Inductive Attributed Community Search: to Learn Communities across Graphs

Shuheng Fang
The Chinese University of Hong Kong
shfang@se.cuhk.edu.hk

Kangfei Zhao
Beijing Institute of Technology
zkf1105@gmail.com

Yu Rong
Alibaba DAMO Academy
yu.rong@hotmail.com

Zhixun Li
The Chinese University of Hong Kong
lizhixun1217@gmail.com

Jeffrey Xu Yu
The Chinese University of Hong Kong
yu@se.cuhk.edu.hk

ABSTRACT

Attributed community search (ACS) aims to identify subgraphs satisfying both structure cohesiveness and attribute homogeneity in attributed graphs, for a given query that contains query nodes and query attributes. Previously, algorithmic approaches deal with ACS in a two-stage paradigm, which suffer from structural inflexibility and attribute irrelevance. To overcome this problem, recently, learning-based approaches have been proposed to learn both structures and attributes simultaneously as a one-stage paradigm. However, these approaches train a transductive model which assumes the graph to infer unseen queries is as same as the graph used for training. That limits the generalization and adaptation of these approaches to different heterogeneous graphs.

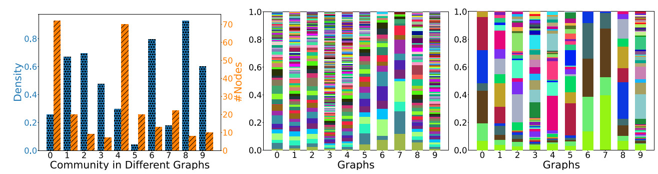
In this paper, we propose a new framework, Inductive Attributed Community Search, *IACS*, by inductive learning, which can be used to infer new queries for different communities/graphs. Specifically, *IACS* employs an encoder-decoder neural architecture to handle an ACS task at a time, where a task consists of a graph with only a few queries and corresponding ground-truth. We design a three-phase workflow, "training-adaptation-inference", which learns a shared model to absorb and induce prior effective common knowledge about ACS across different tasks. And the shared model can swiftly adapt to a new task with small number of ground-truth. We conduct substantial experiments in 7 real-world datasets to verify the effectiveness of *IACS* for CS/ACS. Our approach *IACS* achieves 28.97% and 25.60% improvements in F1-score on average in CS and ACS, respectively.

PVLDB Reference Format:
Shuheng Fang, Kangfei Zhao, Yu Rong, Zhixun Li, and Jeffrey Xu Yu. Inductive Attributed Community Search: to Learn Communities across Graphs . PVLDB, 17(10): 2576 - 2589, 2024.
doi:10.14778/3675034.3675048

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

*Kangfei Zhao is the corresponding author.

Proceedings of the VLDB Endowment, Vol. 17, No. 10 ISSN 2150-8097.
doi:10.14778/3675034.3675048



(a) Community Density & Size (b) Degree Distributions (c) Attribute Distributions

Figure 1: Heterogeneous characteristics of communities, degree distributions and attribute distributions of 10 Facebook ego-networks

1 INTRODUCTION

Attributed community search (ACS) has been widely studied to identify communities in an attributed graph, which satisfy both structure cohesiveness and attributes homogeneity for a given query that consists of query nodes and query attributes. ACS serves as a crucial building block in various real-world applications, such as social network analysis, recommendation systems [50, 59], bioinformatics [32] and fraud detection [48]. Conventionally, ACS has been solved by algorithmic approaches [19, 28] in a two-stage paradigm. First, it identifies candidate communities based on pre-defined structural patterns (e.g., k-core, k-truss, etc) for any graph. Second, it further refines the candidate communities using attribute homogeneity constraints in the graph. However, as pointed out by [17, 21, 29], the algorithmic approaches suffer from structural inflexiblity and attribute irrelevance, and fail to capture the joint correlations between structure and attribute.

To overcome the limitations of the algorithmic approaches in a two-stage paradigm, learning approaches are proposed to learn both structures and attributes simultaneously as a one-stage paradigm. Here, these methods train a model based on samples which are query-result pairs. The results are communities for some queries in a single graph. The learned model infers the communities from the same graph for a new query. Such learning-based models are *transductive* as they implicitly assume that the graph used to train with query samples and the graph used to infer unseen queries are the same. The transductive learning approaches have limitations. Different from the algorithmic approaches that can be used for any graph in general, the transductive learning approaches need to train/infer for each graph separately, and fail to capture diverse structure patterns and attribute distributions across different heterogeneous graphs. In addition, the transductive learning approaches may not be able to infer communities for unseen queries if the same graph evolves and becomes different from the graph used to train. That is because, with evolving graphs, graph distributions, attributes, and communities are all subject to change. To illustrate the limitation of a transductive learning approach to learn across different graphs, we conducted empirical analysis. Fig. 1 reveals data heterogeneity on 10 Facebook ego-networks regarding the structure of communities, graphs and attribute distributions. Fig. 1(a) depicts the density and size of the largest communities extracted from the 10 ego-networks. Fig. 1(b) and Fig. 1(c) plot the degree distributions and attribute distributions of the 10 ego-networks, respectively. As all are so different across graphs, transductive learning approaches, that rely on the natural generalization of Graph Neural Network (GNN) to deal with ACS for a graph, cannot construct a model to deal with ACS across heterogeneous graphs.

In this paper, we explore a new inductive approach for ACS across graphs for a query with multiple query nodes and query attributes. By inductive, we learn shared latent knowledge across different communities/graphs to capture diversified patterns in both structure cohesiveness and attribute homogeneity. We characterize the existing learning-based CS/ACS approaches and our approach in Table 1. Here, single/multi-node query indicates the ability to deal with a query with a single/multiple query node(s), and the attributed query indicates the ability to deal with query attributes. *AQD-GNN* [29]/*ICS-GNN* [21] do not have inductive ability since their trained models are tailored for specific graph/community. *CommunityAF* [11] and *COCLEP* [35] have a limited inductive ability, as they rely on the natural generalization of GNN to deal with CS for a graph. In other words, the inductive ability derives from GNN. These approaches cannot be easily extended to support heterogeneous data, since they follow supervised learning paradigm. The emergence of different attribute domains, graph structures, and community patterns poses serious obstacles for existing models to adapt to heterogeneous data. *CGNP* [17] is a meta-learning approach and has inductive ability. However, like *CommunityAF* and *COCLEP*, *CGNP* is designed for single-node queries and fails to deal with ACS across attributed graphs for multi-node queries and attributed queries. It is also challenging to extend CGNP to handle ACS due to the difficulty of aligning the attributes from heterogeneous graphs and encoding query information with graph structure effectively and collaboratively.

There are two main challenges in designing an inductive learning framework for ACS. *(1) How to empower the model to support complex ACS queries by inductive learning, procuring effective generalization on new communities, graphs and attributes? (2) How to enable the shared model to absorb and induce prior effective common knowledge about ACS across different tasks that exhibit heterogeneity in communities, graphs and attributes?* For (1), we construct a shared model across multiple ACS tasks, where a task is a graph with only a few training samples. The shared model requires aligning the node attributes and query attributes in graphs across different tasks. Specifically, to support queries containing multiple query nodes and query attributes, we align the node attributes of graphs for training and inference, in a compatible, fixed-length vector space, and we integrate the topological information of query nodes and semantic information of query attributes to the largest extent. For (2), we adopt a three-phase workflow, i.e., "training-adaptation-inference". We first train the shared model to learn some common

Table 1: Learning-based Community Search Approaches

Approaches	Single-node Query	Multi-node Query	Attributed Query	Induction
AQD-GNN [29]	✔	✔	✔	✘
ICS-GNN [21]	✔	✘	✘	✘
CommunityAF [11]	✔	✘	✘	✔✘
COCLEP [35]	✔	✘	✘	✔✘
CGNP [17]	✔	✘	✘	✔
IACS (Ours)	✔	✔	✔	✔

patterns of communities about the structure or attribute distribution from a wide range of heterogeneous tasks. Then, the shared model is fine-tuned via explicit adaptation, before the inference in new graphs/communities.

We propose a DL framework named Inductive Attributed Community Search (*IACS*), for ACS by inductive learning. In brief, *IACS* is equipped with an encoder-decoder neural network architecture that processes one ACS task at a time. To empower the model by inductive learning, we deploy the "training-adaptation-inference" three-phase workflow. We adopt a meta algorithm to train a shared model by a collection of training ACS tasks, and explicitly adapt the model for a new ACS task by exploiting a small number of training samples. To enable the model to learn effective common knowledge from heterogeneous tasks, we align node attributes across different graphs in tasks by devising an enhanced attribute encoding via pre-training node embeddings in attributed augmented graphs. To further enhance generalization and adaptability, we design an adaptive encoder with simple yet effective adaptation modules. The contributions of this paper are summarized as follows.

- We propose a new DL framework, Inductive Attributed Community Search (*IACS*), to comprehensively support CS/ACS queries, ranging from simple queries with a single node to complex queries with multiple nodes and multiple attributes.
- We devise a GNN-based encoder-decoder neural network model which is capable of dealing with heterogeneous ACS tasks. We design a pre-trained attribute embedding for the GNN encoder to align the input node features, enabling the model to be shared by different attribute sets. To prompt model adaptation, we propose an adaptive decoder with two variants.
- We propose a three-stage workflow to fulfill inductive ACS, i.e., training a shared model by a meta algorithm on multiple ACS tasks, adapting the model to a new ACS task by fine-tuning on limited training samples and deploying the model for online queries.
- We conduct substantial experimental studies on 7 real-world datasets with ground-truth communities. Compared with 3 algorithmic methods, 3 ML/DL-based methods and 2 meta-learning methods, our *IACS* framework outperforms these baselines with higher effectiveness and efficiency.

Roadmap. The rest of the paper is organized as follows. §2 reviews our related work. In §3, we introduce the problem statement followed by existing GNN framework for learning-based ACS. We elaborate on the architecture and workflow of *IACS* in §4 and §5, respectively. We present our comprehensive experimental studies in §6 and conclude the paper in §7.

2 RELATED WORK

Attributed Community Search. Attributed community search (ACS) not only focuses on query nodes but also incorporates query attributes. When dealing with attributed graphs, algorithmic approaches [19, 23, 28, 38, 42] have been proposed by considering both the structural cohesiveness and attribute homogeneity. Algorithmic approaches such as [23, 38, 42] rely on their predefined subgraph patterns to identify communities which limits their flexibility. *ATC* [28] and *ACQ* [19] are two representative approaches for ACS. They both adopt a two-stage process; they first identify potential communities based on structural constraints and then compute attribute scores to verify the candidates. However, the independent two-stage process fails to capture the correlations between structures and attributes in a joint fashion, leading to unsatisfactory results. With the development of ML/DL, learning-based approaches for CS [11, 17, 21, 35] have been developed. However, as the summary in Table 1, except *AQD-GNN*, these approaches cannot support multi-node queries or ACS. *AQD-GNN* [29] proposes a GNN-based supervised model for ACS in a single graph. The model is trained by a collection of ACS queries with corresponding ground-truth, and predicts the communities for unseen queries. The limitation of *AQD-GNN* is that the model lacks generalizability for new communities, graphs and attributes that are not encountered in the training phase.

GNN for Graph Analytics. GNN iteratively aggregates neighbor information of nodes by learnable weights to learn powerful representation on graphs. In addition to CS, GNN is widely applied in various graph analytical tasks, including graph combinatorial optimization problems [10, 22, 36], subgraph matching [4, 15, 41], subgraph counting [39, 65, 66], community collapsing [67] and community detection [12, 37]. To further enhance the capabilities of GNN, inductive learning of GNN [24, 60] has emerged, specifically targeting model generalization on unseen nodes, edges or graphs. The down-streaming tasks range from inductive node classification [24, 26, 47, 60] to graph classification [5, 34, 64], graph-based recommender system [18, 44, 58, 62], fraud detection [8, 43] and inductive link prediction [14, 25, 52]. To the best of our knowledge, our approach is the first to deal with ACS problem by inductive learning, facilitating better generalization to new communities, graphs and node attributes.

ML for Subgraph Extraction. Recently, ML techniques have been employed for many subgraph extraction tasks, including community search, community detection [55], maximum common subgraph (MCS) and subgraph isomorphism counting (SIC) [46, 49]. LGNN [12] and CommDGI [63] utilize GNN for community detection which include a GNN representation module and a detection module. In contrast to existing MCS algorithms that are based on Branch and Bound (BnB) algorithm with rule-based heuristics, McSplit [40] and GLSearch [6] introduce reinforcement learning to the BnB search process, learning more powerful search heuristics. For the SIC problem, conventional algorithms are trapped into the dilemmas of scalability or sampling failure. To overcome these dilemmas, DIAMNet [39], ALSS [66] and NeurSC [53] establish neural network regression models to answer subgraph counting queries approximately.

Table 2: Frequently-used Notations

Notation	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$	an attributed graph
n/m	number of nodes/edges of graph $\mathcal{G}$
$q = (\mathcal{V}_q, \mathcal{A}_q)$	a query with node set $\mathcal{V}_q$ and attribute set $\mathcal{A}_q$
C_q	the community containing query q
$\mathcal{T} = (\mathcal{G}, Q, L)$	one task
l_q^+/l_q^-	positive/negative samples for query q
e_a	pre-trained attribute embedding for attribute $a \in \mathcal{A}$
$e(v)$	original feature encoding for node v
$I_l v$	indicator whether the node v is in the community
$e_{\mathcal{V}_q}$	average query nodes embeddings
$e_{\mathcal{A}_q}$	average query attributes embeddings
$\mathcal{S}_i = (Q_i, L_i)$	support set of a task $\mathcal{T}_i$

3 PRELIMINARIES

In this section, we first introduce the definitions of CS and ACS formally, and then formulate the inductive learning-based CS. Finally, we describe the existing GNN-based framework for CS as the technical background. Table 2 depicts the frequently-used notations and their descriptions.

3.1 Definitions & Concepts

An undirected simple graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes, $\mathcal{V}$, and a set of undirected edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Let $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$ denote the number of nodes and edges, respectively. The neighborhood of node v_i is denoted as $\mathcal{N}(v_i) = \{v_j | (v_j, v_i) \in \mathcal{E}\}$.

Community Search (CS). For a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, given a node set $\mathcal{V}_q \subseteq \mathcal{V}$ as a query q, the problem of Community Search aims to find the query-dependent community $C_q \subseteq \mathcal{V}$, where the nodes in C_q are intensively intra-connected, i.e., maintaining cohesive structure.

An undirected attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ has an additional attribute set $\mathcal{A}$. Each node v_i possesses its attribute set $\mathcal{A}_i$, and $\mathcal{A}$ is the union of all the node attribute sets, i.e., $\mathcal{A} = \mathcal{A}_1 \cup \cdots \cup \mathcal{A}_n$.

Attributed Community Search (ACS). For an attributed graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, given a query $q = (\mathcal{V}_q, \mathcal{A}_q)$ where $\mathcal{V}_q \subseteq \mathcal{V}$ is a set of query nodes, and $\mathcal{A}_q \subseteq \mathcal{A}$ is a set of query attributes, the problem of Attributed Community Search (ACS) aims to find the query-dependent community $C_q \subseteq \mathcal{V}$. Nodes in community C_q need to be structure cohesive and attribute homogeneous simultaneously, i.e., the nodes in the community are densely intra-connected in structure and the attributes of these nodes are similar.

Learning-based CS/ACS. The general process of the learning-based approaches [11, 17, 21, 29, 35] consists of two stages, the training stage and the inference stage. In the training stage, for a graph $\mathcal{G}$, a parametric ML model $\mathcal{M} : q \mapsto [0, 1]^n$ is constructed offline from a set of queries and corresponding ground-truth communities. In the inference stage, for an online new query, the model $\mathcal{M}$ predicts the likelihood of whether each node is in the community of the query, as a vector $\hat{y} \in [0, 1]^n$. The query supported can be non-attributed queries ($q = (\mathcal{V}_q, \emptyset)$) for CS and attributed queries ($q = (\mathcal{V}_q, \mathcal{A}_q)$) for ACS. To be concise, we consider the ACS problem in this paper and regard CS as a special case of ACS ($\mathcal{A}_q = \emptyset$). Distinguished from prior algorithmic approaches [19, 27, 28], the community C_q discovered by learning-based approaches is not restricted to any specific k-related subgraph.

3.2 Problem Statement

For existing learning-based ACS [29], the model $\mathcal{M}$ trained in a graph $\mathcal{G}$ is expected to serve the same graph involving the same communities in the inference stage. In this paper, we aim to explicitly empower the model to generalize and adapt to new communities and graphs by inductive learning, in the following two perspectives:

For new communities. For a graph $\mathcal{G}$, given a set of training queries $Q = \{q_1, \cdots, q_i\}$ with corresponding ground-truth labels from the community set $\{C_{q_1}, \cdots, C_{q_i}\}$, the model trained by Q is used to answer query q^* from a new community C_{q^*}, i.e., $C_{q_1} \cap C_{q^*} = \emptyset, \cdots, C_{q_i} \cap C_{q^*} = \emptyset$. Furthermore, the graph $\mathcal{G}$ may even not contain the community C_{q^*}, e.g., C_{q^*} is in a large online social network where $\mathcal{G}$ is a local subgraph extracted offline.

For new graphs. For a graph $\mathcal{G}$, a model constructed from queries in $\mathcal{G}$ is used to answer queries from a new graph $\mathcal{G}^*$.

Example 3.1: Fig. 2 demonstrates a toy example of above two perspectives of inductive ACS. In Fig. 2(a), the model is constructed by training data of an academic community (in orange), and will be used for answering queries from a new musical community (in blue). Although the two communities are in a single large graph, their local structures and attribute sets are different. In Fig. 2(b), the model is trained by a graph containing one community (on the left), and is expected to answer ACS queries in a new graph (on the right).

The challenges of model generalization on new communities and graphs lie in data heterogeneity, i.e., *structural heterogeneity* and *attribute heterogeneity*. For one thing, the topological structures are heterogeneous across different communities and graphs. For the other thing, the attribute set, their semantics and distribution are heterogeneous across different communities and graphs. In general, data heterogeneity across different graphs would be more severe than that across different communities. To this end, in this paper, we construct a model $\mathcal{M}$ by inductive learning from multiple ACS tasks.

ACS Task. We formulate an ACS task as a triplet $\mathcal{T} = (\mathcal{G}, Q, L)$. And $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ is an attribute graph, $Q = \{q_1, \cdots, q_i\}$ is a set of queries, i.e., $q_j = (\mathcal{V}_{q_j}, \mathcal{A}_{q_j})$, $\mathcal{V}_{q_j} \subseteq \mathcal{V}$, $\mathcal{A}_{q_j} \subseteq \mathcal{A}$, $\forall j \in [1, \cdots, i]$, $L = \{l_{q_1}, \cdots, l_{q_i}\}$ is the set of ground-truth of j queries, correspondingly. Specifically, l_q is a nonempty node set in $\mathcal{G}$ w.r.t. query q, containing a set of positive node samples, $l_q^+ \subseteq C_q$, and a set of negative samples, $l_q^- \subseteq (\mathcal{V} \setminus C_q)$.

By inductive learning, the model $\mathcal{M}$ is trained on a set of training tasks, $\{\mathcal{T}_1, \cdots, \mathcal{T}_N\}$, and will be used in a new ACS task $\mathcal{T}^* = (\mathcal{G}^*, Q^*, L^*)$. Here, $\mathcal{G}^*$ and the graphs in the training tasks are either different local subgraphs in a large graph, which do not have overlapping communities, or are fully different graphs. Here, Q^* and L^* are a small number of queries with corresponding ground-truth for $\mathcal{T}^*$, i.e., $|Q^*| \ll |\mathcal{V}|$, which can be exploited by $\mathcal{M}$ to adapt to the specific task $\mathcal{T}^*$. The number of queries in Q^*, $|Q^*|$, is called shot in the following.

3.3 GNN for Learning-based ACS

Existing learning-based CS/ACS approaches [11, 21, 29, 35] employ GNN as the backbone of their models. A GNN of K-layers follows a neighborhood aggregation paradigm to generate a new embedding for each node by aggregating the embeddings of its neighbors in K iterations. Let $h^{(k)}(v)$ denote the embedding of node v in the k-th iteration. In the k-th iteration (layer), an aggregate function $f_{\mathbb{A}}^{(k)}$ aggregates the embeddings of the neighbors of v generated in $(k-1)$-th layer as Eq. (1). Subsequently, a combine function $f_{\mathbb{C}}^{(k)}$ updates the embedding of v as Eq. (2). The aggregate and combine functions of each layer are neural networks.

$$a^{(k)}(v) = f_{\mathbb{A}}^{(k)}(\{h^{(k-1)}(u) | u \in \mathcal{N}(v)\}), \tag{1}$$

$$h^{(k)}(v) = f_{\mathbb{C}}^{(k)}(h^{(k-1)}(v), a^{(k)}(v)). \tag{2}$$

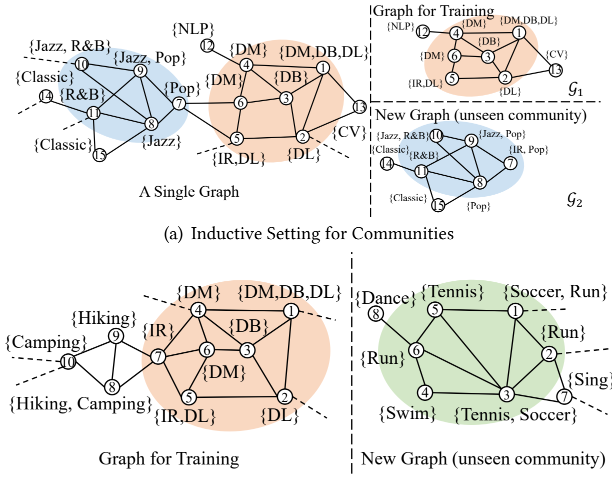
For dealing with ACS, the query information is injected into the initial node embedding, $h^{(0)}(v)$, by concatenating identifiers of query nodes and query attributes to the original node features as Eq. (3).

$$h^{(0)}(v) = [I_q(v) \| I_{\mathcal{A}}(v) \| \mathcal{A}(v)], \tag{3}$$

Here, $I_q(v) \in \{0, 1\}$ identifies whether node v is a query node, $I_{\mathcal{A}}(v) \in \{0, 1\}^{|\mathcal{A}|}$ identifies which attributes are the query attributes, and $\mathcal{A}(v)$ is the vectorized representation of the attributes of v. Through the transformation of K layers, GNN-based models predict the likelihood that v is in the community of the query by a prediction layer $\hat{f}$, i.e., $\hat{y}(v) = \hat{f}(h^{(K)}(v))$, where the prediction layer $\hat{f}$ may contain extra neural network layers, followed by sigmoid activation. Thereby, an ACS task $\mathcal{T} = (\mathcal{G}, Q, L)$ is a query-specific binary classification task in $\mathcal{G}$, where GNN-based models are trained by minimizing the binary cross entropy (BCE) loss on queries Q and ground-truth L as Eq. (4).

$$\mathcal{L}(q; \theta) = -\sum_{v^+ \in l_q^+} \log \hat{y}(v^+) - \sum_{v^- \in l_q^-} \log(1 - \hat{y}(v^-)) \tag{4}$$

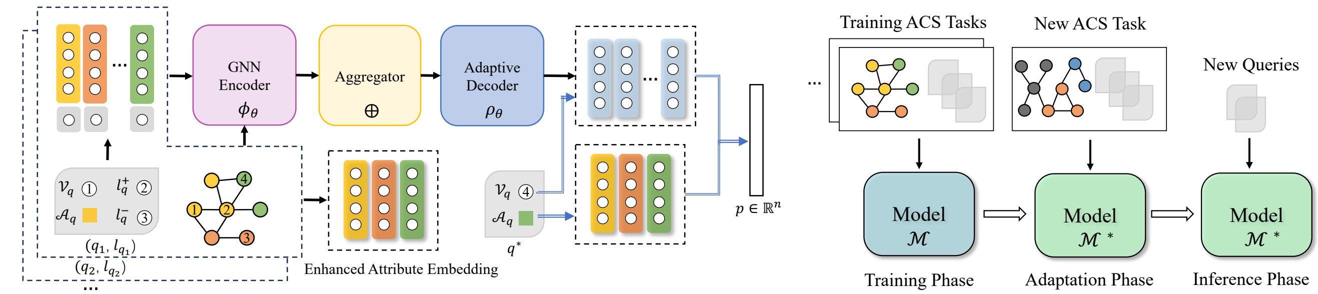
Existing approaches train a distinct model for each ACS task by implicitly assuming that the graph used for training and inference for unseen queries are the same. Such transductive models cannot infer communities for queries from different graphs and different attribute sets.



(a) Inductive Setting for Communities

(b) Inductive Setting for Graphs

Figure 2: Two Cases of Inductive ACS



(a) Architecture: the model processes one ACS task at a time by a GNN encoder, an aggregator and a decoder (b) The Three-phase Workflow

Figure 3: Architecture and Workflow of *IACS*

4 *IACS* ARCHITECTURE

In this section, we first present a high-level overview of *IACS*, and then introduce the architecture of each component in detail.

4.1 Overview

We present the overview of our *IACS* framework, including the model architecture and workflow.

Architecture. Fig. 3(a) shows the architecture of *IACS*. Given an ACS task $\mathcal{T} = (\mathcal{G}, Q, L)$, *IACS* directly models the predictive distribution $p(y_{q^*}|q^*, \mathcal{T})$ for a new query node $q^* \in \mathcal{V} \setminus Q$, where $y_{q^*} = \{y_{q^*}(v)\}_{v \in \mathcal{V}} \in \{0,1\}^n$ is the binary target prediction indicating whether each node v in $\mathcal{G}$ is in the community of the query q^*. *IACS* adopts an encoder-decoder neural architecture which processes a task at a time by Eq. (5).

$$p(y_{q^*}|q^*, \mathcal{T}) = \rho_\theta\Bigg(q^*, \bigoplus_{(q,l_q)\in(Q,L)} \phi_\theta(q, l_q)\Bigg) \tag{5}$$

Here, $\phi_\theta(\cdot)$ is a neural encoder that transforms a query q with the corresponding ground-truth l_q into a context embedding. We devise an enhanced attribute encoding via pre-training node embeddings in attributed augmented graphs to align node attributes across different graphs. We adopt a GNN encoder to encapsulate hidden graph structure and query-specific knowledge into the context embedding. Then, a commutative aggregate operator $\oplus$ aggregates all the context embeddings generated from Q and L in the task into a task-level context embedding in a permutation-invariant fashion. The task-level context embedding, serving as a neural index, is used to predict the community membership for new queries. In other words, finally, a neural decoder $\rho_\theta(\cdot)$ takes the task-level embedding and any query q^* to predict its community. In the following, we will elaborate on the design details of the encoder, the aggregator and the decoder, respectively.

Workflow. As Fig. 3(b) illustrates, the workflow of *IACS* consists of three phases sequentially, the training, adaptation and inference phases. In the training phase, the model of *IACS*, $\mathcal{M}$, is trained over a set of training tasks, $\{\mathcal{T}_1, \cdots, \mathcal{T}_N\}$. By learning shared encoder, aggregator and decoder, prior knowledge for ACS is induced from these multiple tasks. Then, the shared model $\mathcal{M}$ is deployed to a new ACS task $\mathcal{T}^* = (\mathcal{G}^*, Q^*, L^*)$ by an adaptation phase. The shared model is slightly adjusted to a task-specific model $\mathcal{M}^*$ by the ground-truth Q^* and L^* provided, and the task-level context

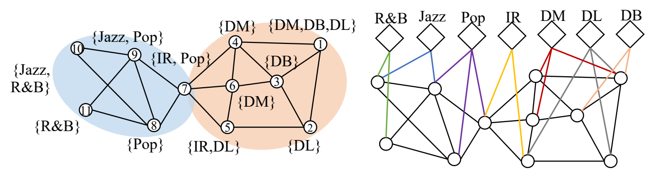


Figure 4: A graph $\mathcal{G}$ and its attribute-augmented graph $\mathcal{G}_{\mathcal{A}}$

embedding is constructed by $\mathcal{M}^*$. In the inference phase, the decoder queries the task-level embeddings for new queries to form the ACS results. We defer the details of the three-stage workflow of *IACS* in §5.

4.2 Encoder

For each query $q = (\mathcal{V}_q, \mathcal{A}_q) \in Q$ and the corresponding ground-truth $l_q \in L$, the encoder $\phi_\theta(\cdot)$ is a K-layer GNN that transforms (q, l_q) together with the graph $\mathcal{G}$ into a node embedding matrix $H_q = \{h^{(K)}(v)_{v\in\mathcal{V}}\} \in \mathbb{R}^{n \times d^K}$. Here, $h^{(K)}(v)$ is a d^K dimensional vectorized output of the K-th layer of GNN for node v. The subscript q of H_q indicates that the node embeddings H_q, as the query-level context embedding, is particularly for query q. To be specific, all the queries in one task share the encoder, and queries across different tasks share the encoder. For one thing, to fulfill induction over multiple tasks, the input of the encoder should be aligned to a compatible, fixed-dimensional vector space. Unfortunately, the encoders of existing methods [29, 35] typically require retraining for a new task due to the supervised learning paradigm they adopt. For the other thing, to support accurate ACS, we enhance the input of the GNN encoder by fusing the features of the query node and node attribute in-depth. To this end, we construct the initial node embedding $h^{(0)}(v) \in \mathbb{R}^{(d+1)}$ as Eq. (6), by concatenating the binary ground-truth identifier $I_l(v) \in \{0, 1\}$ and an enhanced attribute embedding $e(v) \in \mathbb{R}^d$.

$$h^{(0)}(v) = [I_l(v) \| e(v)], \text{where } I_l(v) = \begin{cases} 1 & v \in l_q^+ \cup \mathcal{V}_q, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

In Eq. (6), $I_l(v)$ identifies whether the node v is in the community ground-truth for the query q, under the close-world assumption. The enhanced attribute embedding $e(v)$ encodes the attributes associated with the node v, whose details are given as follows.

Enhanced Attribute Encoding. One-hot attribute encoding is widely used by GNN in node classification and link prediction [24, 31]. However, the GNN model cannot be shared by different ACS tasks with different attribute sets. Moreover, such sparse encoding is lack of insight for ACS problem, which is correlated with both attribute information and topological structure. These motivate us to design a fixed-length, enhanced attribute encoding.

To establish a connection between nodes and attributes, a previous approach, *AQD-GNN* [29], constructs a bipartite graph containing two types of nodes, i.e., graph nodes and attribute nodes, where the graph nodes are connected to the attribute node associated. However, the bipartite graph fails to incorporate the knowledge of the original graph structure. To capture the integration between graph structure and attribute, we pre-train an enhanced attribute encoding on an attribute-augmented graph $\mathcal{G}_{\mathcal{A}} = (\mathcal{V} \cup \mathcal{V}_{\mathcal{A}}, \mathcal{E} \cup \mathcal{E}_{\mathcal{A}})$, which is constructed from a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$. Precisely, here $\mathcal{V}_{\mathcal{A}}$ is a set of nodes where each node v_a represents an attribute $a \in \mathcal{A}$. And $\mathcal{E}_{\mathcal{A}} \subseteq \mathcal{V} \times \mathcal{V}_{\mathcal{A}}$ is a collection of edges, if node v possesses an attributed a, there exists an edge connecting node v to the corresponding attributed node v_a. Fig. 4 shows the attribute-augmented graph $\mathcal{G}_{\mathcal{A}}$ for an attributed graph $\mathcal{G}$. We use a scalable, task-independent graph embedding algorithm, ProNE [61], to pre-train a node embedding for the attribute-augmented graph $\mathcal{G}_{\mathcal{A}}$. With the pre-trained attribute embedding, we encode the enhanced attributed embedding of node v as the summation of the embedding of its attributes (Eq. (7))

$$e(v) = \sum_{a \in \mathcal{A}(v)} e_a, \tag{7}$$

where $e_a \in \mathbb{R}^d$ is the pre-trained embedding of attributed node v_a in $\mathcal{G}_{\mathcal{A}}$, and $\mathcal{A}(v)$ denotes the set of attributes associated with node v. Therefore, via node embedding of the attribute-augmented graph, the graph nodes in different tasks acquire a fixed-length attribute encoding which also leverages the graph structure of the task.

4.3 Aggregator

Recall that given an ACS task $\mathcal{T} = (\mathcal{G}, Q, L)$, the GNN encoder $\phi_\theta(\cdot)$ generates a query-level embedding $H_q \in \mathbb{R}^{n \times d^K}$ for each query q with corresponding ground-truth l_q. The aggregator combines the query-level embedding H_q for all queries in Q into one task-level context embedding $H \in \mathbb{R}^{n \times d^K}$. We use the permutation invariant operator, *average*, as the parameter-free aggregator as Eq. (8).

$$H = \frac{1}{|Q|} \sum_{q \in Q} H_q \tag{8}$$

The task-level embedding H serves as a learned query index of the task $\mathcal{T}$, which will be used to be queried by the embedding of a query $q = (\mathcal{V}_q, \mathcal{E}_q)$ in the decoder $\rho_\theta(\cdot)$. This idea is enlightened by a neural network architecture for 3D scene understanding and rendering, Generative Query Network (GQN) [16], where embeddings of few-shot 3D views are aggregated for querying the view of a new 3D perspective.

4.4 Adaptive Decoder

Distinguished from existing works [17, 29] without any adaptability enhancement in their models, in *IACS*, we design an adaptive decoder $\rho_\theta(\cdot)$ to predict the community membership for a new ACS query $q^* = (\mathcal{V}_{q^*}, \mathcal{A}_{q^*})$, by leveraging the task-level embedding H as a hidden query index. Intuitively, H preserves the status of all the nodes in $\mathcal{G}$, involving the graph structure and attribute distribution from the enhanced attributed encoding, and known community information from Q and L. The community membership of q^* is predicted by computing the similarity of the embedding of q^* with the embedding H.

Firstly, to improve the adaptability of model prediction on one task, we introduce a specific adaptation modulation, Feature-wise Linear Modulation (FiLM), into the decoder, which has demonstrated to be highly effective in various applications [7, 9, 45]. Specifically, FiLM applies a feature-wise affine transformation on H, conditioned on H itself as shown in Eq. (9).

$$\begin{aligned} \gamma &= W_\gamma H, \ \beta = W_\beta H, \\ \widehat{H} &= \gamma \odot H + \beta. \end{aligned} \tag{9}$$

In Eq. (9), $W_\gamma, W_\beta \in \mathbb{R}^{d^K \times d^K}$ are two weight matrices and $\odot$ is the element-wise matrix multiplication. The matrices γ and β learned from H, scales and shifts H into a self-adaptive task-level embedding $\widehat{H} \in \mathbb{R}^{n \times d^K}$ in a feature-wise way. Furthermore, to improve the effectiveness of the modulation, we design a variant of FiLM with a gating mechanism to avoid filtering out informative features. This FiLM variant is shown in Eq. (10) where $W_\delta, W_\epsilon \in \mathbb{R}^{d^K \times d^K}$ are weight matrices.

$$\begin{aligned} &\delta = \mathsf{sigmoid}(W_\delta H), \ \epsilon = W_\epsilon H, \\ &\widehat{\gamma} = \gamma \odot \delta + \epsilon \odot (1 - \delta), \ \widehat{\beta} = \beta \odot \delta + \epsilon \odot (1 - \delta), \\ &\widehat{H} = \widehat{\gamma} \odot H + \widehat{\beta}. \end{aligned} \tag{10}$$

The adaptive task-level embedding $\widehat{H}$ from FiLM modulation is used to answer ACS query $q^* = (\mathcal{V}_{q^*}, \mathcal{A}_{q^*})$ by similarity computation. Specifically, we construct the vector representation of q^* by firstly formulating the respective query node embedding and query attribute embedding as Eq. (11). Since $\widehat{H}$ is a node embedding of $\mathcal{G}$, the query node embedding $e_{\mathcal{V}_{q^*}} \in \mathbb{R}^{d^K}$ is the average node embedding in $\widehat{H}$ for query nodes $v \in \mathcal{V}_{q^*}$. And the query attributed embedding $e_{\mathcal{A}_{q^*}}$ is the average of attribute embeddings for query attributes $a \in \mathcal{A}_{q^*}$, where the attribute embeddings are the pre-trained embeddings of the attribute-augmented graph. Then, as shown in Eq. (12), the final query embedding $e_{q^*} \in \mathbb{R}^{d^K}$ is generated by concatenating the query node embedding and query attribute embedding, followed by the mapping of a multi-layer perception (MLP).

$$e_{\mathcal{V}_{q^*}} = \frac{1}{|\mathcal{V}_{q^*}|} \sum_{v \in \mathcal{V}_{q^*}} \widehat{H}(v), \ e_{\mathcal{A}_{q^*}} = \frac{1}{|\mathcal{A}_{q^*}|} \sum_{a \in \mathcal{A}_{q^*}} e_a \tag{11}$$

$$e_{q^*} = \mathsf{MLP}\left(e_{\mathcal{V}_{q^*}} \| e_{\mathcal{A}_{q^*}}\right) \tag{12}$$

Finally, we use the inner product operation $\langle \cdot \rangle$ to compute the similarity score of the query embedding e_{q^*} and the adaptive task-level embedding $\widehat{H}$ as Eq. (13). And the similarity score is transformed into the predictive probability that one node is in the same community with query q^*. The inner product operation indicates that the smaller the angle between the node embeddings in $\widehat{H}$ and the query embedding in the vector space, the more likely the nodes are from the same community of the query.

$$p(\hat{l_{q^*}} | q^*, \mathcal{T}) = \mathsf{sigmoid}\left(\langle e_{q^*}, \widehat{H} \rangle\right) \tag{13}$$

Algorithm 1: *IACS* Training Phase

Input : training task set $\mathcal{D} = \{\mathcal{T}_i\}_{i=1}^{N}$, learning rate α, number of epochs T

Output: parameters θ of meta model $\mathcal{M}$

1 **for** *epoch* $\leftarrow 1$ *to* T **do**
2   Shuffle the task set $\mathcal{D} = \{\mathcal{T}_i\}_{i=1}^{N}$;
3   **for** $\mathcal{T}_i = (\mathcal{G}_i, Q_i, L_i) \in \mathcal{D}$ **do**
4     $\mathcal{S}_i \sim (Q_i, L_i)$; $\triangleright$ *sample a support set*
5   **for** $(q, l_q) \in \mathcal{S}_i$ **do**
6       $H_q \leftarrow \phi_\theta(q, l_q, \mathcal{G}_i)$; $\triangleright$ *compute query-level embedding*
7     $H \leftarrow \bigoplus_{(q,l_q)\in\mathcal{S}_i} H_q$; $\triangleright$ *compute task-level embedding*
8   **for** $(q, l_q) \in (Q_i, L_i)$ **do**
9       $p(\hat{l_q}|q, \mathcal{T}_i) \leftarrow \rho_\theta(q, H)$; $\triangleright$ *compute predictive probability*
10   Compute the Loss $\mathcal{L}(q)$ by $p(\hat{l_q}|q, \mathcal{T}_i)$ and l_q;
11     $\mathcal{L} \leftarrow \sum_{(q,l_q)\in(Q_i,L_i)} \mathcal{L}(q)$;
12     $\theta \leftarrow \theta - \alpha\nabla_\theta\mathcal{L}$; $\triangleright$ *update model parameters*
13 **return** θ;

Algorithm 2: *IACS* Adaptation Phase

Input : test task $\mathcal{T}^* = (\mathcal{G}^*, Q^*, L^*)$, parameters θ of meta model $\mathcal{M}$, learning rate α, fine-tuning step s

Output: fine-tuned parameters θ^* and self-adaptive embedding $\widehat{H}$

1 $\mathcal{S}^* \leftarrow (Q^*, L^*)$; $\theta^* \leftarrow \theta$ $\triangleright$ *initialize fine-tuning data and parameters*
2 **for** $i \leftarrow 1$ *to* s **do**
3   **for** $(q, l_q) \in \mathcal{S}^*$ **do**
4     $H_q \leftarrow \phi_\theta(q, l_q, \mathcal{G}^*)$; $\triangleright$ *compute query-level embedding*
5   $H \leftarrow \bigoplus_{(q,l_q)\in\mathcal{S}_i} H_q$; $\triangleright$ *compute task-level embedding*
6   **for** $(q, l_q) \in \mathcal{S}^*$ **do**
7     $p(\hat{l_q}|q, \mathcal{T}^*) \leftarrow \rho_\theta(q, H)$; $\triangleright$ *compute predictive probability*
8   Compute the Loss $\mathcal{L}(q)$ by $p(\hat{l_q}|q, \mathcal{T}^*)$ and l_q;
9   $\mathcal{L} \leftarrow \sum_{(q,l_q)\in\mathcal{S}^*} \mathcal{L}(q)$;
10   $\theta^* \leftarrow \theta^* - \alpha\nabla_\theta\mathcal{L}$; $\triangleright$ *update model parameters*
11 **return** θ^* *and* $\widehat{H}$; $\triangleright$ $\widehat{H}$ *is computed from Eq.* (9) *or Eq.* (10)

It is worth mentioning that our *IACS* model can be easily generalized to deal with CS queries without query attributes, i.e., $q^* = (\mathcal{V}_{q^*}, \emptyset)$. Here, we use $e_{\mathcal{V}_{q^*}}$ in Eq. (11) as the query embedding e_{q^*}, and then compute the predictive probability by Eq. (13).

5 *IACS* WORKFLOW AND ANALYSIS

We present the 3-phase workflow of *IACS* in this section, followed by a detailed complexity analysis.

Model Training Phase. In the training phase, we construct a shared model $\mathcal{M}$ by training on a set of possible heterogeneous tasks $\mathcal{D} = \{\mathcal{T}_i\}_{i=1}^{N}$ offline. Then, the procured model can be used for model adaptation and inference for online ACS queries. The training objective is to minimize the negative log-likelihood of the predicted community membership across all the training tasks as shown in Eq. (14), which is aligned to the BCE loss (Eq. (4)) between the prediction and ground-truth.

$$\begin{aligned}\mathcal{L} &= \sum_{\mathcal{T}_i\in\mathcal{D}} \sum_{(q,l_q)\in(Q_i,L_i)} -\log p(\hat{l_q}|q, \mathcal{T}_i) \\ &= \sum_{\mathcal{T}_i\in\mathcal{D}} \sum_{(q,l_q)\in(Q_i,L_i)} \left(-\sum_{v^+\in l_q^+} \log \hat{y}(v^+) - \sum_{v^-\in l_q^-} \log(1-\hat{y}(v^-)) \right)\end{aligned} \tag{14}$$

Algorithm 1 presents the training algorithm of *IACS* by stochastic gradient descent. The algorithm iterates on randomly shuffled training tasks and processes one task for a gradient update in line 3-12. Specifically, for each task $\mathcal{T}_i$, we first randomly sample a fixed-size support set $\mathcal{S}_i$ from the given query Q_i and ground-truth L_i in $\mathcal{T}_i$ (line 4). Second, the GNN encoder, $\phi_\theta(\cdot)$, computes a query-level embedding H_q for each query q and its corresponding ground-truth l_q in $\mathcal{S}_i$ (line 5-6). Third, all the query-level embeddings are aggregated into one task-level embedding H by the permutation-invariant average aggregator $\oplus$ of Eq. (8) in line 7. Fourth, for each query in Q_i, we compute the predictive probability $p(\hat{l_q}|q, \mathcal{T}_i)$ by the adaptive decoder $\rho(\cdot)$ (line 9) and the query-specific loss $\mathcal{L}(q)$ (line 10). Finally, the model parameters are updated by one gradient step based on the aggregated task-specific loss (line 11-12).

Model Adaptation Phase. For a new ACS test task, the model $\mathcal{M}$ is first fine-tuned via a swift adaptation phase, utilizing a possible limited amount of community ground-truth, then $\mathcal{M}$ is used for inference on ACS queries. Algorithm 2 outlines the adaptation phase of *IACS* for a task $\mathcal{T}^* = (\mathcal{G}^*, Q^*, L^*)$. The algorithm fine-tunes model $\mathcal{M}$ by s gradient steps, by leveraging (Q^*, L^*) as the support set $\mathcal{S}^*$, where the forward-pass of the model (line 3-7) is similar to Algorithm 1. Here, we also use (Q^*, L^*) for making predictions and computing the loss. After model adaptation, the algorithm returns fine-tuned model parameters and the self-adaptive task-level embedding $\widehat{H}$ procured by the FiLM module in the adaptive decoder (Eq. (9) or Eq. (10)). The embedding $\widehat{H}$ is persisted as a neural index for computing online queries.

Model Inference Phase. For a query $q^* = (\mathcal{V}_{q^*}, \mathcal{A}_{q^*})$ in $\mathcal{G}^*$, we first extract query node embedding $e_{\mathcal{V}_{q^*}}$ from $\widehat{H}$ and extract query attribute embedding $e_{\mathcal{A}_{q^*}}$ from the enhanced attributed embeddings. Then, the two embeddings are fused by Eq. (12) to generate the overall query embedding e_{q^*}. As shown in Eq. (13), the query embedding e_{q^*} is used to predict the probability of community membership of each node by computing the inner product similarity with the persisted embedding $\widehat{H}$. To transform the probability into community membership, we use a threshold γ, i.e., the node is regarded as in the community of query q^* if and only if the probability is greater than γ.

Complexity Analysis. We analyze the complexity of *IACS* briefly. For time complexity, to be concise, we assume that basic vector operations such as addition, multiplication, concatenation, and inner product take constant time. The GNN encoder in *IACS* takes $O(Km|\mathcal{S}|)$ time for a single task, where K is the number of GNN layers, m is the number of edges and $|\mathcal{S}|$ is the number of shots. The complexity of the big $\oplus$ operation, i.e., an average pooling, is $O(n|\mathcal{S}|)$. For the decoder, the inner product operation takes $O(n|Q|)$ time. In total, the training complexity of Algorithm 1 is $O(TNc(n+m))$, where c is a constant determined by $K, K', |\mathcal{S}|, |Q|$, and T and N correspond to the numbers of iterations and training tasks, respectively. Similarly, the complexity of the fine-tuning algorithm, Algorithm 2, is $O(sc(n+m))$, where s represents the number of fine-tuning steps. And the time complexity of the test algorithm is $O(c(n+m))$ for a single query. In addition, the space

Table 3: The Profiles of Dataset

Dataset	$\|\mathcal{G}\|$	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	$\|\mathcal{A}\|$	$\|C\|$	graph des.	attribute des.	community des.	# tasks
Arxiv [54]	1	169,343	1,166,243	N/A	40	paper citation	NA	research topics	1,000
Amazon2M [13]	1	2,449,029	61,859,140	N/A	47	product co-purchasing	NA	product categories	5,000
Cora [57]	1	2,708	5,429	1,433	7	paper citation	paper keywords	research topics	192
Citeseer [57]	1	3,327	4,732	3,703	6	paper citation	paper keywords	research topics	192
Reddit [24]	1	232,965	114,615,892	1,164	50	post co-comment	synthetic	post categories	1,000
Facebook [33]	10	4,039	88,234	2,281	193	social friendship	user profiles	friend circles	10
Twitter [33]	973	81,306	1,768,149	512,985	4,065	social friendship	user profiles	friend circles	973

complexity for training (adaptation) and inference are $O(|\mathcal{S}|nd^K + |\mathcal{A}|d^a)$ and $O(nd^K + |\mathcal{A}|d^a)$, respectively. Here, nd^K is the size of the persisted task-level embedding and $|\mathcal{A}|d^a$ is the size of the persisted attribute embedding.

6 EXPERIMENTAL STUDY

We introduce the experimental setup in §6.1 and test our *IACS* in-depth in the following facets: ① Compare the effectiveness of *IACS* on CS and ACS queries with various baselines (§6.2) ② Study the efficiency and scalability of *IACS* (§6.3) ③ Conduct an ablation study to investigate the effect of different GNN layers and aggregate operations (§6.4).④ Investigate the sensitivity of *IACS* regarding parameter configurations (§6.5) ⑤ Explore the capability of model adaptation in task streaming (§6.6) ⑥ Conduct a case study of the visualization of CS results (§6.7).

6.1 Experimental Setup

Datasets. We use 7 real-world graph datasets, whose profiles are summarized in Table 3. Here, 5 of them (Arxiv, Amazon2M, Cora, Citeseer and Reddit) are single graphs, aiming to test model induction across different communities, and the other two datasets (Facebook and Twitter) contain multiple graphs, aiming to test induction across graphs. Notably, Arxiv and Amazon2M are only used for testing non-attributed CS due to the absence of discrete attributes. For Reddit, we generate synthetic attributes following the protocol of [28].

Task & Queries Settings. For single graph datasets, we construct the CS/ACS task by partitioning the graph into disjoint components using METIS algorithm [30]. For Twitter and Facebook, each graph is an ego-centric social network that forms a task. The numbers of total tasks are listed in Table 3. The tasks are approximately split by 60% for training, 10% for validation and 30% for test. In addition, we also investigate model generalization ability across different datasets. One is that models are trained by 128 tasks from Cora and are deployed for inference on 32 tasks from Citeseer (Cora2Citeseer). The other is that models are trained by tasks from Twitter and are deployed for inference on tasks from Facebook (Twitter2Facebook).

For each task, we construct two settings, 4-shot and 8-shot, which use 4 and 8 queries as the support set $\mathcal{S}$ and make the prediction on the other 16 queries, respectively. For each query $q = (\mathcal{V}_q, \mathcal{A}_q)$, we randomly sample $1 \sim 3$ nodes from a community as the query node set $\mathcal{V}_q$. For ACS task, we additionally sample $1 \sim 3$ attributes from the top-3 most frequent attributes in that community as the query attribute set $\mathcal{A}_q$ For training tasks, we sample 5% nodes from C_q and the remaining nodes as the training labels l_q^+ and l_q^-, respectively.

Baselines. To comprehensively evaluate the performance of *IACS*, we compare with 8 baselines that fall into 3 categories, i.e., algorithmic approaches, supervised-learning based approaches and meta-learning based approaches. We briefly introduce them as below.

- *Algorithmic approaches.* ❶ Closest Truss Community (*CTC*) [27] finds the k-truss with the largest k that contains query nodes and has the minimum diameter among the truss. ❷ Attributed Truss Community Search (*ATC*) [28] is an attributed community search algorithm. Given query nodes and query attributes, it finds the maximal (k, d)-truss containing the query nodes and iteratively removes unpromising nodes from the truss to obtain a maximal attribute score. ❸ Attributed Community Query (*ACQ*) [19] aims to search a subgraph whose nodes are tightly connected and share common attributes with the given query nodes.
- *Supervised-learning based approaches.* ❶ Interactive Community Search via GNN (*ICS-GNN*) [21] constructs a GNN model for each query node v, and predicts the scores for the remaining nodes by this model. Subsequently, *ICS-GNN* extracts the subgraph with a fixed number of nodes connecting to v by maximizing the summation of scores. ❷ Query Driven-GNN (*QD/AQD-GNN*) [29] proposes a GNN-based model to combine the representations of the graph, query nodes and possible query attributes. *QD/AQD-GNN* does not have explicit transferability or inductive learning capability. ❸ Supervised GNN (*Supervise*). One GNN model is constructed for each test task from scratch by training on the support set following the baseline in [17]. To support ACS, we extend the baseline by concatenating multi-hot vectors to the input matrix of GNN, which identifying the query nodes and query attributes.
- *Meta-learning based approaches.* We also compare 2 meta-learning based approaches for CS, which are originally proposed in [17]. Similar to *Supervise*, we conduct the same extension for the two approaches to support ACS. ❶ *MAML* adopts Model-Agnostic Meta-Learning algorithm [20] to train a GNN model on all the training tasks. The task-specific parameters of GNN are updated in an inner loop and the task-sharing parameters are updated in an outer loop. ❷ Feature Transfer (*FeatTrans*) trains a GNN model on all the training tasks. For a test task, the final layer of the GNN is fine-tuned on the support set, while all the other parameters are kept intact.

Since *CTC* and *ICS-GNN* do not support attributed queries, we only compare them for CS problem. Moreover, *ACQ* and *ICS-GNN* only

Table 4: Overall Performance on Non-Attributed CS (%)

Dataset	Approach	4-shot Pre	4-shot Rec	4-shot F1	8-shot Pre	8-shot Rec	8-shot F1
Arxiv	*CTC*	$54.23_{\pm0.53}$	$2.16_{\pm0.04}$	$4.15_{\pm0.09}$	$54.04_{\pm0.72}$	$2.16_{\pm0.05}$	$4.15_{\pm0.09}$
	ICS-GNN	$62.72_{\pm0.26}$	$21.09_{\pm0.07}$	$31.57_{\pm0.10}$	$62.53_{\pm0.36}$	$21.12_{\pm0.05}$	$31.57_{\pm0.08}$
	QD-GNN	$59.97_{\pm0.41}$	$83.60_{\pm1.18}$	$69.84_{\pm0.47}$	$58.91_{\pm0.29}$	$89.62_{\pm1.14}$	$71.09_{\pm0.29}$
	Supervise	$67.99_{\pm0.33}$	$69.78_{\pm1.49}$	$68.87_{\pm0.86}$	$69.09_{\pm0.39}$	$74.29_{\pm0.98}$	$71.60_{\pm0.38}$
	MAML	$63.51_{\pm1.07}$	$60.25_{\pm2.50}$	$61.81_{\pm1.42}$	$62.77_{\pm0.72}$	$60.34_{\pm3.64}$	$61.48_{\pm1.82}$
	FeatTrans	$65.35_{\pm0.64}$	$55.18_{\pm1.81}$	$59.81_{\pm0.88}$	$64.18_{\pm0.69}$	$55.42_{\pm1.09}$	$59.47_{\pm0.74}$
	IACS	$63.65_{\pm0.62}$	$89.26_{\pm1.05}$	**$74.31_{\pm0.37}$**	$64.14_{\pm0.49}$	$90.21_{\pm1.31}$	**$74.97_{\pm0.31}$**
	IACS-G	$59.75_{\pm0.42}$	$97.99_{\pm0.76}$	$74.23_{\pm0.13}$	$65.06_{\pm0.81}$	$88.12_{\pm1.65}$	$74.84_{\pm0.36}$
	IACS-P	$61.99_{\pm2.56}$	$92.63_{\pm5.77}$	$74.12_{\pm0.21}$	$65.45_{\pm0.42}$	$87.07_{\pm0.53}$	$74.72_{\pm0.24}$
Amazon2M	*CTC*	$80.30_{\pm0.35}$	$4.06_{\pm0.02}$	$7.73_{\pm0.04}$	$80.27_{\pm0.27}$	$4.06_{\pm0.01}$	$7.73_{\pm0.02}$
	ICS-GNN	$79.50_{\pm0.27}$	$6.55_{\pm0.01}$	$12.11_{\pm0.02}$	$79.63_{\pm0.29}$	$6.55_{\pm0.02}$	$12.11_{\pm0.03}$
	QD-GNN	$75.46_{\pm0.33}$	$95.15_{\pm0.53}$	$84.17_{\pm0.04}$	$75.33_{\pm0.26}$	$96.68_{\pm0.13}$	$84.67_{\pm0.21}$
	Supervise	$83.86_{\pm0.09}$	$77.07_{\pm0.44}$	$80.32_{\pm0.25}$	$84.46_{\pm0.35}$	$80.18_{\pm0.52}$	$82.27_{\pm0.29}$
	MAML	$78.48_{\pm1.62}$	$65.83_{\pm8.70}$	$71.38_{\pm5.59}$	$79.13_{\pm0.88}$	$62.38_{\pm4.76}$	$69.66_{\pm2.83}$
	FeatTrans	$78.41_{\pm0.92}$	$57.89_{\pm1.39}$	$66.60_{\pm1.14}$	$78.69_{\pm0.34}$	$57.18_{\pm1.22}$	$66.22_{\pm0.72}$
	IACS	$80.52_{\pm0.34}$	$93.42_{\pm0.83}$	$86.48_{\pm0.22}$	$81.44_{\pm0.75}$	$93.34_{\pm1.07}$	$86.97_{\pm0.21}$
	IACS-G	$79.92_{\pm0.31}$	$94.25_{\pm0.93}$	$86.49_{\pm0.21}$	$80.49_{\pm0.16}$	$94.60_{\pm0.52}$	$86.98_{\pm0.24}$
	IACS-P	$79.63_{\pm0.88}$	$94.77_{\pm1.09}$	**$86.54_{\pm0.29}$**	$80.62_{\pm0.81}$	$94.86_{\pm0.74}$	**$87.16_{\pm0.26}$**

Table 5: Overall Performance on ACS in Single Graph (%)

Dataset	Approach	4-shot Pre	4-shot Rec	4-shot F1	8-shot Pre	8-shot Rec	8-shot F1
Citeseer	*ATC*	$58.99_{\pm0.87}$	$5.01_{\pm0.17}$	$9.24_{\pm0.29}$	$57.83_{\pm0.36}$	$4.97_{\pm0.25}$	$9.16_{\pm0.42}$
	ACQ	$70.59_{\pm2.14}$	$6.97_{\pm1.32}$	$12.66_{\pm2.19}$	$69.15_{\pm1.43}$	$6.79_{\pm0.99}$	$12.36_{\pm1.64}$
	AQD-GNN	$52.70_{\pm1.04}$	$84.29_{\pm7.59}$	$64.77_{\pm2.78}$	$52.26_{\pm2.06}$	$85.15_{\pm5.57}$	$64.74_{\pm3.01}$
	Supervise	$60.45_{\pm2.00}$	$63.20_{\pm1.53}$	$61.79_{\pm1.67}$	$62.30_{\pm1.88}$	$66.74_{\pm0.84}$	$64.43_{\pm1.16}$
	MAML	$55.53_{\pm1.61}$	$43.18_{\pm4.27}$	$48.46_{\pm2.45}$	$56.15_{\pm0.94}$	$45.02_{\pm3.05}$	$49.93_{\pm1.94}$
	FeatTrans	$58.67_{\pm2.54}$	$37.97_{\pm1.38}$	$46.08_{\pm1.51}$	$58.44_{\pm1.74}$	$39.86_{\pm2.17}$	$47.38_{\pm1.91}$
	IACS	$64.74_{\pm1.55}$	$71.15_{\pm1.45}$	$67.78_{\pm0.94}$	$67.06_{\pm1.45}$	$71.48_{\pm1.84}$	$69.19_{\pm1.41}$
	IACS-G	$65.75_{\pm0.54}$	$70.12_{\pm1.33}$	**$67.86_{\pm0.56}$**	$67.48_{\pm1.62}$	$71.90_{\pm1.98}$	**$69.59_{\pm0.87}$**
	IACS-P	$65.52_{\pm1.15}$	$70.37_{\pm1.43}$	$67.84_{\pm0.66}$	$67.25_{\pm1.86}$	$72.08_{\pm0.74}$	$69.57_{\pm0.98}$
Reddit	*ATC*	$82.84_{\pm0.77}$	$39.15_{\pm1.68}$	$53.16_{\pm1.63}$	$83.73_{\pm0.87}$	$39.00_{\pm0.91}$	$53.21_{\pm0.79}$
	ACQ	$97.74_{\pm0.27}$	$22.82_{\pm1.02}$	$36.99_{\pm1.35}$	$98.16_{\pm0.03}$	$22.49_{\pm1.20}$	$36.58_{\pm1.59}$
	AQD-GNN	$85.88_{\pm1.63}$	$89.54_{\pm1.89}$	**$87.67_{\pm1.38}$**	$85.51_{\pm1.73}$	$92.20_{\pm1.39}$	**$88.73_{\pm1.42}$**
	Supervise	$86.16_{\pm1.38}$	$78.14_{\pm1.93}$	$81.95_{\pm1.51}$	$87.14_{\pm0.95}$	$80.15_{\pm0.49}$	$83.50_{\pm0.49}$
	MAML	$88.30_{\pm0.78}$	$64.46_{\pm3.15}$	$74.66_{\pm2.27}$	OOM	OOM	OOM
	FeatTrans	$87.76_{\pm2.41}$	$34.78_{\pm3.22}$	$49.72_{\pm3.34}$	$88.07_{\pm0.59}$	$36.46_{\pm2.43}$	$51.53_{\pm2.40}$
	IACS	$84.03_{\pm1.20}$	$84.11_{\pm3.84}$	$84.01_{\pm1.26}$	$83.50_{\pm1.69}$	$85.07_{\pm4.36}$	$84.20_{\pm1.49}$
	IACS-G	$83.81_{\pm1.73}$	$85.33_{\pm2.23}$	$84.54_{\pm1.14}$	$86.07_{\pm0.86}$	$84.01_{\pm2.63}$	$85.00_{\pm1.11}$
	IACS-P	$84.85_{\pm1.28}$	$83.90_{\pm2.53}$	$84.35_{\pm1.31}$	$85.89_{\pm1.84}$	$83.21_{\pm1.72}$	$84.51_{\pm1.33}$

support single query node, thereby we randomly sample one query node when $|\mathcal{V}_q| > 1$. All the approaches are tested on the same set of queries to achieve a fair comparison.

Implementation & Parameter Setting. For the GNN encoder of *IACS*, we try *GCN* [31], *GraphSAGE* [24], *GIN* [56] and *GAT* [51], and use *GAT* as the default GNN model. We set the number of the GNN layers as 3, and each GNN layer has 128 hidden units. All the learning-based baselines follow the same GNN configurations. We use ProNE [61] to generate the 128-dim attributed embedding from the attribute-augmented graph. For the decoder of *IACS*, the number of hidden units in the FiLM module and MLP is set to 128.

The learning framework of *IACS* is built on PyTorch [2] with PyTorch Geometric [3]. The models are trained by Adam optimizer for 200 epochs over the training task set, and are fine-tuned for 10 steps on each test task by default, with a learning rate of 0.001. It is worth mentioning that the performance of *IACS* is robust in the empirical intervals of training hyper-parameters. For other learning-based baselines, the training hyper-parameters are kept as their default configurations. The experiments of all the learning-based approaches are conducted on a Tesla A100 with 80GB memory. The algorithmic approaches are tested on the same Linux server with 96 AMD EPYC 7413 CPUs and 512GB RAM.

Evaluation Metrics. To evaluate the quality of the searched communities, we use precision, recall and F1-score between the prediction and the ground-truth as quantitative metrics. F1-score is the

Table 6: Overall Performance on ACS in Multiple Graphs (%)

Dataset	Approach	4-shot Pre	4-shot Rec	4-shot F1	8-shot Pre	8-shot Rec	8-shot F1
Facebook	*ATC*	$60.23_{\pm5.10}$	$11.99_{\pm0.88}$	$19.97_{\pm1.26}$	$41.14_{\pm4.18}$	$11.11_{\pm3.27}$	$17.22_{\pm3.71}$
	ACQ	$38.86_{\pm3.52}$	$66.92_{\pm5.79}$	$48.90_{\pm1.92}$	$40.60_{\pm3.06}$	$64.00_{\pm3.33}$	$49.65_{\pm3.07}$
	AQD-GNN	$37.71_{\pm1.65}$	$96.70_{\pm5.93}$	$54.26_{\pm2.59}$	$36.71_{\pm1.03}$	$96.29_{\pm4.68}$	$53.14_{\pm1.75}$
	Supervise	$59.32_{\pm1.22}$	$79.61_{\pm5.37}$	$67.95_{\pm2.73}$	$64.34_{\pm1.63}$	$83.38_{\pm3.20}$	$72.59_{\pm1.08}$
	MAML	$47.06_{\pm6.12}$	$89.04_{\pm3.86}$	$59.85_{\pm8.12}$	$46.12_{\pm3.30}$	$73.30_{\pm5.89}$	$56.44_{\pm2.45}$
	FeatTrans	$50.11_{\pm6.62}$	$68.34_{\pm8.66}$	$56.84_{\pm4.28}$	$50.82_{\pm1.16}$	$59.77_{\pm6.87}$	$72.16_{\pm3.40}$
	IACS	$85.61_{\pm2.16}$	$79.55_{\pm4.13}$	**$78.09_{\pm4.09}$**	$66.13_{\pm1.62}$	$84.33_{\pm3.80}$	$74.08_{\pm1.34}$
	IACS-G	$85.75_{\pm2.27}$	$81.31_{\pm4.51}$	$77.42_{\pm3.82}$	$64.87_{\pm1.67}$	$88.17_{\pm3.04}$	$74.72_{\pm1.64}$
	IACS-P	$81.82_{\pm4.42}$	$80.79_{\pm2.69}$	$77.37_{\pm4.62}$	$65.92_{\pm4.20}$	$87.36_{\pm1.66}$	**$75.05_{\pm2.31}$**
Twitter	*ATC*	$32.18_{\pm1.27}$	$13.98_{\pm0.93}$	$19.49_{\pm1.13}$	$32.28_{\pm3.87}$	$13.9_{\pm1.24}$	$19.42_{\pm1.86}$
	ACQ	$42.71_{\pm1.08}$	$10.79_{\pm0.42}$	$17.23_{\pm0.61}$	$47.69_{\pm3.33}$	$9.45_{\pm1.11}$	$15.76_{\pm1.62}$
	AQD-GNN	$28.84_{\pm0.37}$	$85.50_{\pm3.88}$	$43.11_{\pm0.68}$	$29.22_{\pm0.60}$	$91.65_{\pm2.09}$	$44.30_{\pm0.61}$
	Supervise	$36.96_{\pm0.75}$	$69.30_{\pm2.18}$	$48.20_{\pm1.09}$	$40.64_{\pm1.31}$	$73.98_{\pm2.87}$	$52.46_{\pm1.68}$
	MAML	$25.31_{\pm1.74}$	$42.67_{\pm3.92}$	$31.76_{\pm2.42}$	$27.06_{\pm3.38}$	$46.80_{\pm1.38}$	$34.22_{\pm2.88}$
	FeatTrans	$29.46_{\pm0.78}$	$39.66_{\pm1.87}$	$33.79_{\pm1.03}$	$29.31_{\pm0.68}$	$40.61_{\pm1.71}$	$34.03_{\pm0.72}$
	IACS	$48.74_{\pm2.22}$	$65.23_{\pm0.56}$	$55.77_{\pm1.47}$	$53.28_{\pm1.88}$	$71.04_{\pm1.30}$	$60.88_{\pm1.45}$
	IACS-G	$49.91_{\pm2.32}$	$65.36_{\pm1.94}$	**$56.57_{\pm1.67}$**	$54.95_{\pm1.29}$	$71.13_{\pm1.31}$	**$61.99_{\pm1.04}$**
	IACS-P	$49.97_{\pm2.64}$	$65.15_{\pm0.98}$	$56.52_{\pm1.58}$	$54.63_{\pm1.30}$	$71.57_{\pm1.45}$	$61.96_{\pm1.19}$
Cora2Citeseer	*ATC*	$59.44_{\pm2.88}$	$4.94_{\pm0.14}$	$9.12_{\pm0.22}$	$60.64_{\pm3.32}$	$5.13_{\pm0.31}$	$9.45_{\pm0.54}$
	ACQ	$70.90_{\pm1.98}$	$6.31_{\pm0.70}$	$11.58_{\pm1.18}$	$73.81_{\pm2.64}$	$6.79_{\pm0.94}$	$12.43_{\pm1.60}$
	AQD-GNN	$52.75_{\pm3.80}$	$81.87_{\pm8.71}$	$64.01_{\pm4.42}$	$52.91_{\pm3.05}$	$83.50_{\pm4.13}$	$64.76_{\pm3.33}$
	Supervise	$59.61_{\pm3.39}$	$61.67_{\pm2.29}$	$60.60_{\pm2.72}$	$62.81_{\pm3.44}$	$66.29_{\pm1.52}$	$64.47_{\pm2.21}$
	MAML	$55.55_{\pm5.22}$	$45.79_{\pm3.14}$	$43.87_{\pm3.57}$	$55.49_{\pm3.80}$	$70.52_{\pm8.40}$	$60.43_{\pm0.81}$
	FeatTrans	$60.09_{\pm4.43}$	$30.71_{\pm6.18}$	$40.29_{\pm5.65}$	$62.24_{\pm5.94}$	$31.77_{\pm6.48}$	$41.76_{\pm6.56}$
	IACS	$62.12_{\pm4.96}$	$65.49_{\pm4.42}$	$63.60_{\pm3.28}$	$66.46_{\pm3.47}$	$70.43_{\pm1.91}$	$68.36_{\pm2.49}$
	IACS-G	$63.68_{\pm2.46}$	$65.89_{\pm3.32}$	**$64.76_{\pm2.84}$**	$66.60_{\pm3.84}$	$71.89_{\pm2.48}$	**$69.07_{\pm2.28}$**
	IACS-P	$62.69_{\pm4.62}$	$66.14_{\pm2.94}$	$64.33_{\pm3.55}$	$66.14_{\pm3.97}$	$71.99_{\pm3.04}$	$68.89_{\pm2.89}$
Twitter2Facebook	*ATC*	$77.92_{\pm6.75}$	$12.88_{\pm0.74}$	$22.08_{\pm1.04}$	$70.30_{\pm9.71}$	$11.25_{\pm1.91}$	$19.35_{\pm3.05}$
	ACQ	$68.89_{\pm0.81}$	$37.21_{\pm8.31}$	$49.51_{\pm4.76}$	$20.38_{\pm0.58}$	$44.58_{\pm3.51}$	$28.22_{\pm0.47}$
	AQD-GNN	$37.49_{\pm0.49}$	$96.81_{\pm3.49}$	$37.49_{\pm1.03}$	$37.61_{\pm3.45}$	$95.10_{\pm8.97}$	$53.84_{\pm4.54}$
	Supervise	$58.12_{\pm4.20}$	$80.28_{\pm8.12}$	$58.12_{\pm5.27}$	$62.32_{\pm4.41}$	$81.44_{\pm4.99}$	$70.55_{\pm4.15}$
	MAML	$38.12_{\pm0.42}$	$97.57_{\pm1.67}$	$38.12_{\pm0.37}$	$37.97_{\pm1.58}$	$95.01_{\pm4.40}$	$54.20_{\pm1.19}$
	FeatTrans	$38.45_{\pm0.42}$	$98.05_{\pm1.04}$	$38.45_{\pm0.43}$	$38.20_{\pm0.86}$	$97.06_{\pm1.39}$	$54.81_{\pm0.72}$
	IACS	$72.10_{\pm5.43}$	$85.44_{\pm4.30}$	$72.10_{\pm1.58}$	$66.29_{\pm3.99}$	$84.90_{\pm0.58}$	$73.68_{\pm2.88}$
	IACS-G	$67.76_{\pm3.00}$	$86.17_{\pm1.65}$	$67.76_{\pm1.80}$	$64.86_{\pm1.81}$	$86.36_{\pm3.48}$	$74.05_{\pm1.94}$
	IACS-P	$72.38_{\pm4.80}$	$83.69_{\pm6.53}$	$72.38_{\pm1.75}$	$65.14_{\pm0.85}$	$86.07_{\pm2.36}$	$74.28_{\pm1.19}$

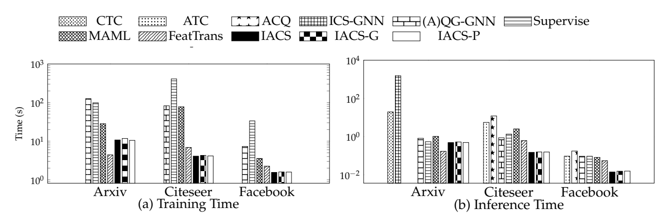


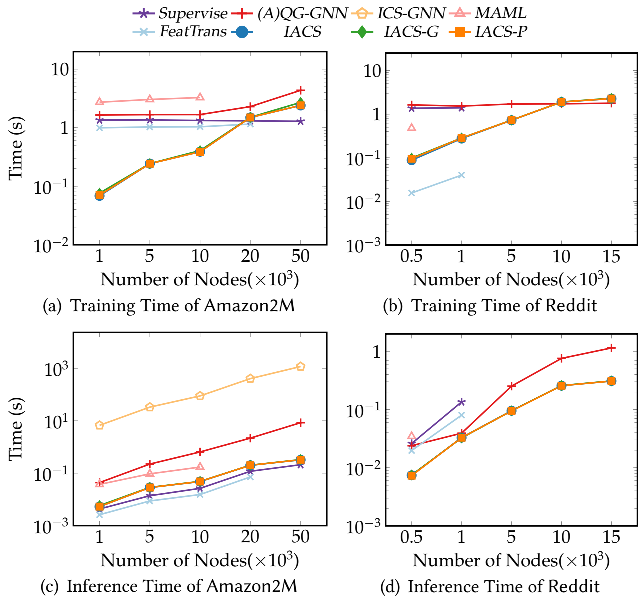
Figure 5: Comparison of Training and Inference Time

harmonic average of precision and recall, which better reflects the overall performance.

6.2 Overall Effectiveness

We discuss the overall performance of *IACS* for both non-attributed CS and ACS, focusing on 4-shot and 8-shot learning settings. Recall that the number of shots refers to the number of queries in the support set $\mathcal{S}$. We compare three variants of *IACS*, i.e., *IACS* without FiLM (*IACS*), *IACS* with plain FiLM (*IACS-P*) (Eq. (9)) and *IACS* with gating FiLM mechanism (*IACS-G*) (Eq. (10)) against the 8 baselines. To ensure statistical robustness, we conduct each experiment 5 times by varying random seeds, and report the mean and the standard deviation in Table 4-6.

Table 4 presents the results on non-attributed CS, where the first and second best F1 scores are highlighted and underlined, respectively. Our observations illustrate that *IACS* models consistently outperform all the baselines. The superiority of *IACS* is primarily evident in its significant improvement in recall (+1.28% compared to the best baseline) while maintaining a relatively high precision (59.75% ∼ 81.44%). *CTC* and *ICS-GNN* exhibit unsatisfactory results.



(a) Training Time of Amazon2M (b) Training Time of Reddit

(c) Inference Time of Amazon2M (d) Inference Time of Reddit

Figure 6: Scalability Test for One Task (Second)

Precisely, *CTC* struggles to identify the inflexible community patterns, resulting in lower recall (2.16% ~ 4.06%) compared to other approaches. That limits its ability to identify all relevant nodes in the community. *ICS-GNN* cannot support multi-node queries; thus, we randomly select one node from the query node set in our experiment, which may also lead to inaccurate results. *QD-GNN* and *Supervise* outperform the meta-learning based baselines, which may be because the meta-learning baselines have a weak generalization ability and the common knowledge they learned has negative transfer impact on new tasks.

Table 5-6 show the performance on ACS. Note that in Table 5, "OOM" indicates the GPU runs out-of-memory. In general, *IACS* achieves the highest F1 score in most cases (5 out of 6), even when the graphs of training and inference are from different datasets, i.e., Cora2Citeseer. *ATC* and *ACQ* exhibit poor performance due to their low recall. Because of the inflexible two-stage search process, they retrieve a limited number of nodes in communities. *AQD-GNN* shows comparatively better accuracy, especially on Reddit. However, due to the limited inductive capability, we need to retrain the model for each new task in order to acquire sufficient task-specific knowledge. We speculate that the community pattern or attribute distribution across heterogeneous tasks differs on Reddit, making the common knowledge less beneficial for new tasks. The supervised-learning based approaches still perform better than meta-learning approaches, which is similar to the results of non-attributed CS. We further verify the generalization ability for more datasets, where the training and inference datasets are different or from different domains, and the results are reported in our online appendix [1].

Additionally, it is worth noting that the improvement in the 8-shot setting is relatively lower compared to the 4-shot setting. The reason lies in that our framework is particularly well-suited for tasks involving only a small number of queries with ground-truth. For the three variants of *IACS*, *IACS-G* performs better than other two models in most cases of ACS tasks, which provides empirical evidence for the effectiveness of the gating mechanism in the FiLM module. In contrast, *IACS* and *IACS-P* tend to perform better in non-attributed CS tasks.

6.3 Efficiency & Scalability

In this section, we evaluate the GPU training/inference time for the three *IACS* models of *IACS* against all the baselines. Here, the training time we reported for all the learning-based approaches is for one epoch. And the inference time corresponds to the cost of the model inference phase. It is noteworthy that the inference time of *ICS-GNN* contains the training time due to its online learning design. Fig. 5 presents the comparison of the training and inference time on three datasets. We report the comprehensive results across all the datasets in our online appendix [1].

In general, *IACS* models exhibit faster training and inference time compared to other baselines, except Arxiv, where *IACS* fails to surpass the efficiency of *FeatTrans*, because of the simple design of *FeatTrans*. *ICS-GNN* costs the longest time, since it needs to retrain a model for each query. *CTC* also spends longer time to compute the diameter and maintain the k-truss structure, especially when dealing with large candidate communities such as Arxiv. For ACS, the three *IACS* models are the most time-efficient approaches. On the contrary, the 2 algorithmic approaches *ATC* and *ACQ*, exhibit the longest computation time due to their two-stage process. *MAML*, *Supervise* and *FeatTrans* require concatenating additional multi-hot vectors representing query attributes to the input matrix of the GNN, leading to lower efficiency compared to *IACS*. Regarding *QD/AQD-GNN*, the usage of extra learning modules such as the query encoder and attribute encoder prevents it from outperforming *IACS* for both non-attributed and attributed CS tasks.

The three *IACS* models share similar training and inference costs, with *IACS-G* requiring slightly longer time. That is because *IACS-G* incorporates an additional gating mechanism in the FiLM module. Regarding model selection for a specific dataset, *IACS* is the optimal choice when efficiency is the principal factor. In summary, our models outperform other baselines by delivering superior overall effectiveness while maintaining competitive efficiency.

Scalability. We test the scalability of *IACS* models and other learning-based approaches in Fig. 6, which illustrates the GPU training and inference time for a single task as the number of nodes in the graph increases. The default data point in Fig. 6 indicates that GPU runs "OOM". Fig. 6(a) and 6(c) show the results for non-attributed CS, where the scalability of *IACS* is not able to surpass that of *Supervise* in graphs with 50, 000 nodes, but it is considerably better than *ICS-GNN*, *QD-GNN* and *MAML*, regardless of the number of nodes. *MAML* and *FeatTrans* fail to scale to graphs with 50, 000 nodes. In Fig. 6(b) and 6(d) for ACS, only *AQD-GNN* and *IACS* can scale to graphs with 15, 000 nodes efficiently. This phenomenon can be attributed to the fact that the other three approaches involve the concatenation of multi-hot vectors to the input matrix, which incurs extra computation and space overhead. The training time of *AQD-GNN* remains stable and becomes faster than *IACS* in 15, 000 nodes. However, *IACS* exhibits significantly faster inference time than *AQD-GNN*, especially when dealing with larger graphs.

Table 7: Performance with Different GNN Layers and Query Feature Fusion Modules (%)

Module	Arxiv			Citeseer			Twitter			Twitter2Facebook			Cora2Citeseer		
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
GAT	63.58	92.37	75.32	62.88	70.18	66.33	51.25	64.97	57.30	71.82	84.91	77.82	64.73	60.32	62.45
GCN	63.40	88.32	73.81	63.97	59.33	61.56	28.72	74.63	41.47	80.94	77.32	**79.09**	63.67	58.37	60.90
GraphSAGE	64.95	87.83	74.68	62.49	60.79	61.63	60.34	57.10	**58.67**	79.17	78.28	78.72	65.28	57.68	61.24
GATBias	64.01	91.69	**75.39**	63.92	69.16	66.43	51.49	65.81	57.77	69.60	88.63	77.97	62.84	63.64	63.24
GIN	61.35	90.35	73.08	66.56	69.28	**67.89**	37.25	60.66	46.15	71.00	74.29	72.61	63.31	69.59	**66.30**
Sum	Not Available			50.15	99.93	66.78	28.58	93.16	43.74	37.93	100.00	54.99	55.22	91.78	**68.95**
Multiplication				51.35	96.56	**67.05**	28.84	53.10	37.38	38.12	99.97	55.20	53.04	97.66	68.75
Concatenation				63.92	69.16	66.43	51.49	65.81	**57.77**	69.60	88.63	**77.97**	62.84	63.64	63.24

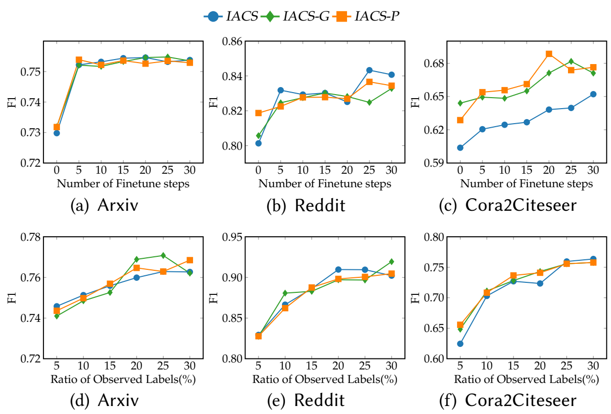


Figure 7: F1 Score of *IACS* under Different Hyper-parameters

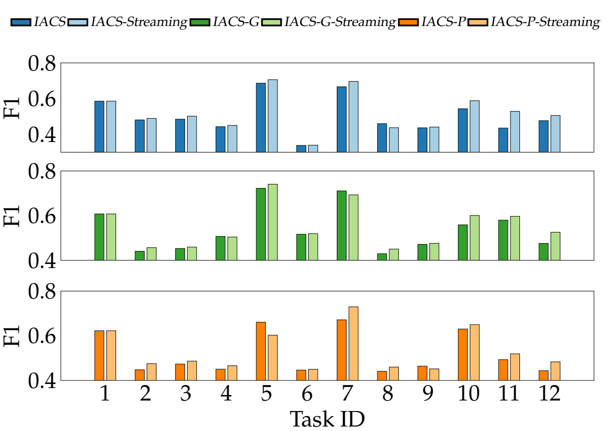


Figure 8: Streaming Model Adaptation on Twitter

6.4 Ablation Studies

We investigate the performance of different GNN layers in the encoder, and fusion operations in Eq. (12) for incorporating query node and query attribute embeddings. Table 7 presents the performance of corresponding *IACS* variants on 5 datasets. For different GNN layers, we test *GAT*, *GCN*, *GraphSAGE*, *GATBias* and *GIN*, by fixing the fusion operation as vector concatenation. *GATBias* incorporates an additional attention bias, the shortest distances between each node and query nodes, into the attention weights for message passing. Regarding the best and the second-best scores, in general, *GATBias* shows competitive performance among all the datasets, and particularly performs better in non-attributed CS tasks. *GATBias* not only applies self-attention to weigh the node representations based on the neighbors' importance, but also explicitly injects query-specific knowledge prior into the attention mechanism. Specially, *GIN* achieves better performance in citation networks for ACS tasks. *GIN* is known for its powerful expressive capability, allowing it to generate highly effective representations for distinguishing the membership of nodes in subgraph patterns. It is worth mentioning that the superior performance of our framework does not rely on the specific choice of GNN.

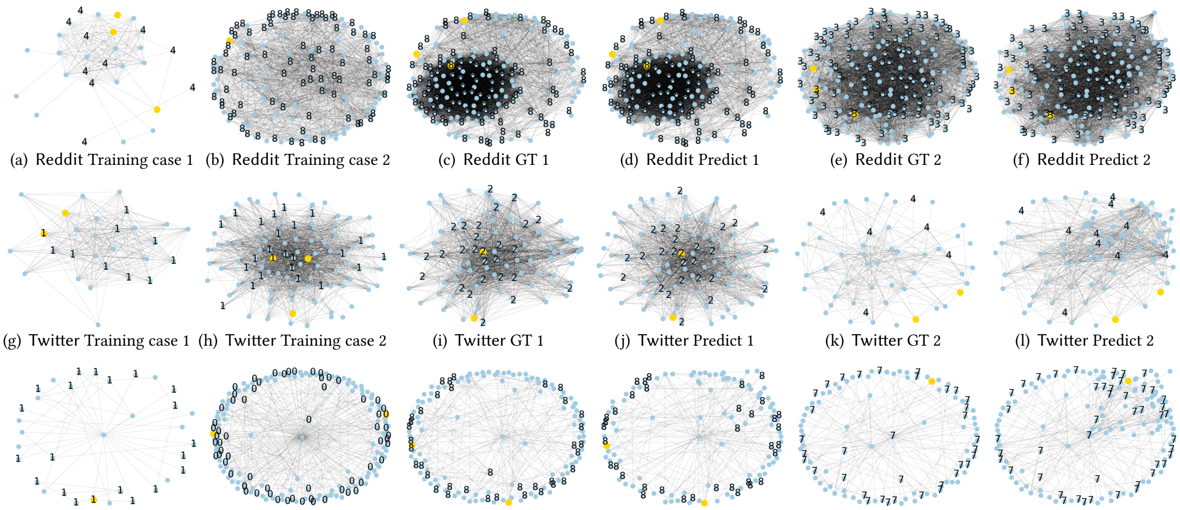
In addition, to test the impact of different fusion operations, we introduce vector summation and element-wise multiplication operations, i.e., $e_{\mathcal{V}_q} + e_{\mathcal{A}_q}$ and $e_{\mathcal{V}_q} \odot e_{\mathcal{A}_q}$, as alternatives to the vector concatenation in Eq. (12). Here, we keep *GATBias* as the GNN layer in the encoder. Since non-attributed CS (Arxiv) does not require fusing the query attribute embedding, the corresponding results are not available in Table 7. Our testing results reveal that the 3 operations yield competitive results. It should be noted that the optimal choice of aggregation operation may vary from the dataset, as different datasets may benefit from different aggregation approaches.

6.5 Parameter Analysis

We study the parameter sensitivity of *IACS* w.r.t. two key hyper-parameters: (1) the number of fine-tuning steps and (2) the ratio of training labels. Fig. 7 shows how the hyper-parameters affect the F1 of *IACS* models for both non-attributed CS (Arxiv) and ACS (Reddit and Cora2Citeseer).

Number of Fine-tuning Steps. Fig. 7(a)-(c) illustrate the F1 scores achieved with different numbers of fine-tuning steps. The step 0 indicates that models are not fine-tuned via an adaptation phase. We find that the model without fine-tuning cannot achieve comparable results to those achieved by the fine-tuned models. In Reddit (Fig. 7(b)) and Cora2Citeseer (Fig. 7(c)), we observe a remarkable performance improvement as the number of fine-tuning steps increases. However, it is worth noting that in some cases, a large number of fine-tune steps can lead to over-fitting, thereby degenerating the model's accuracy. For non-attributed CS, i.e., Arxiv (Fig. 7(a)), we observe that the performance is less sensitive to the number of fine-tuning steps compared to other datasets. In general, we find that using 30 fine-tuning steps tends to achieve the overall best performance.

Ratio of Training Labels. In Fig. 7(d)-(f), we present the F1 scores obtained under different ratios of training labels, which are vary in the range of 5% ~ 30% of the total number of the nodes, for both positive and negative labels. There is an obvious increasing tendency for the three *IACS* models. This trend is intuitive since



(a) Reddit Training case 1 (b) Reddit Training case 2 (c) Reddit GT 1 (d) Reddit Predict 1 (e) Reddit GT 2 (f) Reddit Predict 2

(g) Twitter Training case 1 (h) Twitter Training case 2 (i) Twitter GT 1 (j) Twitter Predict 1 (k) Twitter GT 2 (l) Twitter Predict 2

(m) Cora Training case 1 (n) Cora Training case 2 (o) Citeseer GT 1 (p) Citeseer Predict 1 (q) Citeseer GT 2 (r) Citeseer Predict 2

Figure 9: Visualization of Training, Ground-Truth (GT) and Predicted Community for Reddit**,** Twitter **and** Cora2Citeseer**: The yellows nodes are query nodes and the numbers are query attributes.**

a higher ratio of training labels indicates more prior knowledge available for the model to train. However, there are special cases where the F1 scores decline as the ratio increases. This can be attributed to over-fitting, which can adversely affect the model's performance. In general, a ratio of 30% observed labels tends to achieve the best overall performance. However, it is worth noting that even with a small number of labels, i.e., 5%, *IACS* models can still yield high F1 scores. These results validate the effectiveness of *IACS* in addressing CS/ACS when training data is scarce.

6.6 Streaming Model Adaptation

To explore the adaptability *IACS*, we conduct an experiment to test streaming *IACS* models which are continuously fine-tuned by steaming ACS tasks of Twitter. Fig. 8 presents the 3 *IACS* model variants, compared with their corresponding original models which are fine-tuned by a single task.

As depicted in Fig. 8, the results indicate that three *IACS* models exhibit an improvement ratio of 3% in the streaming adaptation model. The efficacy of the *IACS* stems from its ability to preserve crucial common knowledge among the ACS tasks during the adaptation process. The streaming adaptation process can benefit from this property, leading to higher F1 scores for the streaming model than the original model across a wide range of sequential tasks. In special cases such as Task 5 in *IACS-P*, we observe that the streaming model fails to surpass the original model. We speculate that the performance gap may derive from the discrepancy of graph structures and attribute distributions between Task 5 and the previous tasks. The process of adapting the *IACS-P* to Task 5 may disrupt the previously learned parameters, leading to a negative impact on performance. In summary, the results obtained from the streaming model adaptation verify the distinct adaptation capability of *IACS*.

6.7 Case Study

We conduct a case study to investigate the inductive generalization ability of *IACS* on new communities and graphs. In Fig. 9, we visualize the different patterns of communities that the model has learned, and the prediction results on three datasets, Twitter, Reddit and Cora2Citeseer. The yellow nodes are the query nodes, while the numbers in the nodes represent their corresponding attributes. To enhance the presentation clarity, we only display the query attributes in the figures.

For each row, the 2 figures on the left show distinct communities in the training graphs. As we can observe, these communities exhibit variations, characterized by heterogeneous topological structures and attribute distributions. The four figures on the right depict ground-truth communities and the corresponding predicted communities in the test tasks. We observe a notable overlap between the identified communities and the ground-truth, thus confirming the accuracy of our predictions. Our model demonstrates a remarkable inductive ability by effectively extracting data heterogeneity across multiple training tasks and adapting to new communities and graphs.

7 CONCLUSION

In this paper, we explore a new inductive learning framework called *IACS* to effectively perform CS and ACS across heterogeneous graphs. By leveraging three-phase workflow and encoder-decoder neural architecture, *IACS* overcome the limitations of the prior transductive and algorithmic methods. Experimental results in 7 real-world datasets demonstrated its ability to comprehensively support complex queries while adapting well to new graphs for both CS and ACS. *IACS* outperforms other baselines with higher F1 scores by 28.97% and 25.60% on average in CS and ACS, respectively. The source code and full version have been made available at https://github.com/FangShuheng/IACS.

ACKNOWLEDGMENTS

This work was supported by the Research Grants Council of Hong Kong, China, No. 14203618, No. 14202919 and No. 14205520. Kangfei Zhao is supported by National Key Research and Development Plan, No. 2023YFF0725101.

REFERENCES

[1] [n.d.]. Code and appendix of IACS. https://github.com/FangShuheng/IACS.

[2] [n.d.]. Pytorch. https://github.com/pytorch/pytorch.

[3] [n.d.]. Pytorch Geometric. https://github.com/rusty1s/pytorch_geometric.

[4] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. 2019. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the twelfth ACM international conference on web search and data mining*. 384–392.

[5] Yunsheng Bai, Hao Ding, Yang Qiao, Agustin Marinovic, Ken Gu, Ting Chen, Yizhou Sun, and Wei Wang. 2019. Unsupervised inductive graph-level representation learning via graph-graph proximity. *arXiv preprint arXiv:1904.01098* (2019).

[6] Yunsheng Bai, Derek Xu, Yizhou Sun, and Wei Wang. 2021. Glsearch: Maximum common subgraph detection via learning to search. In *International Conference on Machine Learning*. PMLR, 588–598.

[7] Peyman Bateni, Raghav Goyal, Vaden Masrani, Frank Wood, and Leonid Sigal. 2020. Improved few-shot visual classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14493–14502.

[8] Tian Bian, Xi Xiao, Tingyang Xu, Peilin Zhao, Wenbing Huang, Yu Rong, and Junzhou Huang. 2020. Rumor detection on social media with bi-directional graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 549–556.

[9] Marc Brockschmidt. 2020. Gnn-film: Graph neural networks with feature-wise linear modulation. In *International Conference on Machine Learning*. PMLR, 1144–1152.

[10] Quentin Cappart, Didier Chételat, Elias B Khalil, Andrea Lodi, Christopher Morris, and Petar Velickovic. 2023. Combinatorial optimization and reasoning with graph neural networks. *J. Mach. Learn. Res.* 24 (2023), 130–1.

[11] Jiazun Chen, Yikuan Xia, and Jun Gao. 2023. CommunityAF: An Example-Based Community Search Method via Autoregressive Flow. *Proceedings of the VLDB Endowment* 16, 10 (2023), 2565–2577.

[12] Zhengdao Chen, Xiang Li, and Joan Bruna. 2017. Supervised community detection with line graph neural networks. *arXiv preprint arXiv:1705.08415* (2017).

[13] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 257–266.

[14] Daniel Daza, Michael Cochez, and Paul Groth. 2021. Inductive entity representations from text via link prediction. In *Proceedings of the Web Conference 2021*. 798–808.

[15] Chi Thang Duong, Trung Dung Hoang, Hongzhi Yin, Matthias Weidlich, Quoc Viet Hung Nguyen, and Karl Aberer. 2021. Efficient streaming subgraph isomorphism with graph neural networks. *Proceedings of the VLDB Endowment* 14, 5 (2021), 730–742.

[16] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. 2018. Neural scene representation and rendering. *Science* 360, 6394 (2018), 1204–1210.

[17] 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)*. IEEE, 2358–2371.

[18] Shuheng Fang, Kangfei Zhao, Yu Rong, Zhixun Li, and Jeffrey Xu Yu. 2024. All-in-One: Heterogeneous Interaction Modeling for Cold-Start Rating Prediction. *arXiv preprint arXiv:2403.17740* (2024).

[19] Yixiang Fang, CK Cheng, Siqiang Luo, and Jiafeng Hu. 2016. Effective community search for large attributed graphs. *Proceedings of the VLDB Endowment* (2016).

[20] 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.

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

[22] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. 2019. Exact combinatorial optimization with graph convolutional neural networks. *Advances in neural information processing systems* 32 (2019).

[23] Fangda Guo, Ye Yuan, Guoren Wang, Xiangguo Zhao, and Hao Sun. 2021. Multi-attributed community search in road-social networks. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 109–120.

[24] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[25] Yu Hao, Xin Cao, Yixiang Fang, Xike Xie, and Sibo Wang. 2020. Inductive link prediction for nodes having only attribute information. *arXiv preprint arXiv:2007.08053* (2020).

[26] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. 2018. Adaptive sampling towards fast graph representation learning. *Advances in neural information processing systems* 31 (2018).

[27] Xin Huang, Laks VS Lakshmanan, Jeffrey Xu Yu, and Hong Cheng. 2015. Approximate closest community search in networks. *arXiv preprint arXiv:1505.05956* (2015).

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

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

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

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

[32] Juyong Lee and Jooyoung Lee. 2013. Hidden information revealed by optimal community structure from a protein-complex bipartite network improves protein function prediction. *PloS one* 8, 4 (2013), e60372.

[33] Jure Leskovec and Julian Mcauley. 2012. Learning to discover social circles in ego networks. *Advances in neural information processing systems* 25 (2012).

[34] Jia Li, Yongfeng Huang, Heng Chang, and Yu Rong. 2022. Semi-supervised hierarchical graph classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 5 (2022), 6265–6276.

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

[36] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. 2018. Combinatorial optimization with graph convolutional networks and guided tree search. *Advances in neural information processing systems* 31 (2018).

[37] Fanzhen Liu, Shan Xue, Jia Wu, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Jian Yang, and Philip S Yu. 2020. Deep learning for community detection: progress, challenges and opportunities. *arXiv preprint arXiv:2005.08225* (2020).

[38] Qing Liu, Yifan Zhu, Minjun Zhao, Xin Huang, Jianliang Xu, and Yunjun Gao. 2020. VAC: vertex-centric attributed community search. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 937–948.

[39] Xin Liu, Haojie Pan, Mutian He, Yangqiu Song, Xin Jiang, and Lifeng Shang. 2020. Neural subgraph isomorphism counting. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1959–1969.

[40] Yanli Liu, Chu-Min Li, Hua Jiang, and Kun He. 2020. A learning based branch and bound for maximum common subgraph related problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 2392–2399.

[41] Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, Jure Leskovec, et al. 2020. Neural subgraph matching. *arXiv preprint arXiv:2007.03092* (2020).

[42] Jiehuan Luo, Xin Cao, Xike Xie, Qiang Qu, Zhiqiang Xu, and Christian S Jensen. 2020. Efficient attribute-constrained co-located community search. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1201–1212.

[43] Erxue Min, Yu Rong, Yatao Bian, Tingyang Xu, Peilin Zhao, Junzhou Huang, and Sophia Ananiadou. 2022. Divide-and-conquer: Post-user interaction network for fake news detection on social media. In *Proceedings of the ACM web conference 2022*. 1148–1158.

[44] Erxue Min, Yu Rong, Tingyang Xu, Yatao Bian, Da Luo, Kangyi Lin, Junzhou Huang, Sophia Ananiadou, and Peilin Zhao. 2022. Neighbour interaction based click-through rate prediction via graph-masked transformer. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 353–362.

[45] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[46] Pedro Ribeiro, Pedro Paredes, Miguel EP Silva, David Aparicio, and Fernando Silva. 2021. A survey on subgraph counting: concepts, algorithms, and applications to network motifs and graphlets. *ACM Computing Surveys (CSUR)* 54, 2 (2021), 1–36.

[47] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Hkx1qkrKPr

[48] Dhiman Sarma, Wahidul Alam, Ishita Saha, Mohammad Nazmul Alam, Mohammad Jahangir Alam, and Sohrab Hossain. 2020. Bank fraud detection using community detection algorithm. In *2020 second international conference on inventive research in computing applications (ICIRCA)*. IEEE, 642–646.

[49] Guojiang Shen, Difeng Zhu, Jingjing Chen, and Xiangjie Kong. 2022. Motif discovery based traffic pattern mining in attributed road networks. *Knowledge-Based Systems* 250 (2022), 109035.

[50] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 565–573.

[51] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. 2017. Graph attention networks. *stat* 1050, 20 (2017), 10–48550.

[52] Botao Wang, Jia Li, Yang Liu, Jiashun Cheng, Yu Rong, Wenjia Wang, and Fugee Tsung. 2024. Deep Insights into Noisy Pseudo Labeling on Graph Data. *Advances in Neural Information Processing Systems* 36 (2024).

[53] Hanchen Wang, Rong Hu, Ying Zhang, Lu Qin, Wei Wang, and Wenjie Zhang. 2022. Neural subgraph counting with wasserstein estimator. In *Proceedings of the 2022 International Conference on Management of Data*. 160–175.

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

[55] Su Xing, Xue Shan, Liu Fanzhen, Wu Jia, Yang Jian, Zhou Chuan, Hu Wenbin, Paris Cecile, Nepal Surya, Jin Di, et al. 2022. A comprehensive survey on community detection with deep learning. *IEEE Trans. Neural Netw. Learn. Syst* (2022).

[56] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *7th International Conference on Learning Representations, ICLR 2019,*. OpenReview.net. https://openreview.net/forum?id=ryGs6iA5Km

[57] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*. PMLR, 40–48.

[58] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.

[59] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the twelfth ACM international conference on web search and data mining*. 582–590.

[60] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931* (2019).

[61] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. 2019. Prone: Fast and scalable network representation learning.. In *IJCAI*, Vol. 19. 4278–4284.

[62] Muhan Zhang and Yixin Chen. 2019. Inductive matrix completion based on graph neural networks. *arXiv preprint arXiv:1904.12058* (2019).

[63] Tianqi Zhang, Yun Xiong, Jiawei Zhang, Yao Zhang, Yizhu Jiao, and Yangyong Zhu. 2020. CommDGI: community detection oriented deep graph infomax. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 1843–1852.

[64] Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. 2020. Every document owns its structure: Inductive text classification via graph neural networks. *arXiv preprint arXiv:2004.13826* (2020).

[65] Kangfei Zhao, Jeffrey Xu Yu, Qiyan Li, Hao Zhang, and Yu Rong. 2023. Learned sketch for subgraph counting: a holistic approach. *The VLDB Journal* (2023), 1–26.

[66] Kangfei Zhao, Jeffrey Xu Yu, Hao Zhang, Qiyan Li, and Yu Rong. 2021. A learned sketch for subgraph counting. In *Proceedings of the 2021 International Conference on Management of Data*. 2142–2155.

[67] Kangfei Zhao, Zhiwei Zhang, Yu Rong, Jeffrey Xu Yu, and Junzhou Huang. 2021. Finding critical users in social communities via graph convolutions. *IEEE Transactions on Knowledge and Data Engineering* (2021).