

To Learn or Not to Learn? An In-depth Empirical Analysis of Community Search Approaches

YUHAN ZHOU, Zhejiang University, China

ZHENG WU, Hong Kong Baptist University, China

QING LIU*, Zhejiang University, China

XIN HUANG, Hong Kong Baptist University, China

JIANLIANG XU, Hong Kong Baptist University, China

YUNJUN GAO, Zhejiang University, China

Community search (CS) is a fundamental problem in graph analysis, which aims to identify a query-dependent cohesive subgraph that satisfies a specific community model and contains the given query vertices. While numerous CS approaches have been proposed, including non-learning-based and learning-based ones, existing research lacks a unified experimental evaluation that bridges these two paradigms. Consequently, the relative strengths and weaknesses of these approaches remain insufficiently understood. Moreover, practitioners lack an effective method to recommend appropriate CS approaches for diverse queries. To address these limitations, this paper conducts a comprehensive and in-depth empirical analysis of representative CS approaches across various real-world datasets. We systematically evaluate their overall performance, investigate the impact of intrinsic properties of ground-truth communities, and analyze the approaches' generalizability within a unified experimental framework. The results show that existing CS approaches exhibit limited generalizability across varying queries, implying that no single approach performs optimally in all cases and that the best-performing approach depends on the specific query. Building upon these insights, we propose RecCS, a query-aware recommendation model that selects the top-performing CS approaches. Extensive experiments demonstrate that RecCS improves CS performance efficiently and effectively. Finally, we summarize the limitations of current CS approaches and advocate generalizability as a pivotal direction for future research.

CCS Concepts: • **Mathematics of computing** → **Graph algorithms**.

Additional Key Words and Phrases: Community search; Experimental evaluation

ACM Reference Format:

Yuhan Zhou, Zheng Wu, Qing Liu, Xin Huang, Jianliang Xu, and Yunjun Gao. 2026. To Learn or Not to Learn? An In-depth Empirical Analysis of Community Search Approaches. *Proc. ACM Manag. Data* 4, 4 (SIGMOD), Article 290 (September 2026), 27 pages. <https://doi.org/10.1145/3837128>

1 Introduction

As a fundamental problem in graph analysis, community search (CS) has received significant attention in recent years. Given a graph G , a query vertex set Q , and a community model M , CS aims to identify a query-dependent cohesive subgraph $G' \subseteq G$ that satisfies M and contains Q . It supports

*Corresponding author

Authors' Contact Information: Yuhan Zhou, Zhejiang University, Hangzhou, China, zhou_yh@zju.edu.cn; Zheng Wu, Hong Kong Baptist University, Hong Kong, China, cszhengwu@comp.hkbu.edu.hk; Qing Liu, Zhejiang University, Hangzhou, China, qingliucs@zju.edu.cn; Xin Huang, Hong Kong Baptist University, Hong Kong, China, xinhuang@comp.hkbu.edu.hk; Jianliang Xu, Hong Kong Baptist University, Hong Kong, China, xujl@comp.hkbu.edu.hk; Yunjun Gao, Zhejiang University, Hangzhou, China, gaoyj@zju.edu.cn.



This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2026 Copyright held by the owner/author(s).

ACM 2836-6573/2026/9-ART290

<https://doi.org/10.1145/3837128>

a wide range of applications such as user recommendation [17, 83], biological analysis [63, 64], and collaboration discovery [56, 80]. For example, in product networks, CS can be employed to identify frequently co-purchased products for recommendation. In biological networks, CS helps uncover relevant proteins based on their frequent interactions. Due to its practical importance, CS has been extensively studied [25, 34]. Existing CS approaches can be broadly categorized into two paradigms: *non-learning-based approaches* [3–5, 7, 10, 11, 18–21, 36–39, 41, 43, 52, 57, 71, 79, 84] and *learning-based approaches* [13, 15, 16, 32, 33, 42, 47, 70, 74, 76].

Non-learning-based CS approaches rely on predefined *community patterns* to characterize the target community. Based on the nature of these predefined patterns, the non-learning-based CS approaches can be further categorized into *objective-function-based* [5, 18, 21, 36, 43, 52, 57, 79] and *cohesive-subgraph-based* [3, 4, 7, 10, 11, 19, 20, 37–39, 41, 71, 84]. The former category defines a *goodness metric* for evaluating communities and typically employs heuristic search algorithms to identify the community with the optimal metric. In contrast, the latter aims to extract subgraphs that strictly satisfy the structural constraints imposed by specific cohesive subgraph models (e.g., *k*-core or *k*-truss). Although these approaches do not require ground-truth communities, they often lack the flexibility to accommodate diverse community structures in real-world graphs.

To tackle this issue, recent studies have explored learning-based CS approaches [13, 15, 16, 32, 33, 42, 47, 70, 74, 76], which leverage graph learning models to capture both graph topology and community patterns in a data-driven manner. Based on their learning categories, these approaches can be further divided into *ground-truth-based learning approaches* [13, 15, 16, 32, 42, 47, 70] and *structure-based learning approaches* [33, 74, 76]. Ground-truth-based learning approaches adopt supervised or semi-supervised models to learn the implicit structures underlying labeled communities. In contrast, structure-based learning approaches exploit generated pseudo-labels [33, 76] or structural losses [74] for self-supervised or unsupervised training. Despite their improved flexibility, most learning-based approaches rely on subgraph sampling [16, 32, 74, 76] or partitioning [47] to handle large-scale graphs, which makes it challenging to achieve both efficiency and effectiveness.

Recently, several surveys have reviewed existing CS approaches from different perspectives. For example, [25] focuses on non-learning-based CS approaches, while [34] provides an overview and experimental comparison of learning-based CS approaches. In addition, [72] summarizes CS approaches on complex graphs, and [89] reviews CS from a social-psychological perspective. Despite these efforts, existing works suffer from several limitations.

L1: Absence of cross-paradigm evaluation. Existing surveys lack a comprehensive comparison between different paradigms. While Gou et al. [34] provide an experimental comparison of learning-based approaches, a critical gap remains: there is still no unified framework that systematically assesses both non-learning-based and learning-based approaches under identical settings in order to fully reveal the respective advantages and limitations of these two distinct paradigms.

L2: Lack of in-depth performance explanation. Current studies primarily focus on comparing the performance of different CS approaches, with limited investigation into the underlying factors of experimental results. Although [34] evaluates the relationship between retrieved communities and components of learning-based CS approaches, they overlook the correlation between approach performance and the intrinsic properties of ground-truth communities. This oversight obstructs a clear understanding of the mechanisms driving algorithmic performance, which is essential for diagnosing bottlenecks of existing approaches and inspiring principled algorithm designs.

L3: Inadequate evaluation of CS approaches' generalizability. Existing studies evaluate CS approaches under limited experimental settings, failing to comprehensively capture their performance across diverse real-world scenarios. Specifically, prior evaluations often overlook the impact of varying query characteristics and graph structural properties [25, 34]. Such oversight may introduce evaluation bias, potentially overestimating the approaches' effectiveness in ideal

conditions while masking their vulnerabilities in complex or noisy environments. As a result, it remains unclear how robustly existing CS approaches generalize across different configurations.

L4: Coarse-grained recommendation. Existing studies have proposed some guidelines for selecting CS approaches in a coarse-grained manner based on empirical analyses. However, such guidelines are too general to work well in practice. For example, Gou et al. [34] recommend COCLEP [47] and TransZero [74] for scenarios with limited labeled data. However, even on the same graph with limited labels, COCLEP and TransZero do not always achieve optimal performance across different queries. Therefore, designing a fine-grained CS approach recommendation model tailored to query characteristics is urgently needed.

To address these limitations, we conduct a comprehensive and in-depth empirical analysis and propose an effective query-aware CS approach recommendation model.

For **L1**, we present a systematic review (Section 3) and conduct a unified experimental evaluation of non-learning-based and learning-based CS approaches (Section 5.2). Our results reveal that learning-based approaches do not consistently outperform non-learning-based approaches, even when labels are available. Furthermore, the optimal approach varies across diverse queries. This performance variance confirms that no single approach is universally superior for different queries, highlighting the necessity of query-specific CS approach selection to improve CS performance.

For **L2**, we investigate the underlying factors of the performance of CS approaches by visualizing and analyzing the properties of ground-truth communities (Section 5.3). We discover that ground-truth communities exhibit structural heterogeneity in terms of size and cohesiveness. However, most existing approaches are sensitive to these properties: they prefer specific structural patterns but struggle to generalize across communities with varying sizes and cohesiveness. This analysis identifies the limitations of existing CS approaches in handling diverse community structures.

For **L3**, we evaluate the generalizability of CS approaches across different graph and query configurations (Section 5.4). For graph generalizability, we reveal that learning-based approaches, despite offering stable online efficiency, incur high offline training costs and do not consistently exhibit superior robustness against structural noises compared to non-learning-based approaches. Regarding query generalizability, we find that most approaches are inadequate for CS with multiple query vertices, as they either do not support multiple query vertices or fail to effectively exploit the structural information provided by these query vertices. Moreover, their performance is sensitive to query vertex locations, resulting in suboptimal community identification when query vertices are located at community boundaries.

For **L4**, we introduce RecCS, a fine-grained CS approach recommendation model (Section 4). Specifically, it employs a query-aware encoder to capture the local structure of query vertices and leverages a configuration recommender to recommend CS approaches efficiently. Extensive experiments demonstrate that RecCS can improve CS performance with minor overhead (Section 5.5).

Based on our extensive experimental results, we identify the lack of generalizability as a fundamental bottleneck of existing CS approaches. This underscores the necessity of designing future unified CS frameworks that are capable of achieving generalizability across queries, graphs, and diverse application scenarios (Section 5.6).

In summary, the main contributions of this paper are as follows:

- **Unified Comprehensive Evaluation.** To the best of our knowledge, we are the first to systematically evaluate both non-learning-based and learning-based CS approaches within a unified experimental framework.
- **Insightful Empirical Analysis.** We conduct a comprehensive and in-depth empirical analysis of CS approaches across various settings, uncovering key insights into their performance, underlying influencing factors, and practical applicability.

- **Fine-grained Recommendation.** Building upon empirical insights, we propose RecCS, a query-aware recommendation model that selects the top-performing CS approaches to effectively improve CS performance.
- **Future Directions.** We summarize the limitations of current CS approaches and advocate generalizability as a pivotal direction for future research.

2 Preliminaries

Let $G = (V_G, E_G)$ denote a simple undirected graph where $n = |V_G|$ and $m = |E_G|$. The degree of a vertex v is defined as $\deg_G(v) = |N_G(v)|$, where $N_G(v)$ is the set of neighbors of v in G . For simplicity, we denote the average vertex degree of G by $\deg_G$. The feature matrix of G is represented by $\mathbf{X} \in \mathbb{R}^{n \times d_f}$, where $\mathbf{x}_v \in \mathbb{R}^{d_f}$ is the feature vector of the vertex v and d_f is the dimension of features. Next, we formally define the community search problem.

DEFINITION 1. (*Community Search* [25, 34]). *Given a simple undirected graph $G = (V_G, E_G)$, a query vertex set $Q \subseteq V_G$, and a community model M , the community search aims to find a query-dependent cohesive subgraph $G' \subseteq G$ that satisfies M and contains Q .*

Scope. To provide a focused yet comprehensive empirical analysis, this paper concentrates on the classical CS problem on simple graphs. Related studies that consider other graph types (e.g., directed [2, 26, 51, 54], bipartite [75, 87, 88, 90], temporal [35, 49, 53, 86], heterogeneous graphs [27, 50, 81, 85]) or investigate variant CS problems (e.g., inductive [23, 24], attributed [40, 55, 77, 78], influential CS [8, 12, 48, 91]) fall outside the scope of this work and are left for future exploration.

3 Overview of Community Search

In this section, we first present a structured taxonomy of representative CS approaches, followed by a systematic review of them.

3.1 Structured Taxonomy

Overall, we categorize the existing CS approaches as *non-learning-based* CS [3–5, 7, 10, 11, 18–21, 36–39, 41, 43, 52, 57, 71, 79, 84] and *learning-based* CS [13, 15, 16, 32, 33, 42, 47, 70, 74, 76]. Table 1 summarizes the characteristics of 16 representative CS approaches.

Specifically, the primary distinction between these two categories lies in whether learning-based models are employed. Non-learning-based approaches rely on predefined cohesive metrics and can be further divided into *objective-function-based* CS (e.g., density, modularity, and conductance) and *cohesive-subgraph-based* CS (e.g., core, truss, clique, and ECC), depending on the metrics they adopt. In contrast, learning-based approaches leverage learning-based models (e.g., GNNs or Transformers) to capture community patterns in a data-driven manner. Based on the availability of ground-truth labels, they can be further categorized into *ground-truth-based learning* CS (e.g., supervised and semi-supervised CS) and *structure-based learning* CS (e.g., self-supervised and unsupervised CS). Despite their differences in paradigms, query support, offline processing, label usage, and backbone models (as outlined in Table 1), they all aim to capture the structural characteristics of communities. We next provide a systematic review of them.

3.2 Non-learning-based Community Search

Non-learning-based CS defines community patterns using hand-crafted principles, which generally fall into two categories: (i) objective functions for scoring community quality, and (ii) cohesive subgraph models that impose explicit structural requirements. Accordingly, we group non-learning-based CS into *objective-function-based* [5, 18, 21, 36, 43, 52, 57, 79] and *cohesive-subgraph-based* approaches [3, 4, 7, 10, 11, 19, 20, 37–39, 41, 71, 84].

Table 1. Comparison of representative CS approaches

Category		Approach	Query Vertices	Index/Model	Label Required	Backbone
Non-learning	Objective-function-based	QDC [79]	Multiple	✗	✗	Density
		LM [18]	Single	✗	✗	Modularity
		SGM [57]	Multiple	✗	✗	Modularity
		DMCS [43]	Multiple	✗	✗	Modularity
		APPR [5]	Single	✗	✗	Conductance
	Cohesive-subgraph-based	CSP [7]	Multiple	✓	✗	Core
		EquiTruss (ET) [3]	Single	✓	✗	Truss
		KCPC [84]	Multiple	✓	✗	Clique
		SMCC [10]	Multiple	✓	✗	ECC
Learning	Ground-truth-based	ICS-GNN (ICS) [32]	Single	✓	✓	GNN
		QD-GNN (QD) [42]	Multiple	✓	✓	GNN
		CommunityAF (CAF) [15]	Single	✓	✓	AF
		CommunityDF (CDF) [16]	Single	✓	✓	DDPM
		COCLEP (CLP) [47]	Single	✓	✓	GNN
	Structure-based	CSFormer (CSF) [76]	Single	✓	✗	Transformer
		TransZero (TZ) [74]	Multiple	✓	✗	Transformer

3.2.1 Objective-function-based CS.

Objective-function-based approaches formulate CS as an optimization problem of the given community metric. While various metrics have been proposed [30, 59, 65, 69, 82], density-based, modularity-based, and conductance-based approaches remain the three most representative families (A1-A3).

A1: Density-based. Density-based objectives emphasize internal cohesiveness and typically favor subgraphs with high average connectivity. There are many varieties of density, such as edge density [66], internal density [65], clique density [58, 67], and pattern density [28]. To unify these varied definitions, Dai et al. [21] introduce a generalized framework using weight configurations to formalize a broad class of objective functions for density-based local CS. However, density maximization in CS often suffers from the free-rider effect, where vertices irrelevant to the query are included to increase the density objective. QDC [79] mitigates this issue by introducing *query-biased vertex weights*, where each vertex is weighted by its proximity to the query via the random walk. The density objective is adjusted to a weighted form, defined as the ratio of the number of internal edges to the sum of the query-biased weights of the vertices within the subgraph. It downweights vertices far from Q and pulls the optimal solution toward the query neighborhood.

A2: Modularity-based. The classic modularity [60] is originally introduced to measure the quality of global partitions by comparing the observed internal edges against those in a randomized null model. While it has been extensively studied in the literature [22, 29, 73], directly applying classic modularity to the CS problem suffers from the resolution limit [31] and the free-rider effect [79], which prevents it from identifying small but cohesive communities and leads to the inclusion of irrelevant vertices. To address these limitations, several representative modularity-based approaches tailored for CS have been proposed, including local-modularity-based (LM) [18], subgraph-modularity-based (SGM) [57], and density-modularity-based (DMCS) [43] approaches. Specifically, LM [18] optimizes the sharpness of community boundaries by measuring the fraction of edges incident to boundary vertices that remain within the community. In contrast, SGM [57] evaluates community quality by comparing the internal and external edges of the community. To jointly capture relative and absolute cohesiveness, DMCS [43] integrates classic modularity with density by normalizing the modularity score with the community size.

A3: Conductance-based. Conductance measures the relative separation of a community by normalizing its external cut with respect to its internal volume, thereby capturing the trade-off between internal cohesion and external isolation [69]. Due to its effectiveness, existing works have adopted the conductance in the CS problem [5, 36, 52], where the objective is to identify a community that minimizes conductance. To optimize conductance efficiently, many approaches follow a diffuse-and-sweep paradigm. Specifically, they first compute a localized diffusion score around the seed vertex using techniques such as Personalized PageRank (PPR) [6] or heat-kernel diffusion [44]. Vertices are then ranked according to degree-normalized diffusion scores, and sweep cuts are applied to select the prefix that yields the minimum conductance. A representative method in this line of work is APPR [5], which employs a push-based algorithm to compute an ϵ -approximate PPR vector. By sorting vertices based on their degree-normalized scores and performing sweep cuts, APPR identifies a locally optimal community with respect to conductance. More recently, several studies have formulated conductance-based CS by explicitly enforcing query vertex inclusion and connectivity, considering both bounded-size [36] and unbounded-size [52] community settings.

3.2.2 Cohesive-subgraph-based CS.

Cohesive-subgraph-based approaches define a community with explicit structural constraints and search for subgraphs that satisfy them. Based on the underlying cohesiveness model, existing methods can be grouped into four types (**B1-B4**): core-based, truss-based, clique-based, and edge-connectivity-based (ECC) approaches.

B1: Core-based. A k -core is a maximal subgraph $G' = (V_{G'}, E_{G'})$ in which every vertex satisfies $\deg_{G'}(v) \geq k$. Sozio et al. [71] formulate the core-based CS problem (CSP), which aims to find a connected subgraph containing Q with the maximum k . To tackle this problem, they propose a greedy global algorithm that iteratively removes the vertices with the minimum degree. To avoid the cost of global peeling, Cui et al. [20] develop a local expansion strategy that begins from a query vertex and incrementally grows the subgraph until a valid k -core is obtained. Leveraging the nested property that a k -core is always contained within a $(k - 1)$ -core, Barbieri et al. [7] solve the CSP by designing a hierarchical index structure. This index organizes all connected components of k -cores into a tree, where each vertex represents a component at a specific core level. By checking the upper level of query vertices, it identifies the target community without traversing the entire graph efficiently.

B2: Truss-based. To strengthen cohesiveness beyond degree constraints, the k -truss model requires every edge $e \in E_{G'}$ to participate in at least $k - 2$ triangles within G' . By leveraging triangles as fundamental building blocks, truss-based communities are able to capture stronger and more stable relationships. Early methods such as TCP-index [39] utilize maximum spanning trees to preserve truss numbers and triangle adjacency relationships. To further improve efficiency, Akbas et al. [3] introduce EquiTruss, a more efficient index based on k -truss equivalence. Specifically, EquiTruss builds an index for truss-based CS by partitioning edges into disjoint equivalence classes based on triangle connectivity. The graph is then compressed, where each super-node represents an equivalence class and super-edges preserve triangle connectivity across different classes. By eliminating the redundant maintenance of edges across multiple maximum spanning forests in TCP-index, EquiTruss achieves improved time and space efficiency. Additionally, variants like CTC [41] further extend the truss model by incorporating diameter constraints to mitigate the free-rider effect and ensure a compact community structure.

B3: Clique-based. The k -clique is a complete subgraph of size k . While various k -clique-based community models [1, 61] and their variants [9, 19, 68] have been proposed, the k -clique percolation community (KCPC) model [61] remains a representative approach. KCPC defines a community as the maximal union of adjacent k -cliques, where two k -cliques are adjacent if they share $k - 1$

vertices. Based on the observation that the KCPC is a union of maximal cliques, Yuan et al. [84] develop an index-based approach to efficiently identify both the KCPC and the densest clique percolation community (DCPC) containing the query vertex set Q . By preserving the maximal cliques and their connectivity into a hierarchical index, it enables the algorithm to explore only the subtrees relevant to the query vertices Q , thereby reducing the search space efficiently. Given the strict constraints of the k -clique model, Cui et al. [19] further propose a parameterized relaxation of both the k -clique model and k -clique adjacency, which alleviates the excessive number of k -clique communities and supports more flexible and practical CS.

B4: ECC-based. A subgraph is defined as k -edge-connected if it remains connected after the removal of any $k - 1$ edges. The objective of ECC-based CS is to identify a subgraph containing the query vertices Q with the maximum edge connectivity k [4, 11, 37, 38]. To facilitate efficient online querying, SMCC [10] constructs a compact tree index. It first builds a weighted graph where edge weights encode pairwise connectivities, and then derives a Maximum Spanning Tree (MST) from it. Leveraging the MST structure, the method determines the connectivity value of Q and retrieves the target community by traversing the subtree formed by edges with weights no less than the connectivity value. This index-based approach achieves a query time linear to the output size and supports efficient maintenance under dynamic updates. However, since maximizing connectivity often yields large communities, several studies focus on identifying the ECC-based minimum community [37, 38].

3.3 Learning-based Community Search

Unlike non-learning-based approaches, which rely on predefined paradigms to characterize communities, learning-based approaches operate in a data-driven manner. Given a graph and a query vertex set, learning-based CS approaches leverage trained models to identify the target community relevant to the query. Based on learning objectives, they can be divided into two categories: *ground-truth-based learning CS* [13, 15, 16, 32, 42, 47, 70] and *structure-based learning CS* [33, 74, 76].

3.3.1 Ground-truth-based learning CS.

Ground-truth-based learning CS aims to capture community structures by leveraging known ground-truth communities. Below, we review two primary categories (C1-C2): supervised learning CS and semi-supervised learning CS.

C1: Supervised learning CS. Supervised learning CS optimizes model parameters by explicitly utilizing ground-truth labels as supervision signals. The objective is to maximize the alignment between the predicted community and the ground-truth community. Existing models can be broadly categorized into two paradigms: discriminative and generative models.

Discriminative models predict whether each vertex belongs to the community based on the vertex representation. For example, ICS-GNN [32] applies a Graph Neural Network (GNN) to estimate the membership probability of each vertex within the sampled candidate subgraph and employs a vertex-swapping strategy to extract the k -sized subgraph with the highest aggregated GNN scores. ICS-GNN⁺ [13] extends ICS-GNN by incorporating cluster-level information and designing an efficient adaptive mechanism to update candidate subgraphs during the iterative search. However, both ICS-GNN and ICS-GNN⁺ require training for each query in the online phase, leading to high computational overhead. To mitigate this issue, QD-GNN [42] is proposed by employing a query encoder to capture structural information around the given query vertices and a graph encoder to learn query-independent vertex representations, respectively. Recently, Sima et al. [70] propose a learning-based framework for overlapping CS and introduce a lightweight Simplified Multi-hop Attention Network (SMN) to improve efficiency.

To capture the global distribution of communities, generative models have been introduced for CS, which aim to model the underlying distribution of entire communities. CommunityAF [15] integrates an incremental GNN to learn vertex representations with community-aware structural features, and employs an autoregressive flow (AF) model to generate community members. To improve generation efficiency and robustness, CommunityDF [16] constructs a query-relevant subgraph via a graph sampler and leverages a denoising diffusion probabilistic model (DDPM) guided by both generative and contrastive losses to learn high-quality vertex representations from limited examples.

C2: Semi-supervised learning CS. Semi-supervised CS alleviates the reliance on labeled data by jointly leveraging the labeled communities and unlabeled graph structures. By complementing supervised learning with unsupervised regularization, it improves generalization and robustness under label scarcity. COCLEP [47] introduces a semi-supervised framework based on contrastive learning. Specifically, it utilizes GNN encoders to capture query-dependent structural information and designs a label-aware contrastive learner to align representations across low-order, high-order, and cross-order views. Furthermore, COCLEP incorporates graph partitioning strategies to optimize memory efficiency during model training.

3.3.2 Structure-based learning CS.

Structure-based learning CS aims to optimize models by leveraging the topological structure of the graph, thereby eliminating the reliance on ground-truth communities. In the following, we categorize these approaches into two paradigms (**D1-D2**): self-supervised learning CS and unsupervised learning CS.

D1: Self-supervised learning CS. To compensate for the lack of ground-truth communities, this category utilizes non-learning-based approaches to generate pseudo-labels. These labels are then employed as self-supervision signals to guide the model training. For example, CS-GE [33] injects cohesiveness information of k -core into vertex representations. By leveraging the proximity graph built from core-based vertex representations, CS-GE enables community identification efficiently. Similarly, CSFormer [76] leverages the core number as the pseudo-labels for model training, where the core number of a vertex is defined as the maximum k such that the vertex participates in a k -core. It designs a Transformer-based architecture that encodes structural cohesiveness via n -th order h -index features to predict core numbers. The predicted values are then utilized to infer the target community.

D2: Unsupervised learning CS. Different from self-supervised learning approaches that adopt pseudo-labels, unsupervised learning CS optimizes models directly via structural losses. TransZero [74] follows this paradigm by leveraging a pre-trained Graph Transformer [14]. In the offline phase, it adopts a personalization loss and a link loss to capture vertex-specific characteristics and graph structure, respectively. In the online phase, TransZero computes community scores based on representation similarity between the query vertices and other vertices. It then identifies communities by maximizing the expected score gain, thereby avoiding dependence on label-based score thresholds.

4 Methodology of RecCS

Despite the advances of existing works, our experimental results in Section 5 demonstrate that no single approach achieves optimal performance across all queries due to their inherent lack of generalizability. To address this challenge, we propose RecCS from a novel recommendation perspective to provide a more effective and practical alternative. As a lightweight recommendation model, RecCS is designed to dynamically select the top-performing CS approach for each query.

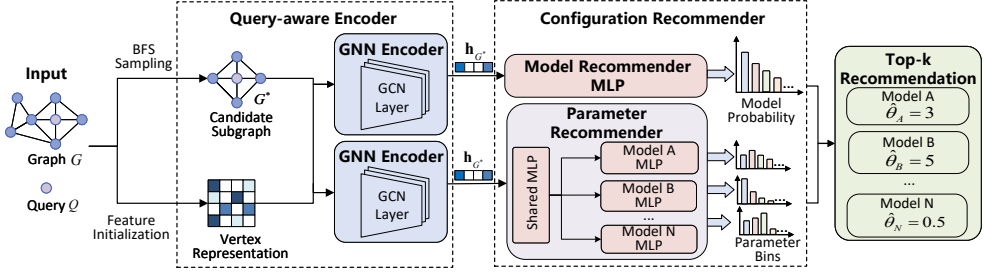


Fig. 1. Framework of RecCS.

4.1 Overview of RecCS

The RecCS model comprises two components: a *query-aware encoder* and a *configuration recommender*, as illustrated in Figure 1. The query-aware encoder captures the local structural characteristics of query vertices by extracting candidate subgraphs and encoding structural features with the GNN to generate an expressive representation for recommendation. Based on the generated representation, the configuration recommender predicts the scores of CS approaches using a dual-branch architecture, where one branch estimates model scores and the other predicts parameter scores. Finally, RecCS outputs the top- k model-parameter configurations for downstream CS tasks. Notably, this distinguishes RecCS from learning-based CS approaches, as it outputs recommended approaches rather than directly predicting the target community. In this paper, we adopt the F1-score as the default recommendation metric, which can be easily extended to other metrics by adjusting the recommendation labels based on the adopted metrics.

4.2 Query-aware Encoder

This module extracts local structural features of query vertices, motivated by the intuition that vertices with similar local structures tend to favor similar models and parameter configurations.

4.2.1 Subgraph Sampling and Feature Initialization.

To capture the local structural features efficiently, we employ a Breadth-First Search (BFS) strategy to sample a candidate subgraph $G^* = (V_{G^*}, E_{G^*})$ centered at the query vertices Q , which has been widely used in CS [13, 16, 32]. Next, we initialize the vertex representations by explicitly incorporating query-relevant features. Specifically, for each vertex v , its initial representation $\mathbf{x}_v$ is constructed by concatenating the raw features, the normalized PPR score [5], and a binary query indicator q_v , where $q_v = 1$ if $v \in Q$ and 0 otherwise. Furthermore, to capture the structural locality w.r.t. Q , we inject structural position information into the vertex features. Formally, for each vertex $v \in V_{G^*}$, the input layer representation $\mathbf{h}_v^{(0)}$ is formulated as:

$$\mathbf{h}_v^{(0)} = \mathbf{W}^{(0)} \mathbf{x}_v + \mathbf{E}_{\text{dist}}[\text{dist}_{G^*}(v, Q)], \quad (1)$$

where $\mathbf{W}^{(0)}$ is a learnable projection matrix for $\mathbf{x}_v$, $\text{dist}_{G^*}(v, Q)$ is the shortest path distance between v and Q in G^* , and $\mathbf{E}_{\text{dist}}$ is a learnable embedding table that encodes $\text{dist}_{G^*}(v, Q)$.

4.2.2 Structure Encoding.

Based on the sampled subgraph and initialized features, we employ the GNN to capture multi-hop structural information. The vertex representations are updated iteratively via message passing:

$$\mathbf{h}_v^{(k)} = \phi(\mathbf{h}_v^{(k-1)}, \text{AGG}(\{\mathbf{h}_u^{(k-1)} : u \in N_{G^*}(v)\})), \quad (2)$$

where $\mathbf{h}_v^{(k)}$ is the representation of v in layer k , ϕ is the message propagation function to update the vertex representation, and AGG is the aggregate function. To obtain a graph-level representation that is both query-focused and context-aware for the subsequent recommendation, we aggregate the representations of the query vertices and all vertices in the candidate subgraph, respectively. Finally, the output of the query-aware encoder is the concatenation of these two views:

$$\mathbf{h}_{G^*} = \text{Concat}\left(\frac{1}{|Q|} \sum_{v \in Q} \mathbf{h}_v^{(l)}, \frac{1}{|V_{G^*}|} \sum_{v \in V_{G^*}} \mathbf{h}_v^{(l)}\right), \quad (3)$$

where Concat denotes the concatenation operation, l is the number of GNN layers, and $\mathbf{h}_v^{(l)}$ is the final layer output of v .

4.3 Configuration Recommender

To recommend top-performing CS approaches, we design a configuration recommender that leverages the *Kullback-Leibler (KL) divergence* [45] to guide the learning process by minimizing the discrepancy between the predicted and ground-truth performance distributions. It adopts a dual-branch architecture consisting of a *model recommender* and a *parameter recommender*, which are trained separately using their respective loss functions.

4.3.1 Model Recommender.

The model recommender employs a Multi-Layer Perceptron (MLP) to learn the mapping between local structural features $\mathbf{h}_{G^*}$ and the performance score of candidate CS approaches. To capture both the ranking order and the performance differences, we optimize the model recommender via two complementary objectives: (i) *model ranking*, which emphasizes correctly ordering the models; and (ii) *performance distribution alignment*, which captures the relative performance gaps among candidate models.

Model Ranking. We adopt a ranking loss $\mathcal{L}_{rank}$ to facilitate accurate ordering of the top-ranked candidate models. Specifically, given a model with ground-truth performance rank r and its predicted score $\hat{s}^{(r)}$, we enforce the predicted scores of all models ranked below it to be lower than $\hat{s}^{(r)}$ by at least a margin γ . The loss is weighted by the rank to emphasize the importance of the top- k candidate models:

$$\mathcal{L}_{rank} = \sum_{r=1}^k \frac{1}{r} \left(\frac{1}{\max\{1, |\mathcal{V}_r|\}} \sum_{i \in \mathcal{V}_r} (\hat{s}_i - \hat{s}^{(r)} + \gamma) \right). \quad (4)$$

We use $\mathcal{S}_{>r}$ to denote the set of models with a true rank index greater than r , $\hat{s}_i$ is the predicted score of the model i , and $\mathcal{V}_r = \{i \in \mathcal{S}_{>r} \mid \hat{s}_i - \hat{s}^{(r)} + \gamma > 0\}$ is the set of models in $\mathcal{S}_{>r}$ that fail to satisfy the margin constraint. The weighting factor $1/r$ ensures that errors in predicting the top-ranked candidate models are penalized more heavily than those with larger rank indexes.

Performance distribution alignment. While the ranking loss optimizes for accurate ordering, it fails to capture the performance differences of different models. To address this problem, we align the predicted score distribution with the ground-truth performance distribution using the KL divergence. Let $\mathbf{v}_m, \hat{\mathbf{v}}_m \in \mathbb{R}^{|\mathcal{M}|}$ denote the vectors of ground-truth and predicted performance scores, respectively, where $\mathcal{M}$ is the set of candidate models. We normalize these scores into probability distributions $\mathbf{p} = \sigma(\mathbf{v}_m/\tau)$ and $\hat{\mathbf{p}} = \sigma(\hat{\mathbf{v}}_m/\tau)$ via the Softmax function $\sigma(\cdot)$ with a temperature parameter τ . The KL divergence loss of the model recommender is formulated as:

$$\mathcal{L}_{KL}^{model} = \sum_{i=1}^{|\mathcal{M}|} p_i \log \left(\frac{p_i}{\hat{p}_i} \right), \quad (5)$$

where p_i and $\hat{p}_i$ are the i -th entries of $\mathbf{p}$ and $\hat{\mathbf{p}}$, respectively.

Finally, the model recommender is optimized jointly by integrating the ranking loss and the KL-divergence loss with a hyperparameter λ_{KL}^{model} :

$$\mathcal{L}_{model} = \mathcal{L}_{rank} + \lambda_{KL}^{model} \mathcal{L}_{KL}^{model}. \quad (6)$$

4.3.2 Parameter Recommender.

Since the optimal parameter relies on both the input query and the characteristics of models, the parameter recommender adopts a shared MLP encoder followed by model-specific MLP heads. The former captures shared features of the query, while the latter projects them into the parameter space of each specific model. We discretize the parameter space into bins and predict a score for each bin. Based on these predicted scores, the parameter recommender is trained from two perspectives: (i) *optimal parameter estimation*, which focuses on minimizing the deviation from the ground-truth optimal parameter; and (ii) *performance trend modeling*, which characterizes how model performance varies with the parameters.

Optimal Parameter Estimation. We formulate parameter prediction as a regression task. Let θ and $\hat{\theta}$ denote the ground-truth and predicted optimal parameter, respectively. We employ the Mean Squared Error (MSE) to quantify the estimation error:

$$\mathcal{L}_{reg} = |\theta - \hat{\theta}|^2. \quad (7)$$

Minimizing $\mathcal{L}_{reg}$ directly guides the parameter recommender to align its predictions with the optimal parameters. To mitigate scale disparities among different parameters, we normalize the loss based on the search range of each parameter. The final $\mathcal{L}_{reg}$ is computed by averaging these normalized losses across parameters and models.

Performance trend modeling. Solely regressing the optimal parameter neglects the performance trend across the parameter space, which offers valuable information for effective recommendations. Intuitively, this trend reflects the trade-off in community constraints guided by parameter settings: relaxing constraints initially incorporates more relevant vertices and improves performance, whereas excessive relaxation inevitably introduces irrelevant vertices and degrades quality. To capture this intrinsic trend, we employ the KL divergence loss to align the predicted parameter performance distribution with the ground truth. Specifically, we represent the performance trend as a vector defined over b discretized parameter bins. Let $\mathbf{v}_p, \hat{\mathbf{v}}_p \in \mathbb{R}^b$ denote the ground-truth and predicted parameter performance scores across these bins, respectively. We normalize these scores into probability distributions and compute the KL divergence loss according to Eq. (5). The resulting loss, denoted as $\mathcal{L}_{KL}^{param}$, is obtained by averaging over all models and parameters.

Finally, we integrate the regression loss and the KL divergence loss with a hyperparameter λ_{KL}^{param} to train the parameter recommender:

$$\mathcal{L}_{param} = \mathcal{L}_{reg} + \lambda_{KL}^{param} \mathcal{L}_{KL}^{param}. \quad (8)$$

5 Experiments

In this section, we conduct comprehensive experiments to provide an in-depth empirical analysis of representative CS approaches and our proposed RecCS, which aims to answer the following questions:

Q1: How do learning-based CS approaches compare with non-learning-based CS approaches? (Section 5.2)

Q2: How does the performance of different CS approaches correlate with ground-truth community characteristics? (Section 5.3)

Q3: How well do the CS approaches generalize under varying configurations? (Section 5.4)

Q4: How does RecCS perform across diverse evaluation settings? (Section 5.5)

Table 2. Dataset statistics ¹

Category	Graphs	Abbreviations	$ V_G $	$ E_G $	$\deg_G$	# C	size(C)
Classification Labels	Cora	CR	2,708	5,278	3.898	47	50.532
	Citeseer	CT	3,327	4,552	2.736	160	13.363
	Photo	PT	7,650	119,081	31.132	16	461.875
	CoCS	CO	18,333	81,894	8.934	55	323.091
	Physics	PS	34,493	247,962	14.378	68	502.074
Community Labels	Email-Eu	EM	986	16,064	32.584	38	23.132
	DBLP	DB	317,080	1,049,866	6.622	6,061	14.618
	Amazon	AZ	334,863	925,872	5.530	1,140	14.620
	Youtube	YT	1,134,890	2,987,624	5.265	2,294	12.607
	Livejournal	LJ	3,997,962	34,681,189	17.349	3,609	23.019
Unlabeled Graphs	Gnutella	GT	6,301	20,777	6.59	-	-
	Vote	VT	7,115	100,762	28.32	-	-
	CondMat	CM	23,133	93,439	8.08	-	-
	Epinions	EN	75,879	405,740	10.69	-	-
	NotreDame	ND	325,729	1,090,108	6.69	-	-

¹ # C: number of ground-truth communities; size(C): average size of ground-truth communities.

5.1 Experimental Setup

Datasets. To ensure a comprehensive evaluation, we select 15 real-world graphs from PyTorch [62] and SNAP [46]. The datasets are chosen from three aspects. First, they cover graphs with node classification labels, community labels, and without labels, bridging the evaluation gap between learning-based and non-learning-based paradigms. Second, they span diverse domains, including citation, co-purchasing, collaboration, email, social, file-sharing, voting, and web networks, which exhibit different interaction patterns and community formation mechanisms. Third, as summarized in Table 2, these graphs exhibit diverse structural characteristics in terms of graph size, average degree, and average ground-truth community size. For datasets with node classification labels, vertices sharing the same label may be disconnected. Therefore, we decompose them into connected components and filter out communities with fewer than three vertices.

Competitors. We evaluate our proposed RecCS model alongside 16 representative CS approaches, as detailed in Table 1. For non-learning-based approaches, we include approaches based on diverse cohesiveness metrics, including density, modularity, conductance, core, truss, clique, and ECC. Notably, the cohesive-subgraph-based approaches among them support index construction to accelerate query processing. For learning-based approaches, we select approaches covering different learning paradigms, including supervised, semi-supervised, self-supervised, and unsupervised learning.

Evaluation metrics. To comprehensively assess the performance of different approaches, we employ a set of metrics commonly used in prior works [34]. To evaluate effectiveness, we employ both ground-truth-based and structure-based metrics. For the datasets with labels, we report F1-score, Precision, Recall, and Jaccard Similarity (JAC). For the unlabeled datasets, we adopt average degree (Deg), internal density (Dens) [65], conductance (Cond) [5], and density modularity (DM) [43]. For efficiency, we report both offline and online running times. The offline time refers to index construction for cohesive-subgraph-based approaches, and data preprocessing and model training for learning-based approaches. Notably, the objective-function-based approaches and ICS-GNN do not require offline processing. Offline times exceeding 10^5 seconds are denoted as INF. For online times, given the high computational cost of certain objective-function-based approaches on large graphs, if the query time exceeds 10^3 seconds, we terminate the search and return the

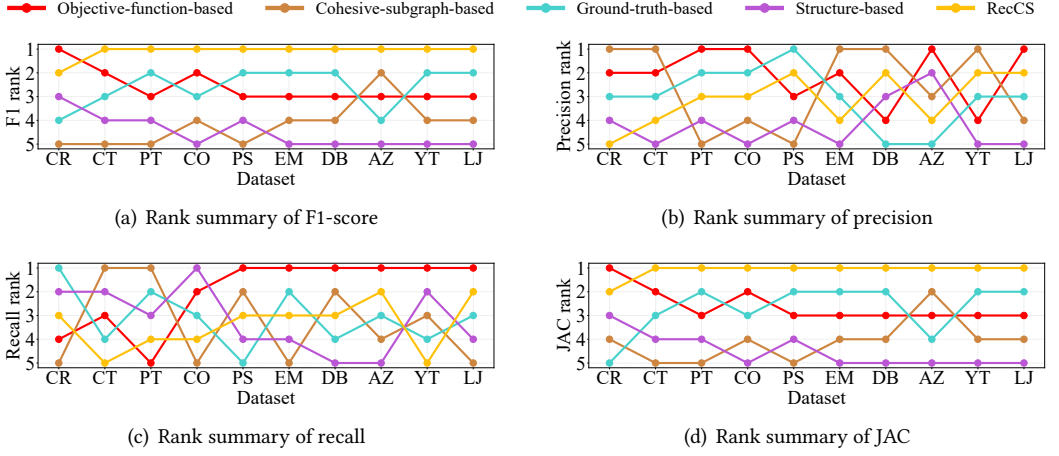


Fig. 2. Rank summary on labeled graphs

community with the best objective value found so far. For memory overhead, we report CPU and GPU memory usage during both the offline and online phases. Approaches exceeding memory limits are denoted as OOM.

Query generation. We split the ground-truth communities into training and testing communities using a 1:1 ratio [74]. The training and validation queries are randomly sampled from training communities, while the testing queries are randomly sampled from testing communities. Following [42, 74], we generate 150, 100, and 100 queries for training, validation, and testing, respectively. Finally, we report the average result over the 100 testing queries. Since some approaches do not support multiple query vertices [3, 5, 15, 16, 18, 32, 47, 76], we set $|Q| = 1$ by default.

Implementation setup. We implement all approaches in Python and conduct experiments on a Linux machine with Intel(R) Xeon(R) Gold 5520+ CPU, 128GB memory, and Nvidia RTX 4090 GPU. For the main hyperparameters of each approach, we adopt the settings reported in their papers or code repositories. For user-specified parameters, we select the optimal settings based on their F1-scores on the validation set. For learning-based approaches, we train the models for a maximum of 200 epochs and employ an early stopping strategy [74] to prevent overfitting. To ensure a fair comparison with non-learning-based approaches, we initialize vertex features with their core numbers [47]. For RecCS, we set $\lambda_{KL}^{param} = 1.0$, $\lambda_{KL}^{model} = 0.1$, $|V_{G^*}| = 500$, $k = 3$, and the batch size to 64 by default. The remaining training settings are kept consistent with those used for learning-based approaches to ensure a fair and unified comparison. We report the best F1-score among the top-3 recommendations, together with the running time and memory usage of the recommendation process.

Note. In the experiments, we use the abbreviated names for CS approaches and datasets as defined in Tables 1 and 2, respectively. Due to space limitations, some experiments report results only on the labeled datasets PT, AZ, and YT, and the unlabeled datasets CM, EN, and ND. The experimental results on the remaining datasets are provided in the appendix of our technical report [92].

5.2 Overall Comparison

In this section, we evaluate the overall performance of representative CS approaches and our RecCS model.

Effectiveness analysis on labeled graphs. We analyze the effectiveness of different approaches on labeled datasets using ground-truth-based metrics. For each category, we select the best-performing

Table 3. Effectiveness comparison on labeled graphs with ground-truth-based metrics¹

Category	Models	F1-score			Precision			Recall			JAC		
		PT	AZ	YT	PT	AZ	YT	PT	AZ	YT	PT	AZ	YT
Non-learning	QDC	0.227	0.718	0.414	<u>0.908*</u>	<u>0.984*</u>	0.513	0.192	0.640	0.436	0.188	0.633	0.300
	LM	0.458	0.543	0.259	0.505	0.493	0.202	0.466	0.884	0.475	0.356	0.400	0.159
	SGM	0.445	0.857	0.226	0.850	0.940	0.191	0.379	0.858	0.697	0.349	0.807	0.155
	DMCS	0.506	0.495	0.005	0.750	0.495	0.002	0.473	<u>0.941*</u>	0.949*	0.399	0.439	0.002
	APPR	0.256	0.774	0.317	0.445	0.969	0.693	0.283	0.694	0.436	0.203	0.683	0.208
	CSP	0.286	0.379	0.001	0.286	0.425	0.001	0.594	0.851	0.710	0.221	0.344	0.001
	ET	0.277	0.878	0.330	0.253	0.959	<u>0.990*</u>	<u>0.937*</u>	0.876	0.210	0.216	0.837	0.208
	KCPC	OOM	<u>0.878</u>	OOM	OOM	0.959	OOM	OOM	0.876	OOM	OOM	0.837	OOM
	SMCC	0.295	0.825	0.155	0.374	0.923	0.299	0.573	0.840	0.505	0.227	0.774	0.110
Learning	ICS	<u>0.559</u>	0.635	0.367	0.661	0.696	0.315	0.540	0.757	0.598	<u>0.440</u>	0.504	0.236
	QD	0.469	OOM	OOM	0.709	OOM	OOM	0.431	OOM	OOM	0.360	OOM	OOM
	CAF	OOM	0.706	INF	OOM	0.658	INF	OOM	0.904	INF	OOM	0.580	INF
	CDF	0.308	0.780	<u>0.421</u>	0.850	0.955	0.506	0.251	0.725	0.425	0.224	0.694	<u>0.304</u>
	CLP	0.403	0.853	0.414	0.327	0.900	0.713	0.772	0.882	0.409	0.300	0.789	0.288
	CSF	0.370	0.730	0.137	0.667	0.806	0.094	0.326	0.782	0.749	0.282	0.628	0.091
	TZ	0.321	0.600	0.274	0.309	0.963	0.666	0.722	0.519	0.275	0.243	0.500	0.178
Ours	RecCS	0.574*	0.912*	0.502*	0.682	0.955	0.719	0.590	0.909	0.492	0.451*	0.867*	0.370*

¹ We use **red**, **brown**, **teal**, and **purple** to highlight the best results for objective-function-based, cohesive-subgraph-based, ground-truth-based, and structure-based CS approaches, respectively. The best result among these baselines is underlined, and the best result across all approaches (including our RecCS) is marked with *.

Table 4. Percentage of optimal F1-score achievements across queries¹

Category	Models	CR	CT	PT	CO	PS	EM	DB	AZ	YT	LJ
Non-learning	QDC	3.99%	3.04%	4.81%	12.64%	6.85%	3.60%	<u>14.86%</u>	7.32%	<u>18.30%</u>	18.82%
	LM	7.26%	6.92%	10.74%	3.45%	7.26%	6.31%	5.41%	0.89%	0.00%	4.12%
	SGM	<u>10.71%</u>	<u>9.13%</u>	11.11%	<u>20.11%</u>	10.89%	18.02%	7.66%	12.64%	6.54%	18.82%
	DMCS	7.80%	6.22%	11.85%	13.22%	2.02%	3.60%	0.90%	5.10%	0.00%	2.35%
	APPR	3.45%	6.36%	4.44%	9.77%	8.06%	5.41%	4.95%	5.32%	9.15%	5.88%
	CSP	5.44%	5.53%	5.19%	0.00%	1.21%	0.00%	0.45%	3.77%	0.65%	1.18%
	ET	4.72%	1.52%	5.19%	8.05%	2.82%	5.41%	7.21%	<u>14.41%</u>	8.50%	11.18%
	KCPC	4.54%	6.92%	OOM	8.62%	2.82%	0.90%	13.06%	<u>14.41%</u>	OOM	OOM
	SMCC	5.44%	6.22%	5.19%	0.57%	3.23%	0.00%	8.11%	11.75%	3.27%	10.00%
Sum		53.36%	51.87%	58.52%	76.44%	45.16%	43.24%	62.61%	75.61%	46.41%	72.35%
Learning	ICS	7.62%	7.47%	<u>12.59%</u>	1.15%	0.00%	13.51%	2.25%	0.89%	14.38%	1.18%
	QD	7.26%	8.02%	8.89%	5.75%	14.92%	<u>22.52%</u>	OOM	OOM	OOM	OOM
	CAF	8.17%	6.92%	OOM	OOM	1.21%	4.50%	4.95%	1.77%	INF	INF
	CDF	2.54%	5.12%	2.22%	6.90%	8.06%	5.41%	7.66%	5.99%	17.65%	1.76%
	CLP	6.17%	7.47%	6.67%	4.02%	<u>16.13%</u>	10.81%	12.16%	10.20%	16.34%	<u>21.76%</u>
	CSF	6.72%	6.09%	5.56%	1.72%	2.42%	0.00%	1.80%	2.22%	1.31%	2.94%
	TZ	8.17%	7.05%	5.56%	4.02%	12.10%	0.00%	8.56%	3.33%	3.92%	OOM
Sum		46.64%	48.13%	41.48%	23.56%	54.84%	56.76%	37.39%	24.39%	53.59%	27.65%

¹ The two "Sum" rows represent the total percentage of queries where non-learning-based and learning-based approaches achieve the optimal F1-score, respectively. We highlight the higher aggregated percentage between these two categories in **bold**, and underline the individual CS approach with the highest percentage.

approach and summarize the overall rankings in Figure 2. Tables 3 and 4 report the detailed performance results and the percentage of queries for which each approach achieves the best F1-score. Notably, ET and KCPC achieve identical results on the AZ dataset because the parameter selected for both approaches on the validation set of AZ is $k = 3$, under which KCPC imposes the same structural constraint as ET. As shown in Figure 2 and Table 3, learning-based

Table 5. Effectiveness comparison on unlabeled graphs with structure-based metrics¹

Category	Models	Deg			Dens			Cond			DM		
		CM	EN	ND	CM	EN	ND	CM	EN	ND	CM	EN	ND
Non-learning	QDC	4.763	2.936	3.177	<u>0.922</u>	<u>0.889</u>	<u>0.880</u>	0.573	0.848	0.891	2.373	1.167	1.217
	LM	3.985	2.858	3.682	0.111	0.029	0.037	0.267	0.476	0.358	1.984	1.426	1.838
	SGM	4.346	4.794	4.080	0.473	0.283	0.156	0.298	0.456	0.194	2.169	1.442	2.037
	DMCS	8.811	27.485	12.404	0.184	0.042	0.035	0.435	0.631	0.281	3.854	6.795	5.909
	APPR	6.503	9.008	3.630	0.324	0.071	0.146	0.267	0.429	0.283	1.890	1.817	1.649
	CSP	<u>11.433</u>	<u>26.056</u>	10.607	0.108	0.005	0.013	0.288	<u>0.383</u>	0.095	1.591	0.786	3.169
	ET	6.973	4.062	4.953	0.342	0.005	0.053	0.404	0.968	0.687	1.318	0.333	2.208
	KCPC	4.735	OOM	4.938	0.450	OOM	0.055	0.632	OOM	0.688	2.079	OOM	2.201
	SMCC	8.909	15.749	<u>11.353</u>	0.305	0.166	0.048	0.445	0.757	0.463	1.742	0.964	2.825
Learning	ICS	<u>7.668</u>	<u>33.098</u>	6.015	0.167	0.083	0.043	0.655	0.842	0.780	3.547	6.084	2.790
	QD	2.175	1.189	OOM	0.367	0.184	OOM	0.656	0.807	OOM	0.989	0.288	OOM
	CAF	2.763	2.596	3.118	0.265	0.141	0.101	0.741	0.971	0.771	1.362	0.945	1.472
	CDF	6.547	20.763	6.824	0.137	0.210	0.069	0.632	0.951	0.878	3.117	-4.070	2.203
	CLP	6.766	7.193	6.433	0.002	0.001	0.002	<u>0.154</u>	<u>0.481</u>	<u>0.085</u>	2.325	2.105	3.045
	CSF	5.682	14.102	5.465	0.242	0.143	0.068	0.565	0.840	0.613	2.730	2.763	2.416
	TZ	3.924	3.845	2.425	0.274	0.083	0.066	0.455	0.905	0.656	0.986	0.269	1.187

¹ We use **red**, **brown**, **teal**, and **purple** to highlight the best results for objective-function-based, cohesive-subgraph-based, ground-truth-based, and structure-based CS approaches, respectively. The best result among them is underlined. For conductance, lower values indicate better performance, whereas for other metrics, higher values are preferred.

approaches do not consistently outperform non-learning-based ones. In contrast, our RecCS model surpasses individual CS approaches w.r.t. F1-score in most cases. Among the non-learning-based approaches, objective-function-based approaches generally achieve superior F1-scores, while among the learning-based approaches, ground-truth-based approaches exhibit comparatively better performance than structure-based approaches. Nevertheless, most existing CS approaches fail to maintain a good balance between precision and recall, and their performance varies considerably across different datasets. Moreover, Table 4 shows that the best-performing approach varies across queries, indicating that selecting an appropriate approach for each query can improve CS performance.

Effectiveness analysis on unlabeled graphs. We compare different approaches using structure-based metrics on unlabeled graphs. For approaches that rely on labels, the training process is omitted. By default, the parameters are selected based on density modularity. As shown in Table 5, non-learning-based CS approaches generally achieve better performance on structure-based metrics because their predefined community patterns are designed based on cohesive assumptions. For example, CSP achieves superior performance in terms of average degree by leveraging a core-based model, while QDC and DMCS perform better on internal density and density modularity, respectively, due to their explicit optimization of query-biased density and density modularity. Notably, CLP achieves superior conductance performance by leveraging graph partitioning, which effectively reduces the number of cut edges. Our results show that most non-learning-based approaches are effective at identifying cohesive subgraphs.

Efficiency analysis. We evaluate the offline time and online time of different CS approaches and our RecCS, as shown in Figure 3. Figure 3(a) shows the time comparison during the offline phase. Learning-based approaches generally incur higher computational costs due to the data processing and model training. However, non-learning-based approaches are more sensitive to graph size, as the index construction becomes increasingly expensive on large graphs. During the online phase, as shown in Figure 3(b), the query time of objective-function-based approaches is not stable across datasets. This is because they rely on iterative optimization procedures that are easily affected by graph cohesiveness. In contrast, cohesive-subgraph-based approaches and

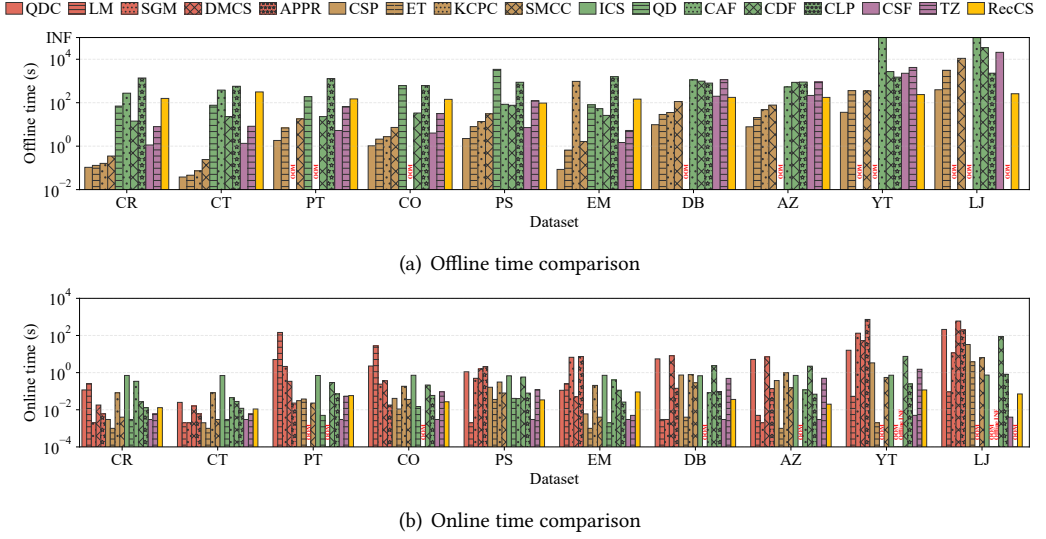


Fig. 3. Time comparison of representative CS approaches

learning-based approaches generally exhibit more stable query time by leveraging pre-built indexes or models. For RecCS, both the offline and online phases exhibit stable running times across different datasets. This is because we employ the subgraph sampling strategy, which effectively reduces computational overhead. Moreover, RecCS achieves recommendation times of less than one second on all datasets, indicating that recommending top- k approaches incurs minor overhead.

Memory analysis. We analyze the memory usage of different CS approaches in both the offline and online phases. Given the distinct computational paradigms of non-learning-based and learning-based approaches, we report CPU memory usage for non-learning-based CS approaches and GPU memory usage for learning-based CS approaches. For RecCS, we report its GPU memory usage. Overall, as shown in Figure 4, compared to existing CS approaches, RecCS maintains moderate GPU memory usage in both phases, exhibiting superior scalability on large graphs.

In the offline phase, Figure 4(a) shows that the CPU memory usage of cohesive-subgraph-based approaches increases with graph size. This is because index construction becomes computationally expensive on dense and large graphs, especially for KCPC. In contrast, as depicted in Figure 4(b), the GPU memory usage of most learning-based approaches remains relatively stable due to strategies like subgraph sampling [16, 32] or graph partitioning [47]. However, QD and CAF are exceptions. This is because QD requires full-graph training, whereas CAF is sensitive to the ground-truth community size.

In the online phase, as illustrated in Figure 4(c), cohesive-subgraph-based approaches consume slightly more CPU memory than objective-function-based approaches. This is because they need to load pre-built indexes. For learning-based approaches shown in Figure 4(d), the GPU memory usage of most learning-based approaches remains stable, with the exceptions of QD and CLP. This is because QD requires full-graph inference, while CLP involves merging multiple partitions to search the community.

Observations. Our analysis demonstrates that objective-function-based and ground-truth-based approaches generally achieve superior performance w.r.t. F1-score. However, their applicability is often limited by the online iterative optimization cost and the offline computational overhead, respectively. In addition, we find that non-learning-based approaches are effective at identifying cohesive subgraphs on unlabeled graphs. Since none of the CS approaches consistently outperform

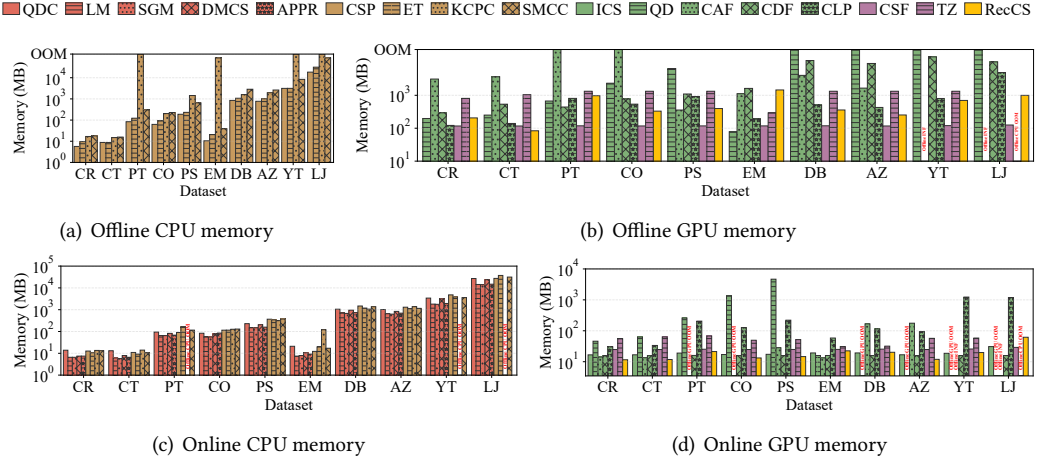


Fig. 4. Memory comparison of representative CS approaches

the others, RecCS improves CS performance by adaptively selecting suitable approaches for each query, while incurring only minor overhead during both offline and online phases.

5.3 Impact Analysis of Ground-truth Communities

We analyze how the characteristics of the ground-truth communities for these queries influence CS performance. We focus on two representative factors: community size and cohesiveness (measured by internal density [65]).

Ground-truth communities visualization. Figure 5 provides a visualization of ground-truth communities, where vertex size and color denote community size and internal density, respectively. Larger variations in vertex sizes indicate greater disparities in ground-truth community sizes, while more diverse vertex colors reflect higher heterogeneity in community cohesiveness. The results reveal that structural heterogeneity in ground-truth communities is prevalent across diverse datasets, indicating that effective CS approaches must be capable of capturing the distinct nature of the ground-truth communities for varying queries. Moreover, we observe that in datasets with classification labels (e.g., PT), ground-truth communities exhibit greater variation in size compared to those with community labels. This is because vertices sharing the same classification label are not necessarily connected and may form multiple connected components of varying sizes. In this paper, we decompose these classes into connected components to serve as ground-truth communities.

Impact of ground-truth community size. We sort the ground-truth communities in ascending order by community size and partition them into five quantiles. Figure 6 illustrates the impact of community size on performance by plotting the cumulative average F1-score across these size quantiles. First, existing CS approaches exhibit strong sensitivities to community size, while they demonstrate consistent performance patterns across datasets. As ground-truth community sizes vary, their F1-scores typically exhibit monotonic trends or form inflection points. This is because the predefined or learned patterns within these approaches cannot universally adapt to varying community sizes. Specifically, performance tends to peak when the ground-truth community closely matches the assumed pattern, and declines as the mismatch increases. Second, we find that the degree of sensitivity is largely governed by structural heterogeneity. For example, the PT dataset exhibits substantially greater variance in community size (Figure 5(c)), leading to more pronounced sensitivity to size variations. These observations indicate that existing approaches struggle to generalize flexibly to ground-truth communities of varying sizes.

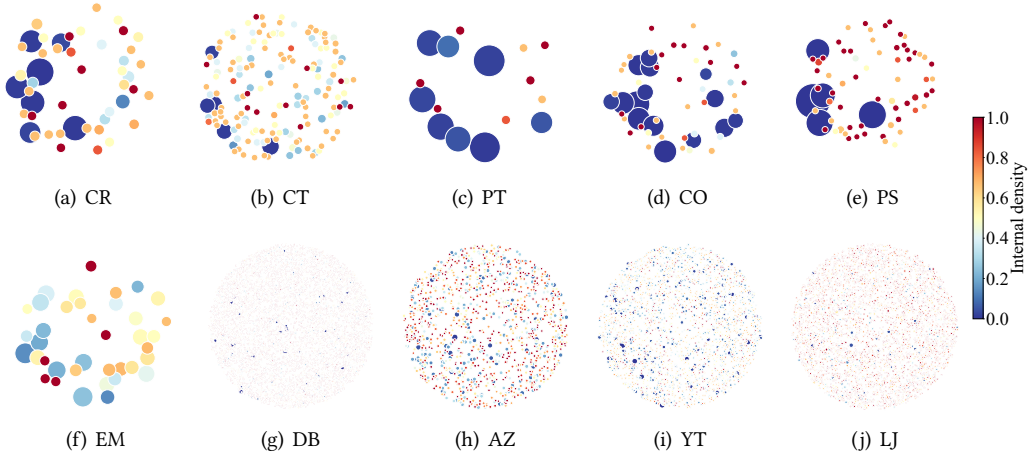


Fig. 5. Visualization of ground-truth communities (vertex size: community size; color: internal density)

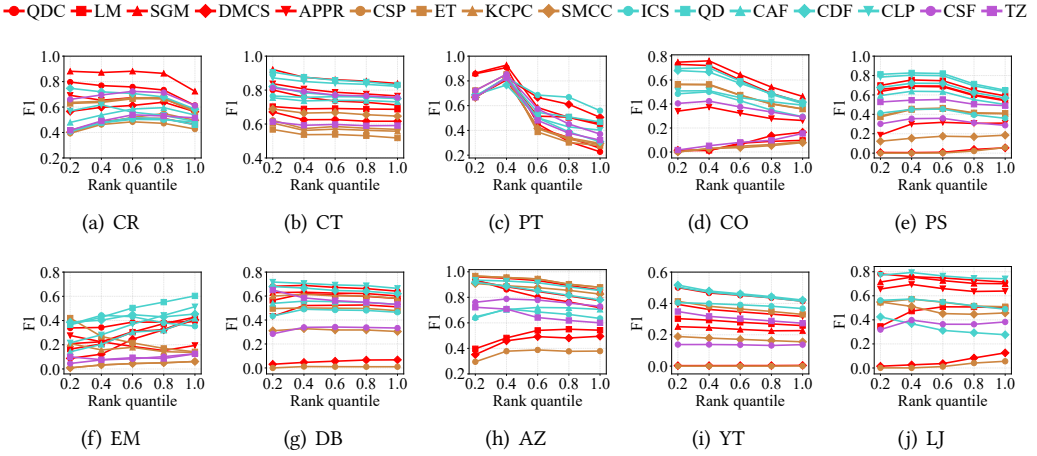


Fig. 6. Impact of community size

Impact of cohesiveness of ground-truth communities. We sort the ground-truth communities in ascending order by internal density and partition them into five quantiles. Figure 7 illustrates the impact of community cohesiveness on performance by plotting the cumulative average F1-score across these quantiles. Similar to the observations of community size, existing CS approaches exhibit strong sensitivity to community cohesiveness. As the internal density of ground-truth communities varies, their F1-scores generally follow monotonic trends or form inflection points. These approaches inherently rely on assumptions of structural cohesiveness. Consequently, performance tends to peak when the ground-truth community conforms to the assumed cohesive structure and declines as it increasingly deviates from this structure. The results show that existing approaches exhibit specific structural biases and struggle to generalize flexibly across communities with varying levels of cohesiveness.

Observations. Our analysis shows that real-world communities exhibit structural heterogeneity. However, existing CS approaches are sensitive to the characteristics of ground-truth communities, with the degree of sensitivity directly driven by the variance in these characteristics. Moreover,

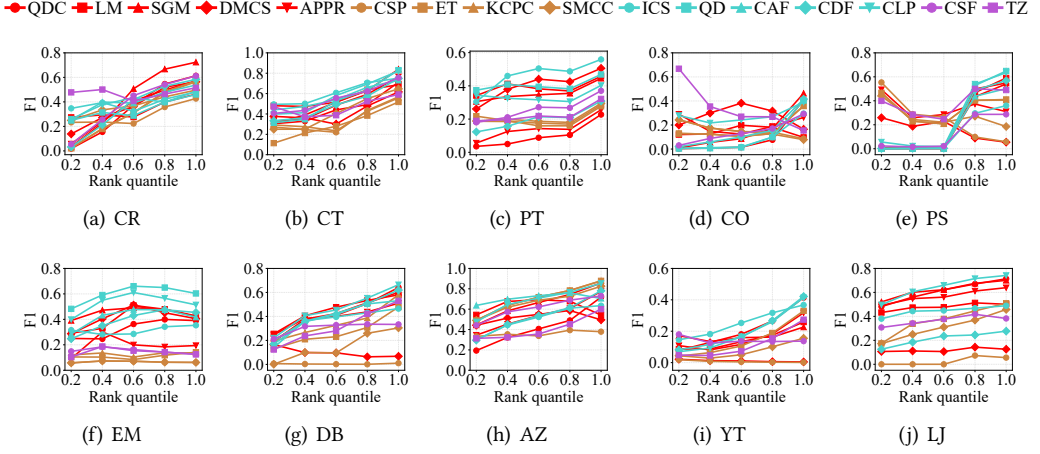


Fig. 7. Impact of community cohesiveness

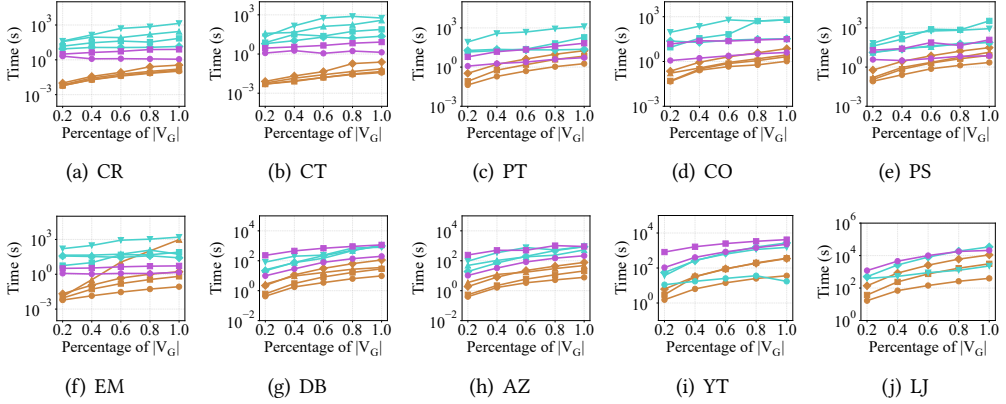


Fig. 8. Scalability analysis of offline phase

this sensitivity often appears as monotonic trends or inflection points, which reflect the poor generalizability of these approaches across diverse community structures and reveal the limitations of their underlying rigid structural assumptions. Therefore, the design of CS approaches should account for the structural characteristics of the ground-truth communities.

5.4 Generalizability Evaluation

In this section, we further investigate the generalizability of different CS approaches by examining sensitivity and robustness under various graph and query configurations.

Scalability analysis. For scalability evaluation, we extract varying fractions of vertices from the original graphs to construct induced subgraphs of different sizes. Figures 8 and 9 show the running time for the offline phase and the online phase, respectively. During the offline phase, the running time of both cohesive-subgraph-based and learning-based approaches generally increases steadily with graph size. However, the learning-based approaches generally incur a significantly higher computational overhead. Notably, the training time of learning-based approaches may fluctuate due to early stopping mechanisms used to prevent overfitting. During the online phase, the cohesive-subgraph-based approaches and learning-based approaches demonstrate robust scalability by leveraging pre-built indexes or models, with query time typically increasing smoothly as the

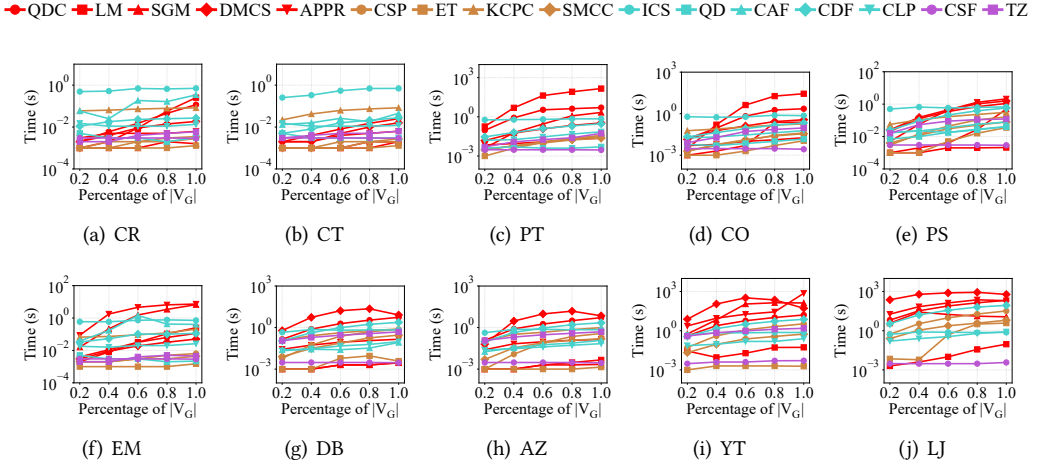


Fig. 9. Scalability analysis of online phase

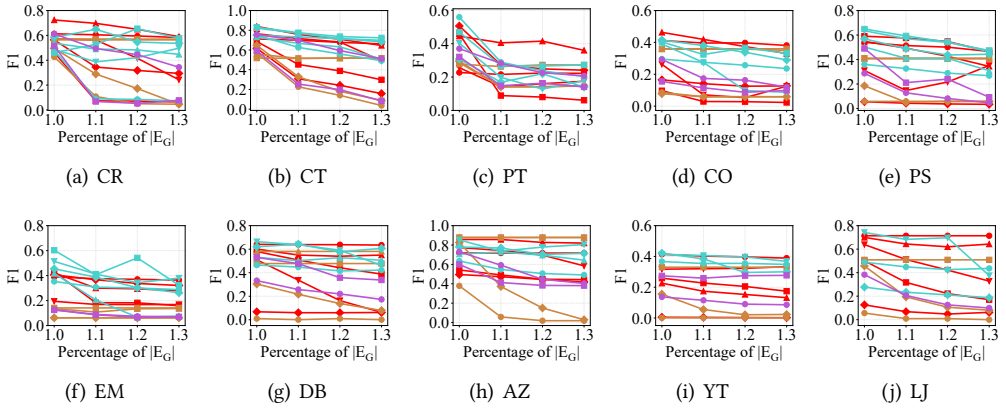
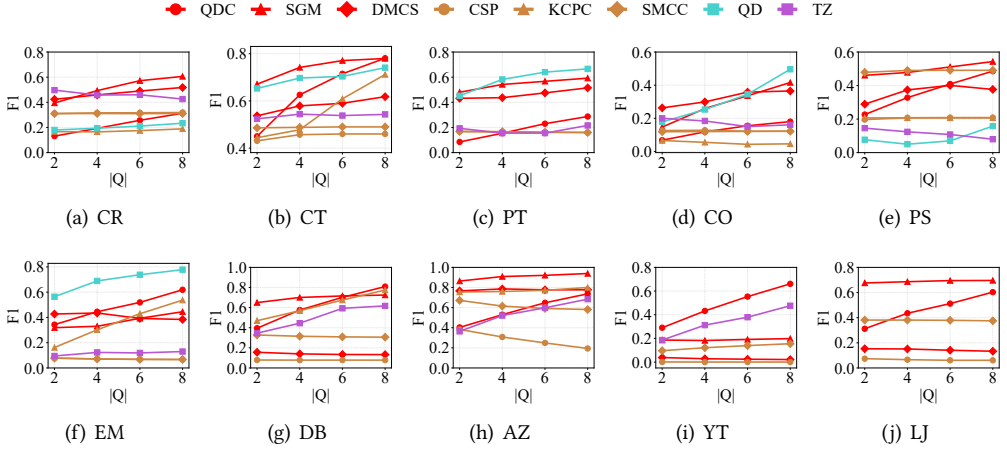


Fig. 10. Robustness analysis of graph structure

graph size grows. Conversely, objective-function-based approaches, which rely on iterative online optimization, are more sensitive to the graph size. Besides, the degree of this sensitivity varies across datasets.

Robustness analysis. We evaluate the robustness of different approaches under graph structure perturbations. Specifically, we introduce noise by randomly inserting 10%, 20%, and 30% additional edges. Note that we do not consider edge deletion as it may disconnect ground-truth communities. This would lead to variations in the connected ground-truth communities used for evaluation for the same query under different noise ratios, thereby compromising the fairness of robustness comparisons. As shown in Figure 10, the F1-scores of most approaches typically degrade as the level of noise increases. Meanwhile, learning-based approaches do not consistently exhibit stronger robustness than non-learning-based approaches, even when they are trained with ground-truth communities. This is because structural noise may distort the underlying structural patterns learned from the ground-truth communities.

Effect of $|Q|$. We analyze the impact of $|Q|$ on CS approaches that support multiple query vertices by varying $|Q|$ from 2 to 8. Compared with non-learning-based approaches, most learning-based approaches do not support multiple query vertices. The results are shown in Figure 11. We observe

Fig. 11. Effect of $|Q|$ Table 6. F1-score under different query vertex location configurations¹

Category	Models	PT		AZ		YT		Avg. $\Delta F1$
		Boundary	Internal	Boundary	Internal	Boundary	Internal	
Non-learning	QDC	0.089	0.078	0.699	0.683	0.265	0.344	+0.017
	LM	0.388	0.414	0.528	0.563	0.200	0.281	+0.047
	SGM	0.406	0.332	0.823	0.852	0.178	0.321	+0.033
	DMCS	0.410	0.547	0.451	0.620	0.019	0.024	+0.104
	APPR	0.171	0.141	0.765	0.767	0.205	0.232	0.000
	CSP	0.183	0.163	0.370	0.412	0.008	0.001	+0.005
	ET	0.157	0.158	0.823	0.855	0.183	0.183	+0.011
	KCPC	OOM	OOM	0.813	0.854	OOM	OOM	+0.041
Learning	SMCC	0.193	0.182	0.812	0.815	0.088	0.214	+0.039
	ICS	0.492	0.572	0.595	0.663	0.267	0.360	+0.080
	QD	0.370	0.423	OOM	OOM	OOM	OOM	+0.053
	CAF	OOM	OOM	0.680	0.720	INF	INF	+0.040
	CDF	0.213	0.212	0.770	0.771	0.264	0.327	+0.021
	CLP	0.281	0.338	0.768	0.855	0.252	0.317	+0.070
	CSF	0.318	0.325	0.607	0.741	0.112	0.372	+0.134
	TZ	0.229	0.209	0.502	0.602	0.156	0.195	+0.040

¹ Avg. $\Delta F1$ denotes the average F1-score difference between internal and boundary query vertices.

that objective-function-based approaches and QD can leverage the query information provided by multiple query vertices, generally exhibiting a positive correlation between F1-score and $|Q|$. However, the remaining approaches fail to exhibit a consistent upward trend across datasets, indicating a limited capacity to effectively exploit the additional guidance from multiple query vertices.

Effect of query vertex location. To explore the impact of query vertex location, we sample query vertices from both boundary and internal vertices of ground-truth communities, where boundary vertices have edges connecting to vertices outside the community, whereas internal vertices have no such external connections [18]. As shown in Table 6, most approaches achieve better performance with internal query vertices than with boundary ones, suggesting that internal query vertices facilitate accurate community identification. Furthermore, learning-based approaches are generally more sensitive to query vertex location than non-learning-based approaches. This is because most non-learning-based approaches merely require the inclusion of the query vertices. In

Table 7. Ablation study on F1-score

Graphs	Full Model	w/o QAE	w/o KL
CR	0.715	0.725	0.676
CT	0.863	0.822	0.863
PT	0.574	0.433	0.550
CO	0.582	0.516	0.586
PS	0.660	0.730	0.686
EM	0.665	0.662	0.553
DB	0.739	0.725	0.475
AZ	0.912	0.890	0.777
YT	0.502	0.478	0.405
LJ	0.785	0.795	0.609
Average	-	-3.69%	-11.41%

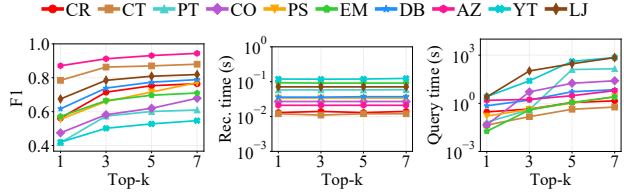
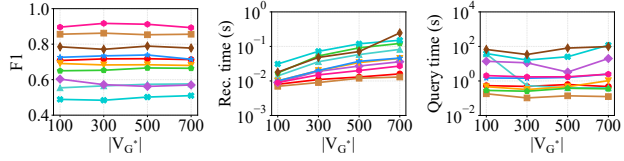


Fig. 12. Effect of top-k

Fig. 13. Effect of $|V_{G^*}|$

contrast, learning-based approaches learn the local structure around query vertices to identify the communities.

Observations. Our analysis reveals that CS approaches exhibit distinct limitations in generalizability. In terms of graph generalizability, while learning-based approaches achieve stable online efficiency, they incur high offline costs and do not generalize better to structural noise than non-learning-based approaches. Regarding query generalizability, most existing approaches fail to effectively leverage query information to support CS with multiple query vertices and are sensitive to query vertex locations.

5.5 Analysis of RecCS

In this section, we conduct the ablation study and sensitivity analysis of RecCS.

Ablation study. We compare RecCS with two variants: (i) w/o QAE, which replaces the output of the query-aware encoder with zero vectors, and (ii) w/o KL, which removes the KL divergence loss. As shown in Table 7, removing the query-aware encoder leads to a 3.69% drop in average F1-score, confirming that local structural features of query vertices benefit the recommendation. Furthermore, eliminating the KL divergence loss results in a 11.41% F1-score degradation, validating the necessity of distribution alignment in enhancing the recommendation accuracy. On certain datasets, removing these components may occasionally lead to performance improvements. This can be attributed to indistinct local structural features of query vertices and distribution shifts between training and testing data in those datasets. Nonetheless, RecCS remains robust overall, effectively providing top-performing recommendations.

Effect of the number of recommended approaches. In this experiment, we analyze the sensitivity of RecCS w.r.t. the number of recommended approaches k . We report the F1-score, recommendation time, and the total query time of recommended approaches. As shown in Figure 12, the F1-score shows a clear improvement as k increases from 1 to 3 and then stabilizes, indicating that RecCS is able to rank top-performing approaches at the top of the recommendation list. Additionally, the recommendation time remains stable across different k values. In contrast, the total query time increases with k , as a larger k necessitates the execution of more CS approaches. Besides, the rate of this increase varies across datasets, which mainly depends on the query time required by the recommended CS approaches. Consequently, we recommend selecting the top-3 approaches by default to strike a balance between query efficiency and community quality.

Effect of $|V_{G^*}|$. In the experiment, we analyze the sensitivity of RecCS w.r.t. the candidate subgraph size $|V_{G^*}|$. Figure 13 shows the impact of candidate subgraph size on community quality, recommendation time, and total query time of recommended approaches. As $|V_{G^*}|$ increases, the F1-score

shows small fluctuations but remains generally stable, suggesting that small candidate subgraphs are sufficient to capture the local structural features of query vertices. Besides, the recommendation time of RecCS increases steadily with $|V_{G^*}|$, demonstrating that subgraph sampling can improve recommendation efficiency while exerting only a minor impact on performance. Additionally, the total query time fluctuates without a clear trend since it depends on the runtime of the specific recommended CS approaches rather than the subgraph size itself.

Observations. Our ablation study demonstrates that the query-aware encoder of RecCS effectively captures the local structural features of query vertices. Additionally, it validates that performance distribution alignment enhances recommendation accuracy. Moreover, the sensitivity analysis confirms that RecCS achieves a balance between runtime and recommendation quality, improving the CS performance with minor computational overhead.

5.6 Discussion

Guidelines. To facilitate real-world application, we provide practical guidelines for model selection. Among non-learning-based approaches, SGM is well suited to parameter-free settings, whereas ET is a reliable choice when appropriate structural parameters are available. Among learning-based approaches, CLP offers stable performance when ground-truth labels are available, while CSF is better suited to unsupervised settings.

Limitations. Despite the advances of these approaches, as evidenced in Section 5.2, no single approach is universally applicable across diverse scenarios, highlighting a fundamental limitation of existing CS approaches: lack of generalizability.

For non-learning-based approaches, they rely on predefined community patterns, which typically emphasize high internal cohesiveness or sparse external connections. If the ground-truth communities resemble their assumed patterns, these approaches can achieve strong performance; otherwise, their effectiveness may be limited when the ground-truth community structures deviate from the predefined patterns.

For learning-based approaches, their generalizability is constrained by limited model capacity and restrictive assumptions. First, to ensure scalability, these approaches typically rely on localized sampling strategies, which limit their global receptive fields and result in performance variation across queries. Second, they generally assume that the training and test data follow similar distributions. However, real-world communities often exhibit structural heterogeneity, causing test communities to deviate from the training distribution and preventing learned patterns from generalizing reliably across diverse queries.

Ultimately, our empirical analysis suggests that the question of “*To Learn or Not to Learn?*” yields no single dominant strategy. Instead, both paradigms suffer from distinct generalizability limitations across diverse scenarios. Therefore, the pivotal challenge shifts from choosing a single paradigm to the adaptive selection of approaches across varying queries, graphs, and scenarios.

Future works. Motivated by these limitations, a key direction for future research is the design of generalizable CS frameworks. Specifically, we envision unified CS frameworks that robustly generalize across three dimensions: (i) *query generalizability*: handling diverse query configurations, including variable sizes, query vertex locations, and structural heterogeneity of the corresponding ground-truth communities; (ii) *graph generalizability*: maintaining robustness across graphs with different scales, data quality, and structural properties; and (iii) *scenario generalizability*: adapting to multi-dimensional user constraints, such as strict runtime and memory limits.

To this end, integrating Large Language Models (LLMs) presents a highly promising direction. Given their strong zero-shot reasoning capabilities, future research could investigate LLM-based autonomous agents to address the challenge of generalizable CS. By analyzing graph topological

features, query characteristics, and scenario-specific constraints, such agents could dynamically recommend, parameterize, or synthesize the most suitable CS approach. Furthermore, exploring effective prompting strategies that enable LLMs to reason directly over complex graph structures may ultimately pave the way for generalizable CS, alleviating the rigid assumptions inherent in existing paradigms.

6 Conclusion

In this paper, we present an in-depth empirical analysis of community search approaches, covering both non-learning-based and learning-based paradigms. Our study provides a comprehensive comparison of existing CS approaches under various settings and reveals that no single approach dominates across all queries. Moreover, most existing approaches are sensitive to the structural heterogeneity of the ground-truth communities and exhibit limited generalizability. Motivated by these observations, we propose RecCS, a query-aware recommendation model that selects top-performing CS approaches for each query, thereby achieving improved CS performance efficiently. For future work, we advocate the development of generalizable LLM-based CS frameworks that can robustly handle diverse query configurations, varying graph characteristics, and multi-dimensional user constraints.

Acknowledgments

This work was supported in part by the NSFC under Grants No. (62302444, 62025206, and U23A20296), and Zhejiang Province's "Lingyan" R&D Project under Grant No. 2024C01259.

References

- [1] Balázs Adamcsek, Gergely Palla, Illés J. Farkas, Imre Derényi, and Tamás Vicsek. 2006. CFinder: locating cliques and overlapping modules in biological networks. *Bioinform.* 22, 8 (2006), 1021–1023.
- [2] Wei Ai, CanHao Xie, Tao Meng, Jiayi Du, and Keqin Li. 2024. A D-Truss-Equivalence Based Index for Community Search Over Large Directed Graphs. *IEEE Trans. Knowl. Data Eng.* 36, 11 (2024), 5482–5494.
- [3] Esra Akbas and Peixiang Zhao. 2017. Truss-based Community Search: a Truss-equivalence Based Indexing Approach. *Proc. VLDB Endow.* 10, 11 (2017), 1298–1309.
- [4] Takuya Akiba, Yoichi Iwata, and Yuichi Yoshida. 2013. Linear-time enumeration of maximal K-edge-connected subgraphs in large networks by random contraction. In *CIKM*. 909–918.
- [5] Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. 2006. Local Graph Partitioning using PageRank Vectors. In *FOCS*. 475–486.
- [6] Reid Andersen and Kevin J. Lang. 2006. Communities from seed sets. In *WWW*. 223–232.
- [7] Nicola Barbieri, Francesco Bonchi, Edoardo Galimberti, and Francesco Gullo. 2015. Efficient and effective community search. *Data Min. Knowl. Discov.* 29, 5 (2015), 1406–1433.
- [8] Fei Bi, Lijun Chang, Xuemin Lin, and Wenjie Zhang. 2018. An Optimal and Progressive Approach to Online Search of Top-K Influential Communities. *Proc. VLDB Endow.* 11, 9 (2018), 1056–1068.
- [9] Mauro Brunato, Holger H. Hoos, and Roberto Battiti. 2007. On Effectively Finding Maximal Quasi-cliques in Graphs. In *LION (Lecture Notes in Computer Science, Vol. 5313)*. 41–55.
- [10] Lijun Chang, Xuemin Lin, Lu Qin, Jeffrey Xu Yu, and Wenjie Zhang. 2015. Index-based Optimal Algorithms for Computing Steiner Components with Maximum Connectivity. In *SIGMOD Conference*. 459–474.
- [11] Lijun Chang, Jeffrey Xu Yu, Lu Qin, Xuemin Lin, Chengfei Liu, and Weifa Liang. 2013. Efficiently computing k-edge connected components via graph decomposition. In *SIGMOD Conference*. 205–216.
- [12] Xueqin Chang, Qing Liu, Yunjun Gao, Baihua Zheng, Yi Cai, and Qing Li. 2025. The Most Influenced Community Search on Social Networks. In *ICDE*. 2601–2614.
- [13] Jiazun Chen, Jun Gao, and Bin Cui. 2023. ICS-GNN⁺: lightweight interactive community search via graph neural network. *VLDB J.* 32, 2 (2023), 447–467.
- [14] Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. 2023. NAGphormer: A Tokenized Graph Transformer for Node Classification in Large Graphs. In *ICLR*.
- [15] Jiazun Chen, Yikuan Xia, and Jun Gao. 2023. CommunityAF: An Example-based Community Search Method via Autoregressive Flow. *Proc. VLDB Endow.* 16, 10 (2023), 2565–2577.

- [16] Jiazun Chen, Yikuan Xia, Jun Gao, Zhao Li, and Hongyang Chen. 2025. CommunityDF: A Guided Denoising Diffusion Approach for Community Search. In *ICDE*. 460–473.
- [17] Zi Chen, Yiwei Zhao, Long Yuan, Xuemin Lin, and Kai Wang. 2023. Index-Based Biclique Percolation Communities Search on Bipartite Graphs. In *ICDE*. 2699–2712.
- [18] Aaron Clauset. 2005. Finding local community structure in networks. *Physical Review E* 72, 2 (2005), 026132.
- [19] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yiqi Lu, and Wei Wang. 2013. Online search of overlapping communities. In *SIGMOD Conference*. 277–288.
- [20] Wanyun Cui, Yanghua Xiao, Haixun Wang, and Wei Wang. 2014. Local search of communities in large graphs. In *SIGMOD Conference*. 991–1002.
- [21] Yizhou Dai, Miao Qiao, and Rong-Hua Li. 2024. On Density-based Local Community Search. *Proc. ACM Manag. Data* 2, 2 (2024), 88.
- [22] Thang N. Dinh, Xiang Li, and My T. Thai. 2015. Network Clustering via Maximizing Modularity: Approximation Algorithms and Theoretical Limits. In *ICDM*. 101–110.
- [23] Shuheng Fang, Kangfei Zhao, Guanghua Li, and Jeffrey Xu Yu. 2023. Community Search: A Meta-Learning Approach. In *ICDE*. 2358–2371.
- [24] Shuheng Fang, Kangfei Zhao, Yu Rong, Zhixun Li, and Jeffrey Xu Yu. 2024. Inductive Attributed Community Search: to Learn Communities across Graphs. *Proc. VLDB Endow.* 17, 10 (2024), 2576–2589.
- [25] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. 2020. A survey of community search over big graphs. *VLDB J.* 29, 1 (2020), 353–392.
- [26] Yixiang Fang, Zhongran Wang, Reynold Cheng, Hongzhi Wang, and Jiafeng Hu. 2019. Effective and Efficient Community Search Over Large Directed Graphs. *IEEE Trans. Knowl. Data Eng.* 31, 11 (2019), 2093–2107.
- [27] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. 2020. Effective and Efficient Community Search over Large Heterogeneous Information Networks. *Proc. VLDB Endow.* 13, 6 (2020), 854–867.
- [28] Yixiang Fang, Kaiqiang Yu, Reynold Cheng, Laks V. S. Lakshmanan, and Xuemin Lin. 2019. Efficient Algorithms for Densest Subgraph Discovery. *Proc. VLDB Endow.* 12, 11 (2019), 1719–1732.
- [29] Zijin Feng, Miao Qiao, and Hong Cheng. 2023. Modularity-based Hypergraph Clustering: Random Hypergraph Model, Hyperedge-cluster Relation, and Computation. *Proc. ACM Manag. Data* 1, 3 (2023), 215:1–215:25.
- [30] Gary William Flake, Steve Lawrence, and C. Lee Giles. 2000. Efficient identification of Web communities. In *KDD*. 150–160.
- [31] Santo Fortunato and Marc Barthélemy. 2007. Resolution limit in community detection. *Proc. Natl. Acad. Sci.* 104, 1 (2007), 36–41.
- [32] Jun Gao, Jiazun Chen, Zhao Li, and Ji Zhang. 2021. ICS-GNN: Lightweight Interactive Community Search via Graph Neural Network. *Proc. VLDB Endow.* 14, 6 (2021), 1006–1018.
- [33] Xiaoxuan Gou, Xiaoliang Xu, Xiangying Wu, Runhuai Chen, Yuxiang Wang, Tianxing Wu, and Xiangyu Ke. 2023. Effective and Efficient Community Search with Graph Embeddings. In *ECAL*, Vol. 372. 891–898.
- [34] Xiaoxuan Gou, Weiguo Zheng, Yuxiang Wang, Xiaoliang Xu, and Zhiyuan Yu. 2025. A Comprehensive Survey and Experimental Study of Learning-based Community Search. *Proc. VLDB Endow.* 18, 9 (2025), 2941–2954.
- [35] Farnoosh Hashemi, Ali Behrouz, and Milad Rezaei Hajidehi. 2023. CS-TGN: Community Search via Temporal Graph Neural Networks. In *WWW (Companion Volume)*. 1196–1203.
- [36] Yue He, Longlong Lin, Pingpeng Yuan, Ronghua Li, Tao Jia, and Zeli Wang. 2024. CCSS: Towards conductance-based community search with size constraints. *Expert Syst. Appl.* 250 (2024), 123915.
- [37] Jiafeng Hu, Xiaowei Wu, Reynold Cheng, Siqiang Luo, and Yixiang Fang. 2016. Querying Minimal Steiner Maximum-Connected Subgraphs in Large Graphs. In *CIKM*. 1241–1250.
- [38] Jiafeng Hu, Xiaowei Wu, Reynold Cheng, Siqiang Luo, and Yixiang Fang. 2017. On Minimal Steiner Maximum-Connected Subgraph Queries. *IEEE Trans. Knowl. Data Eng.* 29, 11 (2017), 2455–2469.
- [39] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. 2014. Querying k-truss community in large and dynamic graphs. In *SIGMOD Conference*. 1311–1322.
- [40] Xin Huang and Laks V. S. Lakshmanan. 2017. Attribute-Driven Community Search. *Proc. VLDB Endow.* 10, 9 (2017), 949–960.
- [41] Xin Huang, Laks V. S. Lakshmanan, Jeffrey Xu Yu, and Hong Cheng. 2015. Approximate Closest Community Search in Networks. *Proc. VLDB Endow.* 9, 4 (2015), 276–287.
- [42] Yuli Jiang, Yu Rong, Hong Cheng, Xin Huang, Kangfei Zhao, and Junzhou Huang. 2022. Query Driven-Graph Neural Networks for Community Search: From Non-Attributed, Attributed, to Interactive Attributed. *Proc. VLDB Endow.* 15, 6 (2022), 1243–1255.
- [43] Junghoon Kim, Siqiang Luo, Gao Cong, and Wenyan Yu. 2022. DMCS : Density Modularity based Community Search. In *SIGMOD Conference*. 889–903.
- [44] Kyle Kloster and David F. Gleich. 2014. Heat kernel based community detection. In *KDD*. 1386–1395.

- [45] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *Ann. Math. Stat.* 22, 1 (1951), 79–86.
- [46] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. <http://snap.stanford.edu/data>.
- [47] Ling Li, Siqiang Luo, Yuhai Zhao, Caihua Shan, Zhengkui Wang, and Lu Qin. 2023. COCLEP: Contrastive Learning-based Semi-Supervised Community Search. In *ICDE*. 2483–2495.
- [48] Rong-Hua Li, Lu Qin, Jeffrey Xu Yu, and Rui Mao. 2015. Influential Community Search in Large Networks. *Proc. VLDB Endow.* 8, 5 (2015), 509–520.
- [49] Rong-Hua Li, Jiao Su, Lu Qin, Jeffrey Xu Yu, and Qiangqiang Dai. 2018. Persistent Community Search in Temporal Networks. In *ICDE*. 797–808.
- [50] Yuan Li, Xiuxu Chen, Yuhai Zhao, Wen Shan, Zhengkui Wang, Guoli Yang, and Guoren Wang. 2024. Self-Training GNN-based Community Search in Large Attributed Heterogeneous Information Networks. In *ICDE*. 2765–2778.
- [51] Xuankun Liao, Qing Liu, Xin Huang, and Jianliang Xu. 2024. Truss-based Community Search over Streaming Directed Graphs. *Proc. VLDB Endow.* 17, 8 (2024), 1816–1829.
- [52] Longlong Lin, Yue He, Wei Chen, Pingpeng Yuan, Rong-Hua Li, and Tao Jia. 2025. Effective and Efficient Conductance-Based Community Search at Billion Scale. *IEEE Trans. Big Data* 11, 6 (2025), 3170–3184.
- [53] Longlong Lin, Pingpeng Yuan, Rong-Hua Li, Chunxue Zhu, Hongchao Qin, Hai Jin, and Tao Jia. 2024. QTCS: Efficient Query-Centered Temporal Community Search. *Proc. VLDB Endow.* 17, 6 (2024), 1187–1199.
- [54] Qing Liu, Minjun Zhao, Xin Huang, Jianliang Xu, and Yunjun Gao. 2020. Truss-based Community Search over Large Directed Graphs. In *SIGMOD Conference*. 2183–2197.
- [55] Qing Liu, Yifan Zhu, Minjun Zhao, Xin Huang, Jianliang Xu, and Yunjun Gao. 2020. VAC: Vertex-Centric Attributed Community Search. In *ICDE*. 937–948.
- [56] Chengyang Luo, Qing Liu, Yunjun Gao, and Jianliang Xu. 2025. Synergetic Community Search over Large Multilayer Graphs. *Proc. VLDB Endow.* 18, 5 (2025), 1412–1424.
- [57] Feng Luo, James Zijun Wang, and Eric Promislow. 2006. Exploring Local Community Structures in Large Networks. In *Web Intelligence*. 233–239.
- [58] Michael Mitzenmacher, Jakub Pachocki, Richard Peng, Charalampos E. Tsourakakis, and Shen Chen Xu. 2015. Scalable Large Near-Clique Detection in Large-Scale Networks via Sampling. In *KDD*. 815–824.
- [59] Mark EJ Newman. 2006. Modularity and community structure in networks. *Proc. Natl. Acad. Sci.* 103, 23 (2006), 8577–8582.
- [60] Mark EJ Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical review E* 69, 2 (2004), 026113.
- [61] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *nature* 435, 7043 (2005), 814–818.
- [62] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*. 8024–8035.
- [63] Paola Pesantez-Cabrera and Ananth Kalyanaraman. 2019. Efficient Detection of Communities in Biological Bipartite Networks. *IEEE ACM Trans. Comput. Biol. Bioinform.* 16, 1 (2019), 258–271.
- [64] Yanjun Qi, Fernanda Balem, Christos Faloutsos, Judith Klein-Seetharaman, and Ziv Bar-Joseph. 2008. Protein complex identification by supervised graph local clustering. In *ISMB*. 250–268.
- [65] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. 2004. Defining and identifying communities in networks. *Proc. Natl. Acad. Sci.* 101, 9 (2004), 2658–2663.
- [66] Barna Saha, Allison Hoch, Samir Khuller, Louiqa Raschid, and Xiao-Ning Zhang. 2010. Dense Subgraphs with Restrictions and Applications to Gene Annotation Graphs. In *RECOMB*, Vol. 6044. 456–472.
- [67] Saurabh Sawlani and Junxing Wang. 2020. Near-optimal fully dynamic densest subgraph. In *STOC*. 181–193.
- [68] Stephen B Seidman and Brian L Foster. 1978. A graph-theoretic generalization of the clique concept. *J. Math. Sociol.* 6, 1 (1978), 139–154.
- [69] Jianbo Shi and Jitendra Malik. 2000. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 22, 8 (2000), 888–905.
- [70] Qing Sima, Jianke Yu, Xiaoyang Wang, Wenjie Zhang, Ying Zhang, and Xuemin Lin. 2025. Deep Overlapping Community Search via Subspace Embedding. *Proc. ACM Manag. Data* 3, 1 (2025), 28:1–28:26.
- [71] Mauro Sozio and Aristides Gionis. 2010. The community-search problem and how to plan a successful cocktail party. In *KDD*. 939–948.
- [72] Longxu Sun and Xin Huang. 2025. Community Search: A Survey of the State-of-the-Art from Algorithms to Learning, Complex Graphs, and Interaction. *IEEE Data Eng. Bull.* 49, 2 (2025), 42–57.

- [73] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific reports* 9, 1 (2019), 1–12.
- [74] Jianwei Wang, Kai Wang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. 2024. Efficient Unsupervised Community Search with Pre-trained Graph Transformer. *Proc. VLDB Endow.* 17, 9 (2024), 2227–2240.
- [75] Kai Wang, Wenjie Zhang, Xuemin Lin, Ying Zhang, Lu Qin, and Yuting Zhang. 2021. Efficient and Effective Community Search on Large-scale Bipartite Graphs. In *ICDE*. 85–96.
- [76] Yuxiang Wang, Xiaoxuan Gou, Xiaoliang Xu, Yuxia Geng, Xiangyu Ke, Tianxing Wu, Zhiyuan Yu, Runhuai Chen, and Xiangying Wu. 2024. Scalable Community Search over Large-scale Graphs based on Graph Transformer. In *SIGIR*. 1680–1690.
- [77] Yuxiang Wang, Zhangyang Peng, Xiangyu Ke, Xiaoliang Xu, Tianxing Wu, and Yuan Gao. 2025. Cohesiveness-aware Hierarchical Compressed Index for Community Search on Attributed Graphs. *Proc. ACM Manag. Data* 3, 1 (2025), 22:1–22:27.
- [78] Yuxiang Wang, Shuzhan Ye, Xiaoliang Xu, Yuxia Geng, Zhenghe Zhao, Xiangyu Ke, and Tianxing Wu. 2024. Scalable Community Search with Accuracy Guarantee on Attributed Graphs. In *ICDE*. 2737–2750.
- [79] Yubao Wu, Ruoming Jin, Jing Li, and Xiang Zhang. 2015. Robust Local Community Detection: On Free Rider Effect and Its Elimination. *Proc. VLDB Endow.* 8, 7 (2015), 798–809.
- [80] Xiaoliang Xu, Jun Liu, Yuxiang Wang, and Xiangyu Ke. 2022. Academic Expert Finding via $(k, \mathcal{P})$ -Core based Embedding over Heterogeneous Graphs. In *ICDE*. 338–351.
- [81] Fanyi Yang, Huifang Ma, Weiwei Gao, and Zhixin Li. 2022. Community Search over Heterogeneous Information Networks via Weighting Strategy and Query Replacement. *Frontiers Comput. Sci.* 16, 4 (2022), 164345.
- [82] Jaewon Yang and Jure Leskovec. 2012. Defining and Evaluating Network Communities Based on Ground-Truth. In *ICDM*. 745–754.
- [83] Yixing Yang, Yixiang Fang, Xuemin Lin, and Wenjie Zhang. 2020. Effective and Efficient Truss Computation over Large Heterogeneous Information Networks. In *ICDE*. 901–912.
- [84] Long Yuan, Lu Qin, Wenjie Zhang, Lijun Chang, and Jianye Yang. 2018. Index-Based Densest Clique Percolation Community Search in Networks. *IEEE Trans. Knowl. Data Eng.* 30, 5 (2018), 922–935.
- [85] Wentong Zhang, Kaiyu Feng, Lanting Fang, Junghoon Kim, Kaibo Zhang, Dahee Kim, Shuliang Wang, Ye Yuan, and Guoren Wang. 2025. Budgeted Strong Community Search in Heterogeneous Graphs. *Proc. ACM Manag. Data* 3, 6 (2025), 1–27.
- [86] Yue Zhang, Yankai Chen, Yingli Zhou, Yucan Guo, Xiaolin Han, and Chenhao Ma. 2025. UTCS: Effective Unsupervised Temporal Community Search with Pre-training of Temporal Dynamics and Subgraph Knowledge. In *SIGIR*. 2647–2651.
- [87] Yuting Zhang, Kai Wang, Wenjie Zhang, Xuemin Lin, and Ying Zhang. 2021. Pareto-optimal Community Search on Large Bipartite Graphs. In *CIKM*. 2647–2656.
- [88] Yuting Zhang, Kai Wang, Wenjie Zhang, Wei Ni, and Xuemin Lin. 2024. Size-bounded Community Search over Large Bipartite Graphs. In *EDBT*. 320–331.
- [89] Yining Zhao, Sourav S. Bhowmick, Nastassja L. Fischer, and S. H. Annabel Chen. 2025. How Cohesive Are Community Search Results on Online Social Networks?: An Experimental Evaluation. In *SIGIR*. 358–369.
- [90] Keqi Zhou, Junchang Xin, Jinyi Chen, Xian Zhang, Beibei Wang, and Zhiqiong Wang. 2023. Effective and efficient community search with size constraint on bipartite graphs. *Inf. Sci.* 647 (2023), 119511.
- [91] Yingli Zhou, Yixiang Fang, Wensheng Luo, and Yunming Ye. 2023. Influential Community Search over Large Heterogeneous Information Networks. *Proc. VLDB Endow.* 16, 8 (2023), 2047–2060.
- [92] Yuhan Zhou, Zheng Wu, Qing Liu, Xin Huang, Jianliang Xu, and Yunjun Gao. 2026. To Learn or Not to Learn? An In-depth Empirical Analysis of Community Search Approaches (Technical Report). <https://github.com/ZJU-DAILY/RecCS/blob/main/RecCS-Technical-Report.pdf>.

Received 17 January 2026; revised 19 May 2026; accepted 12 June 2026