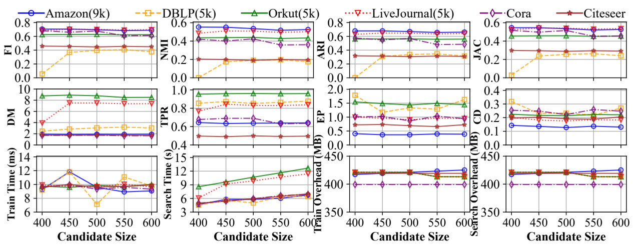


Figure 14: The effect of candidate size (ICS-GNN)

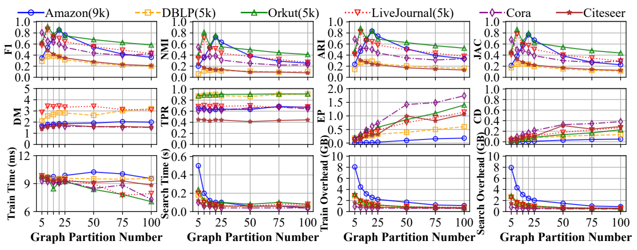


Figure 15: The effect of graph partition number (COCLEP)

techniques (as discussed in Section 4). In this section, we analyze the impact of these techniques on the performance of the methods.

5.4.1 The Effect of Candidate Subgraph. As shown in Figure 14, we investigate the effect of candidate subgraph size on the performance of ICS-GNN. For community evaluation, its accuracy decreases with increasing candidate size on some datasets, while the community quality shows no clear patterns of change. Regarding method overhead, training time, and memory consumption during training and searching, do not exhibit noticeable trends. This might be attributed to ICS-GNN's lightweight model and relatively small candidate subgraphs. However, search time increases with the increase of candidate subgraph size. This is likely due to the expanded search space, which requires more time for the greedy search.

5.4.2 The Effect of Graph Partitioning. We analyze the effect of graph partition number on method performance, as shown in Figure 15. For community evaluation, the results reveal that accuracy metrics initially improve but then decline as the partition number increases, indicating that a higher partition count does not necessarily yield better performance. Notably, DM and TPR exhibit relative insensitivity to changes in partition number, whereas EP and CD show a gradual increase with higher partition counts. This trend may be attributed to the disruption of graph structure caused by graph partitioning, which affects EP and CD to a greater extent. In terms of method overhead, training time per epoch, search time per query, and memory consumption during training and search all decrease as the partition number increases.

5.4.3 The Effect of Contrastive Learning. We present the relationship between the number of labeled samples required by different methods and their performance metrics (averaged on all datasets) in Figure 16. As shown, under comparable amounts of labeled data, contrastive learning-based methods such as Transzero and COCLEP demonstrate strong performance across most metrics, indicating that contrastive learning enables models to capture useful information beyond what is provided by the labels.

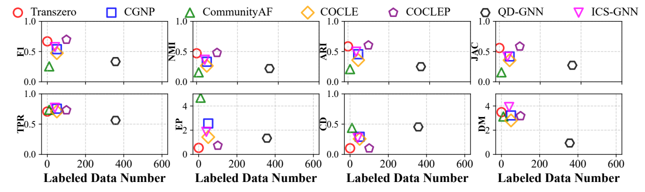


Figure 16: The effect of contrastive learning

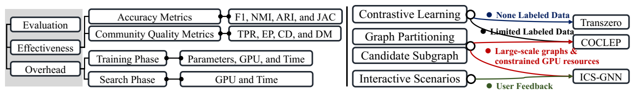


Figure 17: Guidance on method evaluation and selection

6 RECOMMENDATION

Based on the experimental results, we provide the following recommendations for method evaluation and method selection:

Method Evaluation. To comprehensively evaluate learning-based CS methods, we recommend evaluating these methods from two perspectives: effectiveness and overhead, as shown in Figure 17 (left). The effectiveness of the method is evaluated from two perspectives: (1) community quality metrics, including the selected TPR, EP, DM, and CD; and (2) accuracy metrics, including F1 score, NMI, ARI, and JAC, which assess the alignment with ground-truth communities. Overhead evaluation considers resource consumption in both training and search phases. In the training phase, key factors include model parameters, GPU consumption, and training time, while in the search phase, GPU consumption and query search time are rigorously assessed.

Method Selection. According to the experimental results, we find that method selection highly depends on specific conditions, as shown in Figure 17 (right): 1) Limited labeled data: COCLEP and Transzero, leveraging contrastive learning, effectively capture meaningful representations without extensive supervision; 2) Large-scale graphs or constrained computational resources: COCLEP (via graph partitioning) and ICS-GNN (via candidate subgraphs) reduce graph size, thereby lowering the computational costs required for training and inference; 3) Interactive scenarios: ICS-GNN dynamically adjusts candidate subgraphs based on user feedback, providing results more aligned with user interests.

7 CONCLUSIONS

In this paper, we present a comprehensive survey and experimental study of learning-based CS methods, highlighting emerging trends and proposing a unified pipeline. Furthermore, we address the limitations of relying solely on accuracy metrics by including a comprehensive evaluation to measure community holistically. We empirically study the performance and overhead of different methods under various metrics, investigating correlations among metrics and exploring the effects of commonly used techniques.

ACKNOWLEDGMENTS

This work was substantially supported Key Projects of the National NSF of China (U23A20496), National NSF of China (62072149), and the Primary R&D Plan of Zhejiang (2023C03198,2024C01020).

REFERENCES

[1] Esra Akbas and Peixiang Zhao. 2017. Truss-based Community Search: a Truss-equivalence Based Indexing Approach. *PVLDB* 10, 11 (2017), 1298–1309.

[2] Uri Alon and Eran Yahav. 2020. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205* (2020).

[3] Nicola Barbieri, Francesco Bonchi, Edoardo Galimberti, and Francesco Gullo. 2015. Efficient and Effective Community Search. *Data Min. Knowl. Discov.* 29, 5 (2015), 1406–1433.

[4] Vladimir Batagelj and Matjaz Zaversnik. 2003. An O (m) Algorithm for Cores Decomposition of Networks. *arXiv* cs.DS/0310049 (2003).

[5] Francesco Bonchi, Arijit Khan, and Lorenzo Severini. 2019. Distance-generalized Core Decomposition. In *SIGMOD*. 1006–1023.

[6] Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. 2016. *Recent advances in graph partitioning*. Springer.

[7] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning* (Corvalis, Oregon, USA) *(ICML '07)*. Association for Computing Machinery, New York, NY, USA, 129–136. https://doi.org/10.1145/1273496.1273513

[8] Tanmoy Chakraborty, Ayushi Dalmia, Animesh Mukherjee, and Niloy Ganguly. 2017. Metrics for Community Analysis: A Survey. *ACM Comput. Surv.* 50, 4, Article 54 (Aug. 2017), 37 pages. https://doi.org/10.1145/3091106

[9] Lijun Chang and Lu Qin. 2019. Cohesive Subgraph Computation Over Large Sparse Graphs. In *ICDE*. 2068–2071.

[10] 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.

[11] Jiazun Chen, Jun Gao, and Bin Cui. 2022. ICS-GNN+: lightweight interactive community search via graph neural network. *The VLDB Journal* 32, 2 (July 2022), 447–467. https://doi.org/10.1007/s00778-022-00754-0

[12] Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. 2023. NAGphormer: A Tokenized Graph Transformer for Node Classification in Large Graphs. In *The Eleventh International Conference on Learning Representations*.

[13] Jiazun Chen, Yikuan Xia, and Jun Gao. 2023. CommunityAF: An Example-based Community Search Method via Autoregressive Flow. *Proc. VLDB Endow.* 16, 10 (2023), 2565–2577. https://doi.org/10.14778/3603581.3603595

[14] Jiazun Chen, Yikuan Xia, and Jun Gao. 2023. CommunityAF: An Example-Based Community Search Method via Autoregressive Flow. *Proc. VLDB Endow.* 16, 10 (June 2023), 2565–2577. https://doi.org/10.14778/3603581.3603595

[15] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, Article 149, 11 pages.

[16] Zi Chen, Yiwei Zhao, Long Yuan, Xuemin Lin, and Kai Wang. 2023. Index-Based Biclique Percolation Communities Search on Bipartite Graphs. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 2699–2712. https://doi.org/10.1109/ICDE55515.2023.00207

[17] Jonathan Cohen. 2008. Trusses: Cohesive subgraphs for social network analysis. *National security agency technical report* 16, 3.1 (2008), 1–29.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

[20] Joel Dudley, Tarangini Deshpande, and Atul J. Butte. 2011. Exploiting Drug-disease Relationships for Computational Drug Repositioning. *Briefings Bioinform.* 12, 4 (2011), 303–311.

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

[22] Shuheng Fang, Kangfei Zhao, Guanghua Li, and Jeffrey Xu Yu. 2023. Community Search: A Meta-Learning Approach. In *ICDE*. 2358–2371.

[23] Shuheng Fang, Kangfei Zhao, Yu Rong, Zhixun Li, and Jeffrey Xu Yu. 2024. Inductive Attributed Community Search: To Learn Communities Across Graphs. *Proc. VLDB Endow.* 17, 10 (Aug. 2024), 2576–2589. https://doi.org/10.14778/3675034.3675048

[24] Yixiang Fang, Reynold Cheng, Xiaodong Li, Siqiang Luo, and Jiafeng Hu. 2017. Effective Community Search over Large Spatial Graphs. *PVLDB* 10, 6 (2017), 709–720.

[25] Yixiang Fang, Reynold Cheng, Siqiang Luo, and Jiafeng Hu. 2016. Effective Community Search for Large Attributed Graphs. *PVLDB* 9, 12 (2016), 1233–1244.

[26] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. 2019. A survey of community search over big graphs. *The VLDB Journal* 29, 1 (July 2019), 353–392. https://doi.org/10.1007/s00778-019-00556-x

[27] Yixiang Fang, Kai Wang, Xuemin Lin, and Wenjie Zhang. 2021. Cohesive Subgraph Search over Big Heterogeneous Information Networks: Applications, Challenges, and Solutions. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 2829–2838. https://doi.org/10.1145/3448016.3457538

[28] Yixiang Fang, Zhongran Wang, Reynold Cheng, Hongzhi Wang, and Jiafeng Hu. 2019. Effective and Efficient Community Search Over Large Directed Graphs. *IEEE Trans. Knowl. Data Eng.* 31, 11 (2019), 2093–2107.

[29] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. 2020. Effective and Efficient Community Search over Large Heterogeneous Information Networks. *PVLDB* 13, 6 (2020), 854–867.

[30] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*. PMLR, 1126–1135.

[31] Gary William Flake, Steve Lawrence, and C Lee Giles. 2000. Efficient identification of web communities. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 150–160.

[32] Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3 (2010), 75–174. https://doi.org/10.1016/j.physrep.2009.11.002

[33] Jun Gao, Jiazun Chen, Zhao Li, and Ji Zhang. 2021. ICS-GNN: Lightweight Interactive Community Search via Graph Neural Network. *PVLDB* 14, 6 (2021), 1006–1018.

[34] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. 2018. Conditional Neural Processes. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Jennifer Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 1704–1713. https://proceedings.mlr.press/v80/garnelo18a.html

[35] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (Eds.), Vol. 27. Curran Associates, Inc.

[36] Xiaoxuan Gou, Xiaoliag Xu, Xiangying Wu, Runhuai Chen, Yuxiang Wang, Tianxing Wu, and Xiangyu Ke. 2023. Effective and Efficient Community Search with Graph Embeddings. In *ECAI*.

[37] 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.

[38] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 855–864. https://doi.org/10.1145/2939672.2939754

[39] Peng Han, Silin Zhou, Jie Yu, Zichen Xu, Lisi Chen, and Shuo Shang. 2023. Personalized Re-ranking for Recommendation with Mask Pretraining. *Data Science and Engineering* 8, 4 (2023), 357–367.

[40] 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

[41] 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* (Austin, TX, USA) *(WWW '23 Companion)*. Association for Computing Machinery, New York, NY, USA, 1196–1203. https://doi.org/10.1145/3543873.3587654

[42] Allen L Hu and Keith CC Chan. 2013. Utilizing both topological and attribute information for protein complex identification in PPI networks. *IEEE/ACM transactions on computational biology and bioinformatics* 10, 3 (2013), 780–792.

[43] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. 2014. Querying K-truss Community in Large and Dynamic Graphs. In *SIGMOD*. 1311–1322.

[44] Xin Huang and Laks V. S. Lakshmanan. 2017. Attribute-Driven Community Search. *PVLDB* 10, 9 (2017), 949–960.

[45] Xin Huang, Laks V. S. Lakshmanan, Jeffrey Xu Yu, and Hong Cheng. 2015. Approximate Closest Community Search in Networks. *PVLDB* 9, 4 (2015), 276–287.

[46] 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. *Proc. VLDB Endow.* 15, 6 (Feb. 2022), 1243–1255. https://doi.org/10.14778/3514061.3514070

[47] George Karypis. 1997. METIS: Unstructured graph partitioning and sparse matrix ordering system. *Technical report* (1997).

[48] 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* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 889–903. https://doi.org/10.1145/3514221.3526137

[49] Diederik P. Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. *CoRR* abs/1312.6114 (2013).

[50] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

[51] Jon M. Kleinberg. 2000. Navigation in A Small World. *Nature* 406 (2000), 845–845.

[52] Ling Li, Siqiang Luo, Yuhai Zhao, Caihua Shan, Zhengkui Wang, and Lu Qin. 2023. COCLEP: Contrastive Learning-based Semi-Supervised Community Search. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. 2483–2495. https://doi.org/10.1109/ICDE55515.2023.00191

[53] Yuan Li, Xiuxu Chen, Yuhai Zhao, Wen Shan, Zhengkui Wang, Guoli Yang, and Guoren Wang. 2024. Self-Training GNN-based Community Search in Large Attributed Heterogeneous Information Networks. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 2765–2778. https://doi.org/10.1109/ICDE60146.2024.00216

[54] Qing Liu, Xuankun Liao, Xin Huang, Jianliang Xu, and Yunjun Gao. 2023. Distributed (α, β)-Core Decomposition over Bipartite Graphs. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 909–921. https://doi.org/10.1109/ICDE55515.2023.00075

[55] Qing Liu, Minjun Zhao, Xin Huang, Jianliang Xu, and Yunjun Gao. 2020. Truss-based Community Search over Large Directed Graphs. In *SIGMOD*. 2183–2197.

[56] Qing Liu, Yifan Zhu, Minjun Zhao, Xin Huang, Jianliang Xu, and Yunjun Gao. 2020. VAC: Vertex-Centric Attributed Community Search. In *ICDE*. 937–948.

[57] Wensheng Luo, Xu Zhou, Kenli Li, Yunjun Gao, and Keqin Li. 2023. Efficient Influential Community Search in Large Uncertain Graphs. *IEEE Trans. Knowl. Data Eng.* 35, 4 (2023), 3779–3793. https://doi.org/10.1109/TKDE.2021.3131611

[58] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836.

[59] Xiaoye Miao, Yue Liu, Lu Chen, Yunjun Gao, and Jianwei Yin. 2022. Reliable Community Search on Uncertain Graphs. In *ICDE*. 1166–1179.

[60] Mark EJ Newman. 2006. Modularity and community structure in networks. *Proceedings of the national academy of sciences* 103, 23 (2006), 8577–8582.

[61] A Nichol. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* (2018).

[62] Jeff Z. Pan, Elspeth Edelstein, Patrik Bansky, and Adam Wyner. 2021. A Knowledge Graph Based Approach to Social Science Surveys. *Data Intell.* 3, 4 (2021), 477–506. https://doi.org/10.1162/dint_a_00107

[63] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD '14)*. Association for Computing Machinery, New York, NY, USA, 701–710. https://doi.org/10.1145/2623330.2623732

[64] Alec Radford and Karthik Narasimhan. 2018. Improving Language Understanding by Generative Pre-Training. https://api.semanticscholar.org/CorpusID:49313245

[65] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. 2004. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences* 101, 9 (2004), 2658–2663. https://doi.org/10.1073/pnas.0400054101

[66] Danilo Rezende and Shakir Mohamed. 2015. Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Francis Bach and David Blei (Eds.), Vol. 37. PMLR, Lille, France, 1530–1538.

[67] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. 2018. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960* (2018).

[68] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence* 22, 8 (2000), 888–905.

[69] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems* 30 (2017).

[70] Longxu Sun, Xin Huang, Ronghua Li, Byron Choi, and Jianliang Xu. 2020. Index-based Intimate-Core Community Search in Large Weighted Graphs. *IEEE Trans. Knowl. Data Eng.* (2020).

[71] Sebastian Thrun and Lorien Pratt. 1998. *Learning to Learn: Introduction and Overview*. Springer US, Boston, MA, 3–17. https://doi.org/10.1007/978-1-4615-5529-2_1

[72] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *CoRR* abs/1807.03748 (2018). arXiv:1807.03748 http://arxiv.org/abs/1807.03748

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

[74] Jianwei Wang, Kai Wang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. 2024. Efficient Unsupervised Community Search with Pre-Trained Graph Transformer. *Proc. VLDB Endow.* 17, 9 (Aug. 2024), 2227–2240. https://doi.org/10.14778/3665844.3665853

[75] Jianwei Wang, Kai Wang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. 2024. Neural Attributed Community Search at Billion Scale. *Proc. ACM Manag. Data* 1, 4, Article 251 (April 2024), 25 pages. https://doi.org/10.1145/3626738

[76] 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 *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 1680–1690. https://doi.org/10.1145/3626772.3657771

[77] Yuxiang Wang, Jun Liu, Xiaoliang Xu, Xiangyu Ke, Tianxing Wu, and Xiaoxuan Gou. 2023. Efficient and Effective Academic Expert Finding on Heterogeneous Graphs through (k, P)-Core based Embedding. *ACM Trans. Knowl. Discov. Data* 17, 6 (2023), 85:1–85:35.

[78] Yuxiang Wang, Xiaoliang Xu, Qifan Hong, Jiahui Jin, and Tianxing Wu. 2021. Top-k star queries on knowledge graphs through semantic-aware bounding match scores. *Knowledge-Based Systems* 213 (2021), 106655.

[79] Yuxiang Wang, Yuyang Zhao, Xiaoliang Xu, Yue Wu, Tianxing Wu, and Xiangyu Ke. 2023. Random Walk-based Community Key-members Search over Large Graphs. arXiv:2210.17403 [cs.DB]

[80] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*.

[81] Xiaoliang Xu, Jun Liu, Yuxiang Wang, and Xiangyu Ke. 2022. Academic Expert Finding via (k, P)-Core based Embedding over Heterogeneous Graphs. In *ICDE*.

[82] Zongyu Xu, Yihao Zhang, Long Yuan, Yuwen Qian, Zi Chen, Mingliang Zhou, Qin Mao, and Weibin Pan. 2023. Effective Community Search on Large Attributed Bipartite Graphs. *Int. J. Pattern Recognit. Artif. Intell.* 37, 2 (2023), 2359002:1–2359002:25. https://doi.org/10.1142/S0218001423590024

[83] 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* (Beijing, China) *(MDS '12)*. Association for Computing Machinery, New York, NY, USA, Article 3, 8 pages. https://doi.org/10.1145/2350190.2350193

[84] Kai Yao and Lijun Chang. 2021. Efficient Size-Bounded Community Search over Large Networks. *PVLDB* 14, 8 (2021), 1441–1453.

[85] Junhao Ye, Yuanyuan Zhu, and Lu Chen. 2023. Top-r keyword-based community search in attributed graphs. In *ICDE*. 1652–1664.

[86] Junhao Ye, Yuanyuan Zhu, and Lu Chen. 2023. Top-r keyword-based community search in attributed graphs. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 1652–1664. https://doi.org/10.1109/ICDE55515.2023.00130

[87] 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* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 1103–1113. https://doi.org/10.1145/3394486.3403154

[88] Zhiwei Zhang, Xin Huang, Jianliang Xu, Byron Choi, and Zechao Shang. 2019. Keyword-Centric Community Search. In *ICDE*. 422–433.