NCSAC: Effective Neural Community Search via Attribute-augmented Conductance

Anonymous Author(s)

ABSTRACT

Identifying locally dense communities closely connected to the user-initiated query node is crucial for a wide range of applications. Existing approaches either solely depend on rule-based constraints or exclusively utilize deep learning technologies to identify target communities. Therefore, an important question is proposed: can deep learning be integrated with rule-based constraints to elevate the quality of community search? In this paper, we affirmatively address this question by introducing a novel approach called Neural Community Search via Attribute-augmented Conductance, abbreviated as NCSAC. Specifically, NCSAC first proposes a novel concept of attribute-augmented conductance, which harmoniously blends the (internal and external) structural proximity and the attribute similarity. Then, NCSAC extracts a coarse candidate community of satisfactory quality using the proposed attribute-augmented conductance. Subsequently, NCSAC frames the community search as a graph optimization task, refining the candidate community through sophisticated reinforcement learning techniques, thereby producing high-quality results. Extensive experiments on six real-world graphs and ten competitors demonstrate the superiority of our solutions in terms of accuracy, efficiency, and scalability. Notably, the proposed solution outperforms state-of-the-art methods, achieving an impressive F1-score improvement ranging from 5.3% to 42.4%. For reproducibility purposes, the source code is available at https://anonymous.4open.science/r/ncsac-7642.

CCS CONCEPTS

• **Mathematics of computing → Graph algorithms**.

KEYWORDS

Community Search; Graph Neural Networks; Graph Clustering

ACM Reference Format:

Anonymous Author(s). 2018. NCSAC: Effective Neural Community Search via Attribute-augmented Conductance. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

A community is defined as a subgraph characterized by internal cohesiveness and external sparsity. Community Search (CS) aims to identify the specific community that exhibits a strong correlation with the user-initiated query nodes [18]. The importance of CS lies in its wide range of practical applications, including social e-commerce [55] and uncovering personal backgrounds [7].

Existing CS models can be generally classified into two categories (details in Section 7): rule-based CS methods [2, 3, 13, 14, 24, 27, 29, 34, 48, 49, 53] and learning-based CS methods [6, 8, 9, 17, 25, 32, 43, 45]. The former typically identifies the resultant community by utilizing predefined structural constraints (e.g., subgraph-based k-core/truss [13, 24] or optimization-based density/modularity[14, 27, 48, 53]). However, they impose cumbersome constraints on the target community, leading to structural inflexibility. For instance, some nodes within the ground-truth community (such as boundary nodes) may struggle to meet these constraints, causing users to question the model's accuracy [8]. Fortunately, learning-based CS methods can relieve the inflexibility of rule-based CS methods. In particular, one popular methodology is to define the CS problem as a node classification task [8, 17, 25, 32, 43]. These methods employ a two-stage framework. Initially, the model is trained using semi-supervised or unsupervised techniques to generate node embedding vectors. Subsequently, these embedding vectors are utilized to evaluate the probability of each vertex belonging to the target community. Another methodology is to frame the CS problem as a generative task, exemplified by CommunityAF [9]. Specifically, it first utilizes the Graph Neural Networks (GNNs) [28] to derive node representations and subsequently employs autoregressive flow to generate the community from the query node. In contrast to node classification models, generative models offer greater flexibility and eliminate concerns regarding community connectivity [9].

However, existing methods rely either entirely on rule-based constraints or solely leverage deep learning techniques for identifying target communities. Thus, a significant inquiry arises: how can deep learning be combined with traditional rule-based constraints to enhance the accuracy of CS? In this paper, we answer this question affirmatively. Two challenges are involved:

Challenge 1: How to identify a coarse candidate community of satisfactory quality? The most straightforward method is to use the traditional rule-based CS models. However, this technique exhibits significant limitations. First, they only consider the internal cohesiveness and ignore the external sparsity, which is an important factor in measuring the quality of a community [21, 26, 31], resulting in unsatisfactory quality. Second, they only optimize the topology and ignore important node attributes, further limiting their performance. Thus, our first challenge is to identify a candidate community that accounts for both the (internal and external) structural proximity and attribute similarity.

Challenge 2: How to refine the candidate community? After identifying the candidate community, it still lacks flexibility and robustness, as it remains a predefined subgraph. Recently, *ALICE* [44] and *TransZero* [43] have used GNNs to refine the candidate community. However, they treat the candidate community as a new graph G_{new} and apply GNNs on G_{new} for node classification tasks. As a result, their methods are heavily dependent on the performance of G_{new}, since nodes outside G_{new} have no chance to contribute to the target community, resulting in poor quality (Section 6). Therefore, refining these outside and inside nodes of G_{new} to enhance the candidate community becomes our second challenge.

Our Contributions. To address these challenges, we propose a novel neural community search model via attribute-augmented conductance (named *NCSAC*) which consists of two principal components: the candidate community extraction module and the neural community refinement module. The former leverages the newly-proposed *attributed-augmented conductance* to identify a candidate

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/10.1145/1122445.1122456

community of satisfactory quality. Following this, the community refinement module non-trivially utilizes powerful reinforcement learning [22, 37, 38] to iteratively add and remove nodes starting from the candidate community, thereby refining the quality of the candidate community. More concretely, to address Challenge 1, we resort to the well-known conductance [3, 29, 31, 51] to capture the internal cohesion and external sparsity. However, the traditional conductance only models the structural information while neglecting crucial attribute information. To overcome this dilemma, we propose a novel attributed-augmented conductance metric, which seamlessly integrates the topology structure and node attributes. Therefore, within the candidate community extraction module, we initiate the process from the given query node to determine a candidate community with the (most favorable) approximate attribute-augmented conductance.

For Challenge 2, we design a neural community refinement module that frames the refinement process as a *combinatorial graph optimization* problem (Definition 5.1). The core idea is to improve the community quality by removing irrelevant nodes or appending new ones from the community boundaries, optimizing our objective through careful node selection. Unfortunately, the community refinement problem is NP-hard (Theorem 5.2), rendering direct solutions impractical. To address this, we present an efficient community refiner based on reinforcement learning. Specifically, we cast this problem as a reinforcement learning task and employ the Proximal Policy Optimization (PPO) [22, 38] technique for exploration. Besides, we design two distinct policy networks: one for predicting node additions and another for predicting node deletions. Through reinforcement learning, our approach can effectively refine the candidate community. On top of that, we introduce a termination strategy to ensure that the final community structure remains flexible and adaptable.

We conduct extensive experiments on six real-life graphs and ten competitors to evaluate the effectiveness and scalability of the proposed solutions. The empirical results show that our proposed solution can achieve an F1-score improvement ranging from 5.3% to 42.4% with comparable runtime in comparison to baselines.

2 PRELIMINARIES

2.1 Problem Definition

Consider an undirected attributed graph $G = (V, E, \mathcal{F})$, where V is a node set of cardinality n, E is an edge set of cardinality m, and $\mathcal{F} \in R^{n \times k}$ is an attribute (feature) matrix. In particular, $\mathcal{F}_{uf} = 1$ indicates that node u possesses the attribute f, while $\mathcal{F}_{uf} = 0$ signifies that node u does not contain the attribute f. We refer to $\mathcal{F}[u] \in R^{1 \times k}$ as the attribute vector of node u. Let $N(u) = \{v | (u, v) \in E\}$ be the neighbors of node u and $d(u) = |N(u)|$ be the degree of u. We use $\boldsymbol{A} \in R^{n \times n}$ to denote the adjacency matrix of G, where $\boldsymbol{A}_{uv} = 1$ if $(u, v) \in E$, and $\boldsymbol{A}_{uv} = 0$ otherwise. Besides, we use $\boldsymbol{D} \in R^{n \times n}$ to represent the degree diagonal matrix of G, in which $\boldsymbol{D}_{uu} = d(u)$. The community search is formulated as follows.

Definition 2.1 (Community Search [2, 3, 13, 14, 24, 27, 29, 34, 48, 53]). Given an undirected attribute graph $G = (V, E, \mathcal{F})$, a score function $g(.)$ (e.g., minimum degree or conductance), and a query node $q \in V$, the goal of community search is to identify a vertex set $C_q \subseteq V$ such that (1) $q \in C_q$; (2) C_q is connected; (3) $g(C_q)$ is optimal.

In this paper, we conceptualize community search as a *graph optimization problem* (Definition 5.1) rather than as a binary classification task, as seen in existing approaches like *ICS-GNN* [8] and *TransZero* [43]. At a high level, we first identify a coarse yet high-quality candidate community through a powerful subgraph extraction process, laying the groundwork for subsequent refinement. Building upon the extracted candidate community, we systematically refine it by incorporating additional relevant nodes while removing extraneous ones. This iterative refinement process is designed to optimize community quality, ensuring a more accurate representation of the underlying relationships within the graph. Table 1 summarizes the state-of-the-art community search methods.

Table 1: A comparison of state-of-the-art community search methods. "SL" represents Supervised Learning, "UL" indicates Unsupervised Learning, "RL" indicates Reinforcement Learning, "NC" represents Node Classification, "GR" represents Generation, and "GO" indicates Graph Optimization.

Methodologies	Internal Cohesiveness &External Sparsity	Learning Paradigm	Problem Definition
Subgraph-based [2, 13, 24]	×	×	N/A
Density [14, 48, 53]	×	×	N/A
Modularity [27]	×	×	N/A
Conductance [3, 29, 34]	✓	×	N/A
ICS-GNN [8]	×	SL	NC
QD-GNN [25]	×	SL	NC
COCLEP [32]	×	Semi-SL	NC
CGNP [17]	×	Semi-SL	NC
TransZero [43]	×	UL	NC
ALICE [44]	×	SL	NC
CommunityAF [9]	×	SL	GR
NCSAC (This paper)	✓	RL	GO

Remark. Attribute community search (i.e., both attributes and nodes serve as query requests, e.g., *AQD-GNN* [25]) and multiple query nodes are orthogonal to ours and fall outside the scope of this paper, presenting potential directions for future research.

2.2 Reinforcement Learning

Reinforcement learning (RL for short) is used to address the challenge of enabling agents to learn strategies that maximize returns or achieve specific goals through interactions with their environment [16, 22, 30, 37, 38, 57]. Typically, the interaction between an agent and its environment is modeled using a Markov Decision Process (MDP), which can be represented by a quintuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. Here, $\mathcal{S}$ denotes the state space, while $\mathcal{A}$ represents the action space, encompassing the set of viable actions within a given environment. $\mathcal{P}$ signifies the state transition function, with $\mathcal{P}(s'|s, a)$ denoting the probability of transitioning from state s to state s' upon taking action a. $\mathcal{R}$ represents the reward function, typically a deterministic function crafted by humans. $\gamma \in [0, 1]$ stands for discount factor, which is a constant. The goal of RL is to learn a policy by maximizing the expected *discounted return* $U_t = E[\sum_{i=t}^{\infty} \gamma^{i-t} \mathcal{R}_i]$, in which $\mathcal{R}_i$ is the reward for transitioning from state s_{i-1} to s_i.

3 THE PROPOSED FRAMEWORK

In this section, we propose an effective community search method, *NCSAC*, as illustrated in Figure 1. Specifically, *NCSAC* first utilizes our proposed *attribute-augmented conductance* to adaptively extract a coarse candidate community by identifying the h-hop subgraph with optimal attribute-augmented conductance. We highlight that the attribute-augmented conductance simultaneously captures both the topological structure (i.e., internal cohesiveness and external sparsity) and the attributes of nodes. Consequently, the resulting coarse candidate community exhibits comparable community quality (Section 4). Subsequently, *NCSAC* pre-trains a graph neural network to serve as the state encoder, integrating community awareness into the state representation through our proposed contrastive loss and triplet loss. Once the coarse candidate community

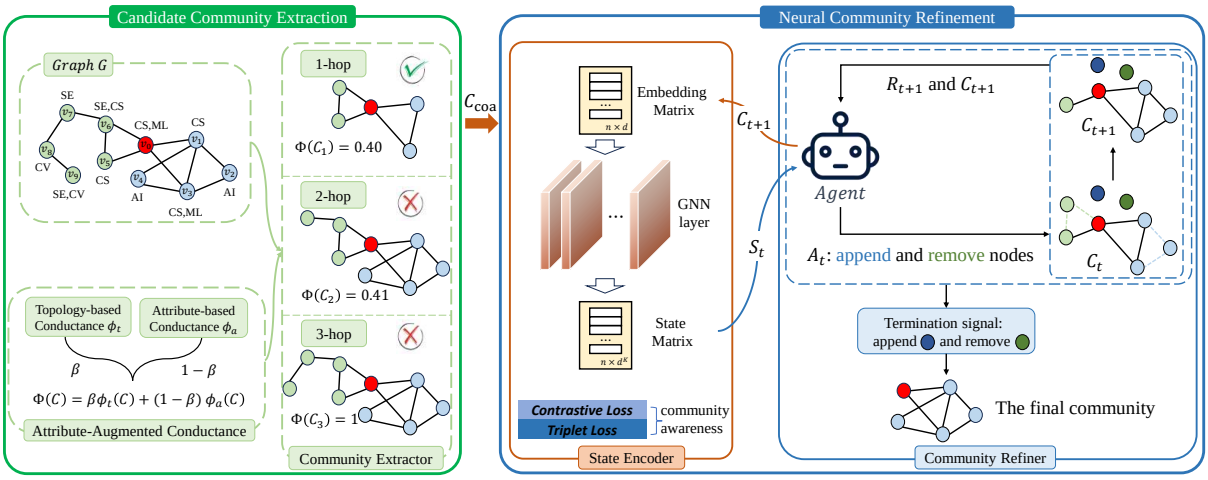


Figure 1: The framework of our proposed *NCSAC*. In the candidate community extraction stage, *NCSAC* systematically extracts a coarse candidate community C_{coa} by leveraging our novel attribute-augmented conductance (the red node is the query node). Subsequently, we pre-train a state encoder to effectively integrate community awareness information into the state encoding. Finally, we apply non-trivial reinforcement learning techniques to flexibly refine the coarse candidate community, ultimately yielding high-quality results (best view in color).

is encoded into the initial state by the state encoder, reinforcement learning is employed to initiate the community refinement process. At each step of refinement, the agent processes the community's state encoding and refines the community by incorporating promising nodes while simultaneously excluding noisy or irrelevant ones. The refined community is then re-encoded by the state encoder to generate the next state, enabling the agent to proceed with the subsequent refinement step. This iterative process continues until the predefined termination strategy is satisfied (Section 5).

4 CANDIDATE COMMUNITY EXTRACTION

This section first proposes a novel concept of attribute-augmented conductance to measure the quality of a community, which seamlessly blends structural and attribute information. Then, we devise an efficient dynamic update algorithm to adaptively identify high-quality candidate communities. **The proofs of the following lemmas and theorems are deferred to the full paper [4].**

4.1 Attribute-augmented Conductance

As previously mentioned, a crucial aspect of community search is the development of an indicator to measure community quality. To this end, we adopt the well-known free-parameter community metric, *conductance*, due to its advantageous structural properties and solid theoretical foundation [3, 21, 29, 31, 33, 34, 51]. Unlike metrics such as k-core [5], k-truss [1], k-clique [12], and modularity [10], which measure only the internal cohesion of a community, conductance takes into account both internal cohesion and external sparsity. This dual consideration makes conductance a more robust measure of community quality [26, 31].

Definition 4.1 (Conductance). Given a graph G and a vertex set C, the conductance of C is defined as follows.

$$\phi(C) = \frac{\left|cut(C, \bar{C})\right|}{\min\{vol(C), vol(\bar{C})\}}. \tag{1}$$

where $\bar{C}$ denotes the complement of the community, defined as $\bar{C} = V \backslash C$. The cut between community C and its complement is represented as $cut(C, \bar{C}) = \{(u, v) \mid u \in C, v \in \bar{C}\}$, while $vol(C) = \sum_{u \in C} d(u)$ indicates the total sum of node degrees within community C. Therefore, a smaller value of $\phi(C)$ suggests that the number of edges exiting community C is relatively low compared to the number of edges contained within it. Consequently, a lower value of $\phi(C)$ indicates a higher quality of community C [3, 29, 31, 51]. A more interesting and useful probabilistic explanation of the conductance is stated as follows.

Theorem 4.2. *Given a graph G and a community C. Consider a random walk $(w_t)_{t \in \mathbb{N}}$ generated by the transition matrix $\boldsymbol{P}(= \boldsymbol{D}^{-1}\boldsymbol{A})$, and the initial state w_0 is in C (or $\bar{C}$) and is randomly chosen following the degree distribution. We have*

$$\phi(C) = max\{Pr(w_1 \in \bar{C} | w_0 \in C), Pr(w_1 \in C | w_0 \in \bar{C})\}. \tag{2}$$

Theorem 4.2 offers a random walk interpretation of conductance. Specifically, the conductance of the community C is the probability that a random walker, beginning within C (or $\bar{C}$), will escape to the outside by one-hop random walk. To distinguish it from the attribute-based conductance discussed later, we redefine conductance as topology-based conductance, denoted by $\phi_t(C)$. However, $\phi_t(C)$ falls short in this context, as it fails to consider the attributes associated with the nodes. Inspired by the random walk interpretation of conductance, we propose the attribute-based conductance to capture structural proximity and attribute similarity.

Definition 4.3 (Attribute Edge). Given an attributed graph $G = (V, E, \mathcal{F})$, the connection between nodes u and v through a specific attribute f is considered an *attribute edge*, denoted as (u, v, f). Namely, (u, v, f) indicates nodes u and v share attribute f (i.e., , $\mathcal{F}_{uf}$=1 and $\mathcal{F}_{vf}$=1).

Definition 4.4 (Attribute Degree). Given an attributed graph $G = (V, E, \mathcal{F})$, the attribute degree of a node u, denoted as $d_a(u)$, is defined as the number of attribute edges connected to u. The formula for its calculation is presented as follows:

$$d_a(u) = \sum_{v \in V \backslash \{u\}} \mathcal{F}[u] \cdot \mathcal{F}[v]^\top. \tag{3}$$

Definitions 4.3 and 4.4 map the original attribute graph into a multigraph (i.e., two vertices can be connected by more than

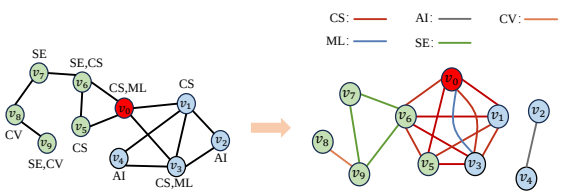


Figure 2: From the original attribute graph to the multigraph.

one edge), as depicted in Figure 2. In this multigraph, the edges correspond to attribute edges, and the degree of each node reflects its attribute degree. Unfortunately, the multigraph may be extremely dense (even close to quadratic), especially if there is an attribute value shared by a decent portion, e..g, 50%, of the nodes in the graph. Section 4.2 will show how to quickly obtain attribute-augmented conductance without materializing the multigraph.

Definition 4.5 (Attribute-based Random Walk). Given an attribute graph G, the attribute-based transition probability from node u to node v is defined as follows:

$$P^a_{uv} = \frac{\mathcal{F}[u] \cdot \mathcal{F}[v]^\top}{d_a(u)}. \tag{4}$$

Intuitively, P^a_{uv} models the transition probability from u to v based on the attribute set of G.

Example 4.6. Consider the example of nodes v_0 and v_3 in Figure 2. v_0 is connected to v_3 through two attribute edges, specifically (v_0, v_3, CS) and (v_0, v_3, ML). Besides, v_0 also has three other attribute edges, namely, (v_0, v_1, CS), (v_0, v_5, CS), and (v_0, v_6, CS). Thus, the attribute degree of node v_0 equals 5 and the attribute transition probability from v_0 to v_3 is $P^a_{v_0 v_3} = \frac{2}{5} = 0.4$.

Definition 4.7 (Attribute-based Conductance). Given an attributed graph G and a community C, the attribute-based conductance, denoted as $\phi_a(C)$, is defined as follows:

$$\phi_a(C) = \frac{\left|cut_a(C, \bar{C})\right|}{\min\{vol_a(C), vol_a(\bar{C})\}}. \tag{5}$$

Here, $cut_a(C, \bar{C}) = \{(v_i, v_j, f) \mid v_i \in C, v_j \in \bar{C}\}$ denotes the number of attribute edges connecting community C and $\bar{C}$, while $vol_a(C) = \sum_{u \in C} d_a(u)$ represents the total sum of attribute degrees within C.

We highlight that while the definitions of attribute-based conductance $\phi_a(C)$ and topology-based conductance $\phi_t(C)$ are similar, computing $\phi_a(C)$ is significantly more complex (details in Section 4.2). Besides, analogous to Theorem 4.2, we can also understand attribute-based conductance from the perspective of attribute-based random walk, thereby enhancing its practical applicability. Specifically, we reformulate Eq. (5) as $\phi_a(C) = max\{Pr(\omega_1 \in \bar{C} | \omega_0 \in C), Pr(\omega_1 \in C | \omega_0 \in \bar{C})\}$, where $(\omega_t)_{t \in \mathbb{N}}$ represents a random walk generated by the attribute-based transition matrix P^a (i.e., Eq. 4). Therefore, a smaller $\phi_a(C)$ indicates a lower probability of transitioning from community C (or $\bar{C}$) to $\bar{C}$ (or C) through the one-hop attribute-based random walk. Intuitively, this suggests that the nodes within the community largely share similar attributes, while there is limited overlap with the attributes of nodes outside the community. This observation aligns with our principles of internal attribute cohesiveness and external attribute sparsity.

Building on the concepts discussed earlier, we now introduce the definition of attribute-augmented conductance, which can capture simultaneously the topology and the attributes of the community.

Definition 4.8 (Attribute-augmented Conductance). Given an attributed graph G, a community C, and a preference parameter β, the attribute-augmented conductance is as follows:

$$\Phi(C) = \beta \cdot \phi_t(C) + (1 - \beta) \cdot \phi_a(C). \tag{6}$$

Remark. $\Phi(C)$ is constructed as a combination of $\phi_t(C)$ and $\phi_a(C)$ weighted by preference parameter β. This formulation captures a trade-off between topological structure and node attributes, allowing us to identify communities with distinct characteristics by varying β. In particular, when $\beta > 0.5$, the community emphasizes the interaction edges among vertices. Conversely, when $\beta < 0.5$, the focus shifts toward the attribute similarity among vertices.

4.2 Adaptive Community Extractor

Local search [12, 13, 23, 27, 52] is a well-established technique for identifying the community to which a query node belongs, utilizing an iterative strategy that incrementally adds nodes in a designed ordering. We emphasize that directly obtaining the optimal solution through this ordering is generally NP-hard [13], and it becomes even more complex when considering conductance [34], ultimately resulting in poor scalability. Fortunately, our principal aim is to efficiently derive a coarse candidate community with satisfactory quality, which will subsequently be refined through our proposed novel and non-trivial reinforcement learning techniques. To this end, we let $N^h(q) = \{v | dist(q, v) \leq h\}$ be the h-hop neighbors of the query node q, where $dist(q, v)$ represents the shortest-path distance between nodes q and v. We designate $N^h(q)$ as the candidate community, which has been well-supported by existing literature [12, 23, 27, 52]. Intuitively, nodes that are closer to the query node are more likely to belong to the target community. This approach implicitly defines the order of node addition (i.e., by adding based on distance in batches).

Example 4.9. In Figure 1, we use v_0 as the query node, representing 1-hop, 2-hop, and 3-hop subgraphs as C_1, C_2, and C_3, respectively. We first compute the topology-based conductance values, obtaining $\phi_t(C_1) = 0.55$, $\phi_t(C_2) = 0.33$, and $\phi_t(C_3) = 0.55$. From these values, we select C_2 as the coarse community by only using the topology-based conductance. Next, when incorporating attributes, the attribute-based conductance values are calculated as $\phi_a(C_1) = 0.25$, $\phi_a(C_2) = 0.5$, and $\phi_a(C_3) = 1$. By setting $\beta = 0.5$, indicating equal weighting of both topological structure and node attribute, the attribute-augmented conductance values are adjusted to $\Phi(C_1) = 0.40$, $\Phi(C_2) = 0.415$, and $\Phi(C_3) = 0.775$. In this case, C_1 is selected as the coarse community, as it better aligns with the community definition compared to C_2.

A failed attempt. According to Definition 4.8, a naive approach to calculating attribute-augmented conductance is first to construct a multigraph (Figure 2). Subsequently, it uses Eq. 5 to obtain the attribute-based conductance and then applies Eq. 6 to derive the attribute-augmented conductance. However, this naive solution has two-fold drawbacks: (1) Materializing the multigraph requires prohibitively high time and space complexities. In particular, it takes $O(kn^2)$ time and $O(kn^2)$ space to materialize the multigraph by Eq. 3, resulting in poor scalability. Where n and k are the number of nodes and attributes of G, respectively. (2) When the subgraph $N^h(q)$ extends to $N^{h+1}(q)$, calculating the attribute-augmented conductance of $N^{h+1}(q)$ from scratch entails significant redundant computations. Consequently, the naive method cannot deal with massive graphs with millions of edges (details in Section 6.3). Thus, an important challenge is quickly calculating and updating attribute-augmented conductance without materializing the multigraph.

Algorithm 1 *Adaptive Community Extractor*

Input: An attributed graph $G = (V, E, \mathcal{F})$, a query node q, and a parameter β.
Output: The coarse candidate community C.

1: $d_a \leftarrow \mathcal{F} \cdot (\mathbf{1}_n \mathcal{F} - \mathbf{1}_k)^\top$; $vol_a(V) \leftarrow \sum_{v \in V} d_a(v)$
2: $cut \leftarrow d(q)$; $vol \leftarrow d(q)$; $cut_a \leftarrow d_a(q)$; $vol_a \leftarrow d_a(q)$
3: $C \leftarrow \{q\}$; $tmp \leftarrow \{q\}$; $\widehat{\Phi} \leftarrow 1$; $att \leftarrow \mathcal{F}[q]$
4: **for** $h = 1$ to $\max\{dist(q,u) | u \in G\}$ **do**
5: $\quad \Delta(h) \leftarrow N^h(q) - N^{h-1}(q)$
6: $\quad$ **for** each $u \in \Delta(h)$ **do**
7: $\quad\quad tmp.add(u)$
8: $\quad\quad cut \leftarrow cut + d(u) - 2 \cdot |C \cap N(u)|$, $vol \leftarrow vol + d(u)$
9: $\quad\quad cut_a \leftarrow cut_a + d_a(u) - 2 \cdot \mathcal{F}[u] \cdot att^T$, $vol_a \leftarrow vol_a + d_a(u)$
10: $\quad\quad att \leftarrow att + \mathcal{F}[u]$
11: $\quad$ **if** $\frac{\beta \cdot cut}{\min\{vol, 2m - vol\}} + \frac{(1-\beta) \cdot cut_a}{\min\{vol_a, vol_a(V) - vol_a\}} < \widehat{\Phi}$ **then**
12: $\quad\quad \widehat{\Phi} \leftarrow \frac{\beta \cdot cut}{\min\{vol, 2m - vol\}} + \frac{(1-\beta) \cdot cut_a}{\min\{vol_a, vol_a(V) - vol_a\}}$
13: $\quad\quad C \leftarrow tmp$
14: **return** C

Several in-depth observations. Analysis of Eq. 6 reveals that the main bottlenecks in calculating and updating attribute-augmented conductance lie in the efficient computation of $vol_a(C)$ and the rapid update of $|cut_a(C, \bar{C})|$. To overcome this challenge, we observe that a simple matrix-vector product can obtain the attribute degrees of all nodes. Besides, we can also quickly and dynamically update $|cut_a(C, \bar{C})|$ by simply maintaining several arrays. For convenience, we state these observations as the following lemmas.

Lemma 4.10. *Consider an attributed graph $G = (V, E, \mathcal{F})$, for any node $u \in V$, the attribute degree of u can be reformatted as $d_a(u) = \mathcal{F}[u] \cdot (\mathbf{1}_n \mathcal{F} - \mathbf{1}_k)^\top$, in which $\mathbf{1}_n \in R^{1 \times n}$ (or $\mathbf{1}_k \in R^{1 \times k}$) is a vector with all elements equal to 1. Besides, we have $d_a = \mathcal{F} \cdot (\mathbf{1}_n \mathcal{F} - \mathbf{1}_k)^\top$.*

Lemma 4.11. *For a community C and any $u \in \bar{C}$, we have $cut(C \cup \{u\}, V \setminus (C \cup \{u\})) = |cut(C, \bar{C})| + d(u) - 2\,|cut(u, C)|$ and $cut_a(C \cup \{u\}, V \setminus (C \cup \{u\})) = |cut_a(C, \bar{C})| + d_a(u) - 2\,|cut_a(u, C)|$.*

Given a community C and any $u \in \bar{C}$, based on Lemma 4.11, we have the following facts.

$$\phi_t(C \cup \{u\}) = \frac{|cut(C, \bar{C})| + d(u) - 2\,|cut(u, C)|}{\min\{vol(C) + d(u), 2m - vol(C) - d(u)\}}. \quad (7)$$

$$\phi_a(C \cup \{u\}) = \frac{|cut_a(C, \bar{C})| + d_a(u) - 2|cut_a(u, C)|}{\min\{vol_a(C) + d_a(u), vol_a(V) - vol_a(C) - d_a(u)\}}. \quad (8)$$

By Fact 4.2, we can know that the challenge with incremental computation of ϕ_t (or ϕ_a) is how to efficiently maintain $|cut(u, C)|$ (or $|cut_a(u, C)|$). Note that $|cut(u, C)| = |N(u) \cap C|$ is easily obtained in $O(d(u))$ time. However, calculating $|cut_a(u, C)|$ is more complicated. This is due to the fact that $|cut_a(u, C)| = |\{(u, v, f) \mid v \in C\}| = \sum_{v \in C} \mathcal{F}[u]\mathcal{F}[v]^\top$ as defined in Definition 4.7, which requires $O(k|C|)$ time to compute. To improve efficiency, we devise several non-trivial data structures to efficiently update ϕ_t and ϕ_a.

Implementation details. Algorithm 1 provides the pseudo-code of our adaptive community extractor. Specifically, in Lines 1-3, it uses Lemma 4.10 to obtain the attribute degree vector d_a and initializes the candidate community C, current search space tmp, and internal attribute att. Then, by Lemma 4.11, it executes the incremental dynamic update process for cut, vol, cut_a, and vol_a (Lines 5-10). Lines 11-13 update the candidate community C and the optimal value $\widehat{\Phi}$. Finally, Line 14 returns C as the candidate community.

Example 4.12. Reconsider Figure 2, Figure 3 illustrates the process of extending from the query node v_0 to $N^1(v_0)$. To demonstrate

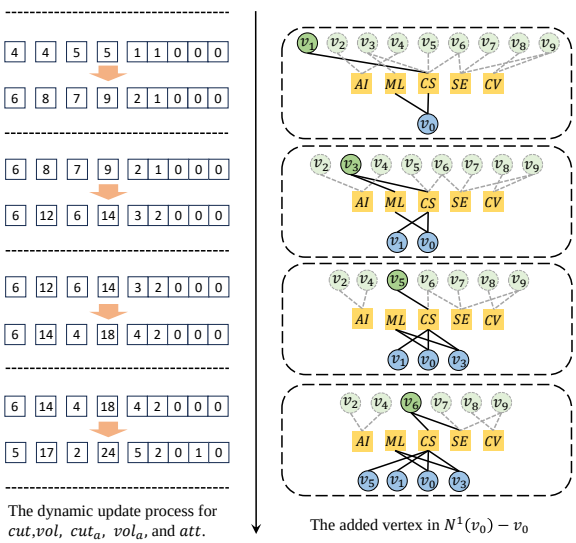


Figure 3: Illustration of Example 4.12 for Algorithm 1. The left side shows the update process that occurs after each node is added. On the right side, the update of cut_a is elaborated using a node-attribute bipartite graph. The dark blue nodes indicate the current search space tmp, while the dark green nodes represent the nodes to be added (best view in color).

the dynamic update algorithm, we focus on the addition of node v_3 (i.e., the second step). Initially, the community C consists of nodes v_0 and v_1. The neighbors of node v_3 include v_0, v_1, v_2, and v_4, with two edges connecting v_3 to community C. Consequently, $|cut(v_3, C)| = 2$. Then cut is updated to $cut = 6 + 4 - 2 \times 2 = 6$, while $vol = 8 + 4 = 12$. At this juncture, the internal attribute vector att is $[2, 1, 0, 0, 0]$, signifying that two edges connecting to the CS attribute and one edge connecting to the ML attribute are associated with community C. Therefore, $|cut_a(v_3, C)| = \mathcal{F}[v_3] \cdot att^T = [1, 1, 0, 0, 0] \cdot [2, 1, 0, 0, 0]^T = 3$, representing the three attribute edges through which node v_3 is linked to community C via its own attributes (i.e., CS and ML). The updated cut_a then becomes $7 + 5 - 2 \times 3 = 6$, and $vol_a = 9 + 5 = 14$. Following the addition of node v_3, att is updated to $[1, 1, 0, 0, 0] + [2, 1, 0, 0, 0] = [3, 2, 0, 0, 0]$. This process continues iteratively, adding nodes sequentially until all nodes (i.e. $\Delta(h)$) are added.

Theorem 4.13. *The time complexity and space complexity of Algorithm 1 are $O(m + nk)$ and $O(m + n + k)$, respectively. k ($k << n$) is the attribute dimension.*

5 NEURAL COMMUNITY REFINEMENT

Section 4 prioritizes scalability over community quality, necessitating further refinement of the coarse candidate community. Therefore, we develop a Neural Community Refinement method, structured into two primary components. In the State Encoder component, we guide the pre-training model using two loss functions that integrate community-aware node representations as state inputs for reinforcement learning. In the Community Refiner component, we employ reinforcement learning to balance exploration and exploitation, thereby identifying nodes to be appended or removed. This process ultimately yields high-quality communities.

5.1 Community-aware State Encoder

In this component, our objective is to incorporate community awareness into the embedded space of nodes. To this end, we design two

loss functions to achieve a strong correlation between nodes and communities: *Contrastive Loss* and *Triplet Loss*. Intuitively, an edge between two nodes suggests a high probability that they belong to the same community, and the absence of such an edge indicates otherwise. Thus, we model this relationship using contrastive loss:

$$L_C = \sum_{u,v \in V} A_{uv}\delta(h_u, h_v) + (1 - A_{uv})max(0, \gamma_1 - \delta(h_u, h_v)). \quad (9)$$

$\delta(\cdot)$ represents the distance metric, for which we utilize cosine distance. γ_1 is the margin hyperparameter that indicate the minimum distance. In the context of the community search task, when provided with a seed node, the remaining nodes can be classified into two distinct groups: those within the same community and those in different communities. To effectively capture this differentiation, we implement triplet loss:

$$L_T = \sum_{(q,q_+,q_-)} max\left\{\delta\left(h_q, h_{q_+}\right) - \delta\left(h_q, h_{q_-}\right) + \gamma_2, 0\right\}. \quad (10)$$

where q, q_+, and q_- denote the query node, the positive sample (i.e., q_+ and q are in the same community), and the negative sample (i.e., q_- and q are in different communities), respectively. γ_2 is the margin hyperparameter that indicates the minimum distance. We concurrently account for both of the aforementioned loss functions. As a consequence, the aggregate loss is formally defined as follows:

$$L = L_T + \alpha L_C. \quad (11)$$

Without loss of generality, in this paper, we utilize Graph Convolutional Network (GCN [28]) as our encoder in this component, defined as follows:

$$h_u^{l+1} = Dr(\sigma(h_u^l W_s^{l+1} + \sum_{v \in N(u)} \frac{h_v^l W^{l+1}}{\sqrt{(d(u)+1)(d(v)+1)}} + b^{l+1})). \quad (12)$$

Here, W_s^{l+1}, W^{l+1} and b^{l+1} represents the trainable weights, and $\sigma(\cdot)$ denotes a nonlinear activation function, e.g., ReLU. The parameter $Dr(\cdot)$ refers to the dropout rate, which is used to prevent overfitting. In this paper, we employ a two-layer GCN architecture.

5.2 RL-based Community Refiner

As previously indicated, after acquiring the candidate community, we can redefine the community search problem as a community refinement graph optimization task.

Definition 5.1 (Community Refinement Graph Optimization). Given a graph G and a candidate coarse community C_{coa}, along with a score function $\Upsilon(.)$, the goal of community refinement is to identify a community C_{opt} with the optimal $\Upsilon(C_{opt})$ by appending or removing nodes for C_{coa}.

Theorem 5.2. *The community refinement problem is NP-hard.*

There are two primary motivations for employing reinforcement learning (RL for short) in our approach. First, since the community refinement problem is NP-hard (Theorem 5.2), rendering direct solutions impractical. Second, our goal is to achieve adaptive node selection during the refinement stage, eliminating the need to establish a predetermined threshold for nodes or fix the resulting community size (details in Section 6.2). Fortunately, reinforcement learning naturally has the advantage of solving the above two issues [22, 37, 38]. Thus, we can formalize the community refiner component as a Markov Decision Process (MDP), characterized by the tuple $<\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma>$ (Section 2.2), which is stated as follows:

• **State.** During the refinement process, we employ the trained state encoder to encode both the community and its neighbors, which subsequently serves as the input for the state representation. We define the initial state $\mathcal{S}_1$ as follows:

$$\mathcal{S}_1(u) = f_s(\mathcal{F}[u], \mathbb{I}\{u \in C_{coa}\}), \text{where } u \in C_{coa} \cup \partial C_{coa}. \quad (13)$$

Here, $\partial C_{coa} = \{v \in V \setminus C_{coa} | \exists u \in C_{coa}, (u, v) \in E\}$ is the boundary of C_{coa}. $\mathbb{I}$ represents the indicator vector, where it takes the value of 1 when node u is present in the community C_{coa} and 0 otherwise.

• **Action.** At each step of this process, we consider the addition of a node from the current state, particularly from the neighbors of the intermediate community, while concurrently removing a node from the community. Ideally, these two operations should operate independently. To achieve this, we design two distinct prediction networks tailored to these different strategies. We utilize multilayer perceptrons (MLP) as the policy networks, which decode the state vector into scalar score values. Specifically, at step t, the two policy networks, f_ϕ and f_φ, evaluate the action spaces of the two strategies by assigning scores, respectively:

$$Score(u) = \begin{cases} f_\phi(\mathcal{S}_t(u)), & u \in \partial C_t \\ f_\varphi(\mathcal{S}_t(u)), & u \in C_t \end{cases} \quad (14)$$

∂C_t and C_t denote the respective action spaces for node addition and node removal. Separate policy networks are utilized to independently predict the action scores within each of these spaces.

• **State Transition.** Upon obtaining the scores, we identify the node with the highest score from ∂C_t to be appended into the community, while concurrently removing the node with the lowest score from C_t. This procedure yields the new community C_{t+1}. We formulate the state transition as follows:

$$\mathcal{S}_{t+1}(u) = f_s(\mathcal{S}_t(u), \mathbb{I}\{u \in C_{t+1}\}),\ u \in C_{t+1} \cup \partial C_{t+1}. \quad (15)$$

• **Reward.** Our main goal is to maximize the score value of the community (Definition 2.1). To achieve this, we define the reward function as the variation in the score value of the target community(i.e., *Obj*$(\cdot)$) [16, 35, 39, 46, 47]. More specifically,

$$\mathcal{R}_t = Obj\left(C_{t+1}\right) - Obj\left(C_t\right). \quad (16)$$

The reward function $\mathcal{R}_t$ described above introduces a substantial enhancement to the objective function. This formulation strategically guides the model to prioritize improvements in community value, particularly within regions exhibiting high objective function values, thereby facilitating convergence toward the global optimum. Furthermore, the proposed reward mechanism enhances the flexibility of our framework, rendering it adaptable to any objective function. For networks with available ground-truth labels, established evaluation metrics such as ARI and F1 can be employed. Conversely, for real-world networks without labels, the framework can be guided by user-specified objectives, such as the proposed attribute-augmented conductance, thereby facilitating the fine-tuning of communities to achieve superior structural and attribute-aligned configurations.

Moreover, we propose enhancing the reward mechanism to better address the characteristics of labeled networks:

$$r_{label} = \begin{cases} 1 & \text{if the appending is in } \widehat{C} \\ -1 & \text{otherwise} \end{cases} \quad (17)$$

Here, $\widehat{C}$ represents the ground-truth community. Accordingly, the reward function is revised to $\mathcal{R}_t + r_{label}$.

The described above specifically applies to the node addition process. Conversely, during the node removal process, the condition for r_{label} is modified if the removal is not in $\widehat{C}$. This adjustment ensures that the reward mechanism appropriately encourages the elimination of nodes that are not relevant to the community.

Flexible termination strategies. Existing approaches often rely on a community scoring threshold or limit the number of nodes [8, 25], are unsatisfactory due to their restrictive nature. Inspired by these, we introduce two flexible and smooth termination signals, s_a and s_r, to notify the two strategy networks, respectively. Taking the appending process as an example, we conceptualize the termination signal s_a as a virtual node, with its features being randomly generated, and incorporate it into the prediction along with the other node features. The appending process terminates when our RL model selects the s_a node. Upon termination of both processes, the resulting state is considered the final community. The flexible termination strategy is evaluated in Section 6.2.

Offline training. During the offline training phase, we initially sample a candidate coarse community from the training dataset and permit the agent to execute the complete community refinement process. Specifically, the agent begins by encoding the state of the community (Eq. (13)), and subsequently employs the policy network to decode this state into action score values (Eq. (14)). Following the addition and removal of nodes based on their respective scores, the community is updated accordingly, facilitating the subsequent update of the state encoding (Eq. (15)) and the computation of the corresponding reward (Eq. (16)). It is crucial to emphasize that, to strike a balance between exploration and exploitation during training, we adopt an ϵ-greedy action strategy. Under this strategy, there is a probability of $1-\epsilon$ that we will select the node with the highest (or lowest) score, while with a probability of ϵ, a random action is taken. The value of ϵ is linearly annealed from 1.0 to 0.05 throughout the refinement process. Upon completion of this entire procedure over τ episodes, we derive a training trajectory $\mathcal{T}$. More details are put to our full paper [4] due to the space limits.

Policy gradient method [41] is a well-known approach to strategy learning, including Trust Region Policy Optimization (TRPO) [37] and Proximal Policy Optimization (PPO) [22, 38]. In this paper, we adopt PPO as the key component of our model due to its ability to stabilize the training process by imposing constraints on the policy updates, thus preventing large, destabilizing changes. The loss function for the PPO is delineated below:

$$\mathcal{L}(\phi) = \mathbb{E}_t \left[\min \left(r_t(\phi) A_t, \text{clip}(r_t(\phi), 1-\varepsilon, 1+\varepsilon) A_t \right) \right]. \quad (18)$$

$r_t(\phi) = \frac{f_{\phi_{new}}}{f_{\phi_{old}}}$ represents the ratio of the new policy to the old policy. A_t denotes the advantage function, which is typically estimated using temporal differencing methods. The clipping function $\text{clip}(r_t(\phi), 1-\varepsilon, 1+\varepsilon)$ constrains the policy ratio $r_t(\phi)$ within the interval $[1-\varepsilon, 1+\varepsilon]$. Specifically, if $r_t(\phi)$ is less than $1-\varepsilon$, it is set to $1-\varepsilon$; if $r_t(\phi)$ exceeds $1+\epsilon$, it is set to $1+\epsilon$. This clipping mechanism is essential to the loss function, as it prevents excessively large updates to the policy during reinforcement learning training, thereby mitigating the risk of instability in the training process.

Online refinement. The online refinement commences with the agent encoding the candidate community and its neighbors into embedding vectors through the state encoder. These vectors are then decoded into node scores using the policy network. Distinct from the training phase, this stage does not involve exploratory actions; instead, the agent directly selects nodes with the highest (or lowest) scores to construct the new community. This iterative process stops until a predetermined termination criterion is met.

6 EXPERIMENTAL EVALUATION

In this section, we conduct comprehensive experiments to evaluate the effectiveness and efficiency of our solutions. All experiments are conducted on a Linux machine with an Intel Xeon(R) Silver 4210 @2.20GHz CPU and 1 TB RAM. **The scalability testing, parameter sensitivity analysis, and case studies are deferred to the full paper [4] due to the space limits.**

Table 2: Statistics of Datasets.

Dataset	$\|V\|$	$\|E\|$	$\|F\|$	$\|C\|$	$Mean(C)$	$Mean(d)$
Cora [43]	2,708	10,556	1,433	7	386.86	3.89
Facebook [50]	3,622	72,964	317	130	15.62	40.29
Cocs [43]	18,333	163,788	6,806	15	1,222.2	8.93
Twitter [50]	87,760	1,293,985	27,201	2,838	10.88	29.49
Amazon [50]	13,178	3,3767	658	4,517	9.31	5.12
Youtube [50]	216,544	1,393,206	2,165	2,865	7.67	12.86

6.1 Experimental Setup

Datasets. Our solutions have been rigorously evaluated on six publicly available datasets (Table 2), which serve as well-established benchmarks for the community search problem [2, 8, 9, 12, 13, 17, 23, 25, 27, 32, 43, 52]. In these datasets, $|F|$ represents the number of attributes, $|C|$ denotes the number of communities, $Mean(C)$ indicates the average size of the communities, and $Mean(d)$ reflects the average degree of the nodes. We classify the ground-truth communities into three categories: training, validation, and testing communities. The training communities are utilized for model training, the validation communities for fine-tuning parameters, and the testing communities for assessing the model's performance. Consistent with previous research [8, 9, 17, 25, 32, 43], we also randomly allocate these categories in 5:1:4.

Algorithms. The following ten SOTA baselines are compared. (1) Traditional rule-based community search methods: CTC [24] based on k-truss, CST [13] based on k-core, MkECS [2] based on k edge connected component, and conductance [3, 29, 34]. Note that we do not include density [14, 48, 53] and modularity [27] in the experiments because they ignore the external sparsity and are outperformed by conductance [21, 26]. (2) Learning-based methods: ICS-GNN [8], QD-GNN [25], COCLEP [32] CGNP [17], CommunityAF [9] TransZero [43]. NSCAC is our proposed algorithm. There is no comparison with *ALICE* [44], as the authors did not provide the source code, and reproducing the results is highly challenging. However, we discuss its drawbacks in Sections 1 and 7.

Metrics. Several well-established metrics are used to evaluate the quality of the identified communities [27, 32, 43]: the F1-score [36], Normalized Mutual Information (NMI) [15], and Jaccard similarity (JAC) [56]. Higher values for the F1-score, NMI, and JAC signify better performance. We also report the runtime to test the efficiency. Unless otherwise specified, for enhanced reliability, we select one vertex at random from each ground-truth community to serve as the query vertex and report the average runtime and quality.

Parameter Settings. Unless specified otherwise, we take the default parameters of the ten baselines in our experiments. On the other hand, our model NCSAC involves three main parameters: β (the preference parameter in Eq. 6), α (the loss function parameter in Eq. 11), and τ (the number of episodes in offline training of Section 5.2). We set $\beta = 0.2$, $\alpha = 0.5$, $\tau = 2000$ by default. Besides, we also report these parameter analyses in our full paper [4].

6.2 Effectiveness Evaluation

Comparison of our algorithm NCSAC with baselines. Figure 4 shows a comprehensive box plot illustration, depicting the F1-score achieved by different algorithms on all datasets. Since NMI and JAC have similar results, we defer them to the full paper [4] for brevity. We have the following observations: (1) Our model *NCSAC* achieves the best median scores on five of the six

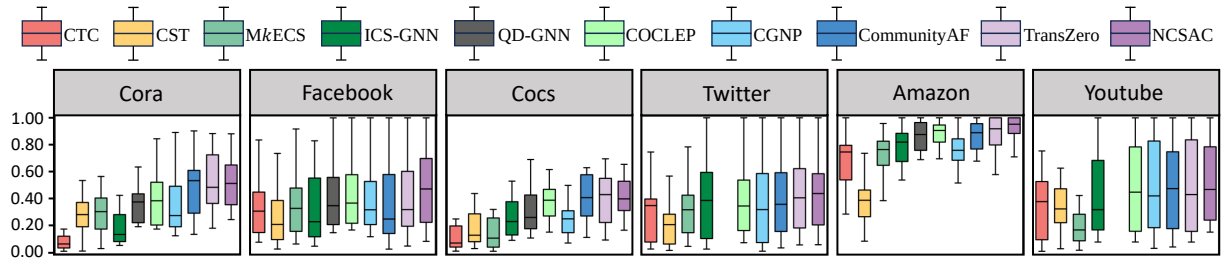


Figure 4: Comparison of our algorithm NCSAC with baselines. Since QD-GNN runs out of memory on Twitter and YouTube, we only report their results on the four remaining datasets (best view in color).

Table 3: Comparison of the traditional conductance and the attribute-augmented conductance.

	Traditional Conductance			Attribute-augmented Conductance		
	F1-score	NMI	JAC	F1-score	NMI	JAC
Cora	0.2195	0.0712	0.1348	0.3721	0.1281	0.2486
Facebook	0.3059	0.2403	0.2191	0.4145	0.3171	0.2938
Cocs	0.2562	0.0858	0.1522	0.3207	0.1580	0.2121
Twitter	0.2292	0.1968	0.1428	0.3647	0.3033	0.2469
Amazon	0.8666	0.8139	0.7752	0.8892	0.8381	0.8038
Youtube	0.2664	0.2380	0.1988	0.3995	0.3511	0.2882
Average +/−	-	-	-	+0.1028	+0.0749	+0.0784

datasets (on Cocs, *NCSAC* is slightly worse than TransZero), with F1-score improvement of 5.3%∼42.4%. For example, on Facebook, the F1-score of *NCSAC* is 0.47, whereas the runner-up is 0.33, representing an improvement of 42.4% in the F1-score. This is because our model integrates the strengths of both rule-based and learning-based methodologies. However, baseline methods study rule-based approaches and learning-based approaches independently, resulting in suboptimal community quality. (2) Our proposed *NCSAC* consistently outperforms all baselines in most cases, except for being slightly less effective than QD-GNN and COCLEP on Facebook, in terms of the quality lower bound. This outcome is attributed to *NCSAC*'s robust two-stage design. Specifically, baseline methods typically add nodes sequentially from a query node based on predefined strategies, which can be susceptible to noise, resulting in early termination and suboptimal community quality. Essentially, some nodes that could potentially improve the community score are excluded due to the limitations of these models. In contrast, *NCSAC* initially utilizes a novel attribute-augmented conductance method to identify a satisfactory and interpretable candidate coarse community. This community is then refined using reinforcement learning, ensuring a more consistent and superior performance. (3) *NCSAC* exhibits stability that is on par with (indicated by a narrower box width) other learning-based methods. Besides, learning-based models are consistently better than traditional rule-based models (e.g., CTC, CST, and MKECS) across all datasets. This can be attributed to the fact that learning-based models possess the flexibility to capture both structural proximity and attribute similarity within communities, a feat that traditional rule-based models are unable to achieve. In short, these results give clear evidence that our model can find higher-quality communities when contrasted with baselines.

Comparison of the traditional conductance and the attribute-augmented conductance. Table 3 offers a comparative analysis between traditional conductance and the newly introduced attribute-augmented conductance. To isolate the evaluation to the community extraction process alone, we report the quality of the candidate coarse community without considering the subsequent refinement stage. As can be seen, the results demonstrate that our proposed attribute-augmented conductance consistently identifies more accurate communities compared to traditional conductance. Remarkably, attribute-augmented conductance achieves an average improvement of 0.1028 in F1-score, 0.0749 in NMI, and 0.0784 in JAC across all datasets, highlighting its exceptional ability to discern community structures.

Table 4: Efficiency of adaptive community extractor.

Dataset	Cora	Facebook	Cocs	Twitter	Amazon	Youtube
NAC	0.3128	0.1174	3.8346	24.8306	0.0231	4.1999
ACE	0.02	0.0097	0.1153	0.0131	0.0003	0.0338

6.3 Efficiency Evaluation

Runtime of different algorithms. Figure 6 depicts the runtime of various algorithms across all datasets. The observations are as follows: (1) Traditional rule-based methods (namely CTC, CST, and MkECS) exhibit consistently faster execution speeds compared to learning-based approaches, albeit at the expense of significantly compromised quality, as discussed in Section 6.2. This is attributable to their reliance on predefined structural constraints for localized community search, eliminating the need for intricate learning processes. However, the inflexibility of these patterns undermines the quality of the results. (2) Our proposed model, *NCSAC*, demonstrates runtime efficiency comparable to other learning-based methods. Notably, *NCSAC* substantially surpasses alternative approaches on datasets with well-defined structures, such as Amazon and Twitter. This superiority stems from the efficacy of our community extractor, which swiftly identifies candidate community structures, thereby minimizing the necessity for extensive refinement. Conversely, in datasets with less discernible community structures, like Cora and Cocs, TransZero shows faster query times due to its simpler termination condition, which ceases the search upon the initial decline in community score. In contrast, CommunityAF employs a more elaborate termination criterion based on a sliding window approach, leading to longer query time. (3) ICS-GNN emerges as the most time-intensive method, as it necessitates retraining for each individual query, equating its query time to the training time. Consequently, ICS-GNN incurs the highest computational cost among all evaluated methods. In summary, these results give some preliminary evidence that our model is indeed efficient in practice.

Efficiency of adaptive community extractor. To evaluate the efficiency of our adaptive community extractor (i.e., *ACE* in Algorithm 1), we implement the naive method, named *NAC* (discussed as a failed attempt in Section 4). Namely, *NAC* materializes the dense multigraph (Figure 2) for computing the attribute-augmented conductance. As shown in Table 4, our proposed *ACE* is at least 12× faster than the naive method *NAC*. Specifically, *ACE* is 15×, 12×,

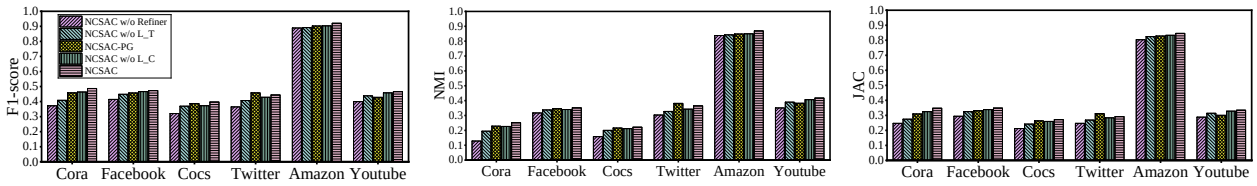


Figure 5: Ablation studies (best view in color).

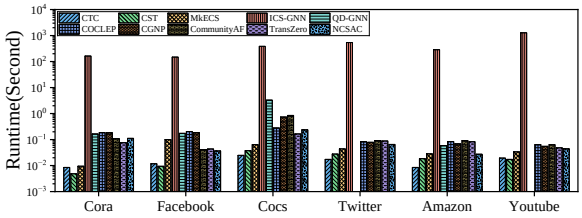


Figure 6: Runtime of different algorithms (best view in color).

33×, 1895×, 77×, 124× faster than *NAC* on Cora, Facebook, Cocs, Twitter, Amazon, and Youtube respectively. This is because our *ACE* requires only linear time complexity (Theorem 4.13), whereas the worst-case scenario for *NAC* entails a quadratic time complexity. Thus, these results align with our theoretical analysis (Section 4).

6.4 Ablation Studies

Figure 5 illustrates the ablation studies of four key components of our *NCSAC*. Specifically, NCSAC w/o Refiner is the NCSAC without the community refinement stage, NCSAC w/o L_C involves training the state encoder without incorporating the contrastive loss L_C, NCSAC w/o L_T trains the state encoder without the triplet loss L_T, NCSAC-PG substitutes the PPO algorithm with a traditional policy gradient (PG) method. Note that we only explain the F1-score because NMI and JAC have similar results. For the F1-score, we have the following conclusions: (1) *NCSAC vs. NCSAC w/o Refiner.* The community refinement module significantly improves the quality of the candidate coarse community, fulfilling its intended purpose. Specifically, the refinement module improved by at least 0.05 across all datasets, with the most significant improvement reaching up to 0.1 on the Cora. (2) *NCSAC vs. NCSAC w/o L_C & NCSAC w/o L_T.* Within the community-aware state encoder (Section 4.2), the contrastive loss and triplet loss can indeed effectively incorporate community awareness into the embedded space of nodes (Eq. 11). In particular, triplet loss L_T plays a more substantial role in enhancing state representations compared to contrastive loss L_C. For example, on Cora, the F1-score of NCSAC w/o L_T is 0.41, while that of NCSAC w/o L_C is 0.49. (3) *NCSAC vs. NCSAC-PG.* In the reinforcement learning component, the proximal policy optimization (as outlined in Eq. 18) demonstrates an average improvement of 0.2 compared to PG across five of the six datasets. Nonetheless, on the Twitter dataset, PG outperforms PPO by approximately 0.1, which we postulate might be attributed to overfitting in PPO under the same sampling conditions. (4) The full set of components (i.e., *NCSAC*) yields the best performance, indicating the effectiveness of these components in enhancing algorithm performance. In summary, these results provide some preliminary evidence that the four key components we have designed are practical and effective in real-world applications.

7 RELATED WORK

Rule-based Community Search. Community search has emerged as an essential instrument for revealing the local structures within networks [18]. Numerous models have been proposed in the literature, which can be categorized into two main types: subgraph-based and optimization-based. Specifically, k-core [5, 13, 19, 40], k-truss [1, 24], and k-ecc [2] are the representative subgraph-based models, which restrict the local connectivity of nodes or edges. For example, k-core is a subgraph such that each node has at least k-neighbors within the subgraph. On the other hand, density [14, 48, 53], modularity [27], and conductance [3, 21, 29, 34] are the representative optimization-based models, which aim to achieve a globally optimal objective function. However, these rule-based community search models suffer from the following defects: (1) They (excluding conductance) focus on internal cohesion and neglects external sparsity, which violates the natural intentions of the community [21, 26, 31]. (2) Conductance is defined based on structure, thus neglecting important node attributes. (3) They are plagued by structural inflexibility, wherein the ground-truth community may not align with user-predefined constraints, thereby resulting in suboptimal community quality [8, 9, 17, 25, 32, 43].

Learning-based Community Search. With the swift progress of artificial intelligence, researchers have commenced the integration of Graph Neural Network (GNN) into community search. For instance, *ICS-GNN* [8] pioneered the GNN-based approach to community search by iteratively identifying the target community with maximum GNN scores after crawling online subgraphs. *QD-GNN* [25] combined local query dependency structures with global graph embedding. *CommunityAF* [9] utilized autoregressive flow to generate communities and can learn various community patterns. *CGNP* [17] and *COCLEP* [32] leveraged meta-learning and contrastive learning to solve the community search problem, respectively. However, all of these approaches completely overlook the traditional rule-based constraints. Recently, *ALICE* [44] and *TransZero* [43] effectively integrate traditional rule-based subgraph metrics with GNN, resulting in substantial improvements. At a high level, they first extracted a predefined new subgraph G_{new}, then identified the target community by executing the node classification task within G_{new}. Unfortunately, they are heavily dependent on the performance of G_{new}, since nodes outside G_{new} have no chance to contribute to the target community, resulting in poor quality (Section 6). However, our method focuses on optimizing communities by strategically removing and adding nodes, starting from G_{new}.

8 CONCLUSION

In this paper, we propose a novel Neural Community Search via Attribute-augmented Conductance, termed *NCSAC*, to integrate deep learning techniques with traditional rule-based constraints, thereby improving the quality of community search. Specifically, *NCSAC* utilizes the proposed attribute-augmented conductance to adaptively extract a coarse candidate community with satisfactory quality. Subsequently, *NCSAC* refines the candidate community through advanced reinforcement learning techniques, ultimately producing high-quality results. Extensive experiments are conducted on six real-life graphs, in comparison with ten competitors, to demonstrate the superiority of our proposed solution.

REFERENCES

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

[2] Takuya Akiba, Yoichi Iwata, and Yuichi Yoshida. 2013. Linear-time enumeration of maximal k-edge-connected subgraphs in large networks by random contraction. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 909–918.

[3] Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. 2006. Local Graph Partitioning using PageRank Vectors. In *FOCS*. 475–486.

[4] Anonymous Author. [n. d.]. Full paper for NCSAC: Effective Neural Community Search via Attribute-augmented Conductance. In *https://anonymous.4open.science/r/ncsac-7642*.

[5] Nicola Barbieri, Francesco Bonchi, Edoardo Galimberti, and Francesco Gullo. 2015. Efficient and effective community search. *Data mining and knowledge discovery* 29 (2015), 1406–1433.

[6] Weibin Cai, Fanwei Zhu, Zemin Liu, and Minghui Wu. 2023. HeteroCS: A Heterogeneous Community Search System With Semantic Explanation. In *SIGIR*. 3155–3159.

[7] Tanmoy Chakraborty, Sikhar Patranabis, Pawan Goyal, and Animesh Mukherjee. 2015. On the formation of circles in co-authorship networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 109–118.

[8] Jiazun Chen, Jun Gao, and Bin Cui. 2023. ICS-GNN+: lightweight interactive community search via graph neural network. *The VLDB Journal* 32, 2 (2023), 447–467.

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

[10] Aaron Clauset. 2005. Finding local community structure in networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 72, 2 (2005), 026132.

[11] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. [n. d.]. *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company.

[12] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yiqi Lu, and Wei Wang. 2013. Online search of overlapping communities. In *Proceedings of the 2013 ACM SIGMOD international conference on Management of data*. 277–288.

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

[14] Yizhou Dai, Miao Qiao, and Lijun Chang. 2022. Anchored Densest Subgraph. In *SIGMOD*. 1200–1213.

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

[16] Yang Deng, Yaliang Li, Fei Sun, Bolin Ding, and Wai Lam. 2021. Unified conversational recommendation policy learning via graph-based reinforcement learning. In *SIGIR*. 1431–1441.

[17] Shuheng Fang, Kangfei Zhao, Guanghua Li, and Jeffrey Xu Yu. 2023. Community search: a meta-learning approach. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. 2358–2371.

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

[19] Yixiang Fang, Zhongran Wang, Reynold Cheng, Hongzhi Wang, and Jiafeng Hu. 2018. Effective and efficient community search over large directed graphs. *IEEE Transactions on Knowledge and Data Engineering* 31, 11 (2018), 2093–2107.

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

[21] 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.

[22] Nicolas Heess, Dhruva Tb, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. 2017. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286* (2017).

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

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

[25] 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.

[26] Raj Kamal and Amitabha Bagchi. 2024. A Lovász-Simonovits Theorem for Hypergraphs with Application to Local Clustering. *Proc. ACM Manag. Data* 2, 4 (2024), 190:1–190:27.

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

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

[29] Kyle Kloster and David F. Gleich. 2014. Heat kernel based community detection. In *KDD*. 1386–1395.

[30] Yu Lei, Hongbin Pei, Hanqi Yan, and Wenjie Li. 2020. Reinforcement learning based recommendation with graph convolutional q-network. In *SIGIR*. 1757–1760.

[31] Jure Leskovec, Kevin J. Lang, and Michael W. Mahoney. 2010. Empirical comparison of algorithms for network community detection. In *WWW*, Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti (Eds.). 631–640.

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

[33] Longlong Lin, Tao Jia, Zeli Wang, Jin Zhao, and Rong-Hua Li. 2024. PSMC: Provable and Scalable Algorithms for Motif Conductance Based Graph Clustering. In *KDD*. 1793–1803.

[34] Longlong Lin, Ronghua Li, and Tao Jia. 2023. Scalable and Effective Conductance-Based Graph Clustering. In *AAAI*. 4471–4478.

[35] Tianhao Peng, Wenjun Wu, Haitao Yuan, Zhifeng Bao, Zhao Pengru, Xin Yu, Xuetao Lin, Yu Liang, and Yanjun Pu. 2024. Graphrare: Reinforcement learning enhanced graph neural network with relative entropy. In *ICDE*. 2489–2502.

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

[37] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. 1889–1897.

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[39] Rishi Shah, Krishnanshu Jain, Sahil Manchanda, Sourav Medya, and Sayan Ranu. 2024. NeuroCUT: A Neural Approach for Robust Graph Partitioning. In *KDD*. 2584–2595.

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

[41] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* 12 (1999).

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

[43] Jianwei Wang, Kai Wang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. 2024. Efficient Unsupervised Community Search with Pre-trained Graph Transformer. *arXiv preprint arXiv:2403.18869* (2024).

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

[45] 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.

[46] Ryan Wickman. 2022. SparRL: Graph Sparsification via Deep Reinforcement Learning. In *SIGMOD*. 2521–2523.

[47] Xixi Wu, Yun Xiong, Yao Zhang, Yizhu Jiao, Caihua Shan, Yiheng Sun, Yangyong Zhu, and Philip S. Yu. 2022. CLARE: A Semi-supervised Community Detection Algorithm. In *KDD*. 2059–2069.

[48] 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.

[49] Xiaoqin Xie, Shuangyuan Liu, Jiaqi Zhang, Shuai Han, Wei Wang, and Wu Yang. 2024. Efficient Community Search Based on Relaxed k-Truss Index. In *SIGIR*. 1691–1700.

[50] Jaewon Yang and Jure Leskovec. 2012. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD workshop on mining data semantics*. 1–8.

[51] Renchi Yang, Xiaokui Xiao, Zhewei Wei, Sourav S. Bhowmick, Jun Zhao, and Rong-Hua Li. 2019. Efficient Estimation of Heat Kernel PageRank for Local Clustering. In *SIGMOD*. 1339–1356.

[52] Kai Yao and Lijun Chang. 2021. Efficient size-bounded community search over large networks. *Proceedings of the VLDB Endowment* 14, 8 (2021), 1441–1453.

[53] Xiaowei Ye, Rong-Hua Li, Lei Liang, Zhizhen Liu, Longlong Lin, and Guoren Wang. 2024. Efficient and Effective Anchored Densest Subgraph Search: A Convex-programming based Approach. In *KDD*. 3907–3918.

[54] Yunfeng Yu, Longlong Lin, Qiyu Liu, Zeli Wang, Xi Ou, and Tao Jia. 2024. GSD-GNN: Generalizable and Scalable Algorithms for Decoupled Graph Neural Networks. In *ICMR*. 64–72.

[55] Guozhen Zhang, Yong Li, Yuan Yuan, Fengli Xu, Hancheng Cao, Yujian Xu, and Depeng Jin. 2021. Community value prediction in social E-commerce. In *Proceedings of the Web Conference 2021*. 2958–2967.

[56] Yao Zhang, Yun Xiong, Yun Ye, Tengfei Liu, Weiqiang Wang, Yangyong Zhu, and Philip S Yu. 2020. SEAL: Learning heuristics for community detection with generative adversarial networks. In *KDD*. 1103–1113.

[57] Shangfei Zheng, Hongzhi Yin, Tong Chen, Quoc Viet Hung Nguyen, Wei Chen, and Lei Zhao. 2023. Dream: Adaptive reinforcement learning based on attention mechanism for temporal knowledge graph reasoning. In *SIGIR*. 1578–1588.

A APPENDIX

A.1 Graph Neural Network

Graph Neural Network (GNN) is a specialized neural network structure tailored for processing graph data [20, 28, 42]. It's engineered to encode network nodes as low-dimensional vectors, preserving both the network topology and node feature information, thus facilitating downstream tasks such as node classification and graph clustering [28, 54]. GNN governs the information propagation and update process by defining node aggregation and update functions, enabling nodes to assimilate information from their neighbors and refine their representations. This iterative process continues until the model converges: $h_u^{(l+1)} = UPDATE(h_u^{(l)}, AGGREGATE(h_v^{(l)}|v \in N(u)))$, in which $h_u^{(l)}$ indicates the representation of vertex u in layer l. $AGGREGATE(\cdot)$ is defined as a node aggregation function that weighs the representation of vertex v w.r.t. vertex u. $UPDATE(\cdot)$ is an update function that iteratively refines the features of all nodes in the graph based on the aggregated information.

A.2 Missing Proofs

Proof of Theorem 4.2. Firstly, according to Eq. (1), we can deduce that $\phi(C) = \frac{|cut(C,\bar{C})|}{\min\{vol(C),vol(\bar{C})\}} = \max\{\frac{|cut(C,\bar{C})|}{vol(C)}, \frac{|cut(C,\bar{C})|}{vol(\bar{C})}\}$. Then, we have $Pr(w_1 \in \bar{C}|w_0 \in C) = \sum_{u\in C}\sum_{v\in\bar{C}} \frac{d(u)}{vol(C)} \cdot \boldsymbol{P}_{uv} = \sum_{u\in C}\sum_{v\in\bar{C}\cap N(u)} \frac{d(u)}{vol(C)} \cdot \frac{1}{d(u)} = \sum_{u\in C}\sum_{v\in\bar{C}\cap N(u)} \frac{1}{vol(C)} = \frac{|cut(C,\bar{C})|}{vol(C)}$. Here, $\frac{d(u)}{vol(C)}$ denotes the probability of randomly selecting node u from community C following the degree distribution, while $\boldsymbol{P}_{uv}$ represents the probability that node u transitions to node v by one-hop random walk. Analogously, we have $Pr(w_1 \in C|w_0 \in \bar{C}) = \frac{|cut(C,\bar{C})|}{vol(\bar{C})}$. Consequently, the theorem is proved. ■

Proof of Lemma 4.10. By Eq. 3, we have $d_a(u) = \sum_{v\in V\setminus\{u\}} \mathcal{F}[u] \cdot \mathcal{F}[v]^\top = \mathcal{F}[u] \cdot \sum_{v\in V\setminus\{u\}} \mathcal{F}[v]^\top = \mathcal{F}[u] \cdot ((\mathbf{1}_n\mathcal{F})^\top - \mathcal{F}[u]^\top) = \mathcal{F}[u] \cdot (\mathbf{1}_n\mathcal{F} - \mathcal{F}[u])^\top = \mathcal{F}[u] \cdot (\mathbf{1}_n\mathcal{F} - \mathbf{1}_k)^\top$. This is because every element in $\mathcal{F}[u]$ is either 0 or 1, $\mathcal{F}[u] \cdot \mathcal{F}[u]^\top = \mathcal{F}[u] \cdot \mathbf{1}_k^\top$. Thus, this lemma is proved. ■

Proof of Lemma 4.11. By Definition 4.1, we have $cut(C \cup \{u\}, V \setminus (C \cup \{u\})) = \left|cut(C,\bar{C})\right| - |cut(u,C)| + \left|cut(u,\bar{C})\right| = \left|cut(C,\bar{C})\right| + d(u) - 2\,|cut(u,C)|$ due to $|cut(u,C)| + \left|cut(u,\bar{C})\right| = d(u)$. Similarly, we can prove $cut_a(C \cup \{u\}, V \setminus (C \cup \{u\})) = \left|cut_a(C,\bar{C})\right| + d_a(u) - 2\,|cut_a(u,C)|$. Thus, this lemma is proved. ■

Proof of Theorem 4.13. Algorithm 1 first takes $O(nk)$ time to calculate the attribute degree vector d_a, $O(n)$ time to obtain $vol_a(V)$, cut, vol, cut_a, and vol_a (Lines 1-2). In Lines 4-13, Algorithm 1 can incrementally calculate and update attribute-augmented conductance by Fact 4.2. Specifically, we assume that u is the added vertex in Line 6, it takes $O(d(u))$ time to incrementally update cut and $O(k)$ time to update cut_a and att. Thus, it takes $O(\sum_{u\in V}(d(u)+k)) = O(m+nk)$ to execute Lines 4-13. As a result, Algorithm 1 takes $O(m+nk)$ time in total. For the space complexity, Algorithm 1 just needs an extra $O(n+k)$ to store d_a, C, tmp, and att. Besides, it needs $O(m+n)$ to store the input graph G. As a consequence, Algorithm 1 needs $O(n+m+k)$ space in total. ■

Proof of Theorem 5.2. Let $\Pi = \{< (u_1, op_1), (u_2, op_2), ... > |u_i \in G, op_i \text{ is } append \text{ or } remove\}$ be the set of all possible refinement orderings, where $(u_i, append)$ (resp., $(u_i, remove)$) means that we append (resp., remove) the node u_i for the current community. Thus, by Definition 5.1, we can know that the goal of community refinement is to find an ordering $\pi \in \Pi$ such that $\Upsilon(C_{coa} + \pi)$ has the optimal score value. Note that $C_{coa} + \pi$ is a new community by executing the operation π for C_{coa}. Based on the above concepts, we can reduce the well-known NP-hard problem of the maximum subset sum problem (decision version) to the community refinement problem within polynomial time. Given a set of integers $S = \{s_1, s_2, ...s_n\}$ and a target integer t, the subset sum decision problem is to check whether there exists a subset $S' \subseteq S$ such that the sum of the elements in S' equals t. From this, we have the following steps. (1) Graph Construction: Construct a graph G where each integer $s_i \in S$ is represented as a node, and let the candidate coarse community $C_{coa} = \emptyset$. (2) Score Function Definition: we define the score function $\Upsilon(C)$ for any community C as follows:

$$\Upsilon(C) = \begin{cases} 1, & \text{if the sum of the nodes in } C \text{ is exactly } t \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

(3) Polynomial Time Reduction: Given the set S and target t, construct the graph G, candidate coarse community C_{coa}, and the score function $\Upsilon(C)$ as described above. The task of finding a subset $S' \subseteq S$ such that the sum of its elements is t translates to finding a community $C_{opt} \subseteq G$ such that $\Upsilon(C_{opt}) = 1$. Therefore, this, in turn, is equivalent to finding an ordering $\pi \in \Pi$ such that $\Upsilon(C_{coa}+\pi) = 1$, where $C_{coa}+\pi$ represents the community obtained by applying the sequence of operations π starting from C_{coa}. (4) Equivalence of Problems: If there exists a subset $S' \subseteq S$ such that the sum of its elements is t, then there exists an ordering $\pi \in \Pi$ such that $\Upsilon(C_{coa} + \pi) = 1$. Conversely, if there exists an ordering $\pi \in \Pi$ such that $\Upsilon(C_{coa} + \pi) = 1$, then the corresponding subset of nodes in G (which represents the elements of S) has a sum of exactly t.

Since the subset sum problem (decision version) is NP-hard [11], and we have shown that it can be reduced to the community refinement problem in polynomial time, it follows that the community refinement problem is also NP-hard. ■

A.3 Offline Training

Figure 7 illustrates the details of the offline training. Specifically, during the offline training phase, we initially sample a candidate coarse community from the training dataset and permit the agent to execute the complete community refinement process. Specifically, the agent begins by encoding the state of the community (illustrated by the color bar in the figure), and subsequently employs the policy network to decode this state into score values (represented by the blue and green bars), which are then utilized to calculate the reward and update the state. It is crucial to emphasize that, to strike a balance between exploration and exploitation during training, we adopt an ϵ-greedy action strategy. Under this strategy, there is a probability of $1-\epsilon$ that we will select the node with the highest (or lowest) score, while with a probability of ϵ, a random action is taken. The value of ϵ is linearly annealed from 1.0 to 0.05 throughout the refinement process. Upon completion of this entire procedure, we derive a training trajectory $\mathcal{T}$.

A.4 Additional Experiments

Scalability testing. To further evaluate the scalability of our adaptive community extractor, we first generate 60 synthetic subgraphs by randomly sampling 20%, 40%, 60%, 80%, and 100% of vertices or edges from each of the six datasets. Subsequently, we present the runtime on these subgraphs in Figure 8. As depicted, the runtime of Algorithm 1 scales nearly linearly with the size of the subgraphs.

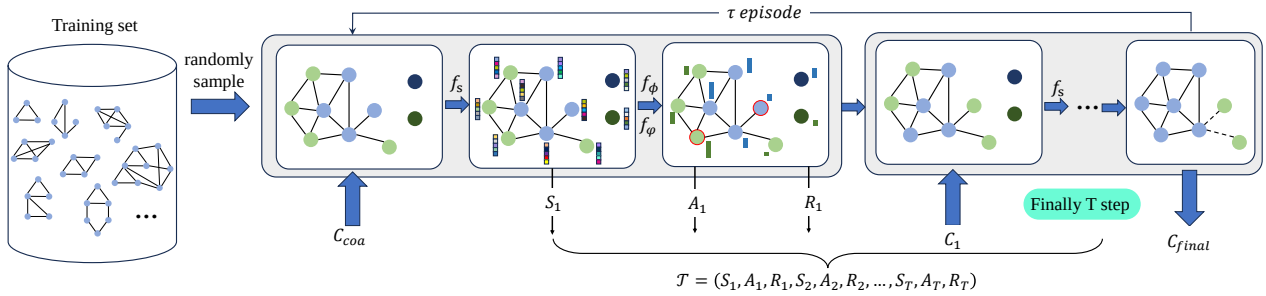


Figure 7: Offline training. Initiating the refinement process from the candidate coarse community, each step generates a new set of states, actions, and rewards, continuing until the predefined termination policy is activated. This procedure is conducted over τ episodes, during which the agent is trained utilizing the τ trajectories obtained (best review in color).

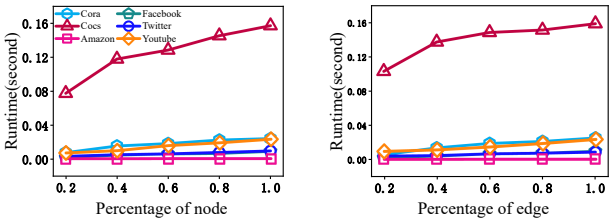


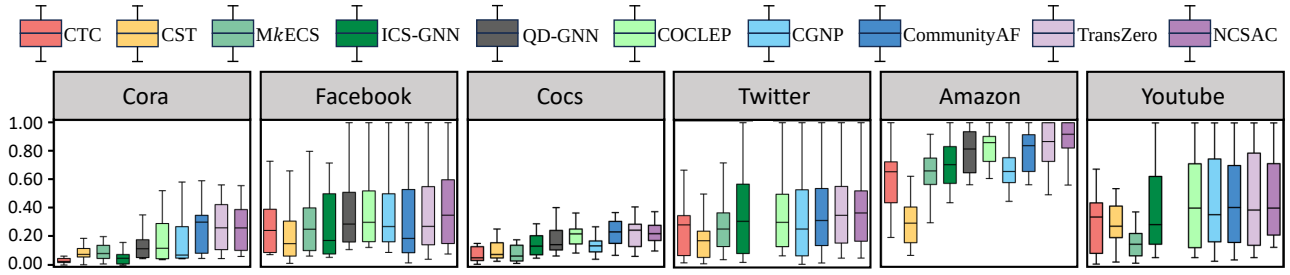
Figure 8: Scalability testing.

These findings suggest that our proposed algorithm is capable of managing large-scale networks.

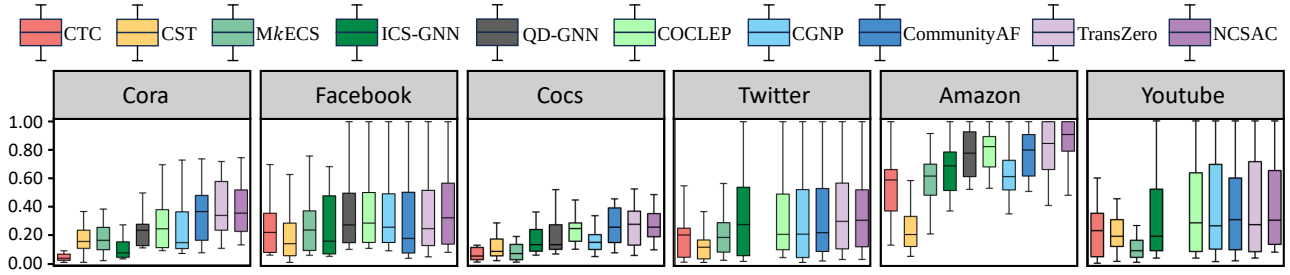
Comparison of our algorithm NCSAC with baselines. Figure 9 provides a detailed box plot showing the NMI and JAC values achieved by different algorithms across all datasets. Since these metrics follow similar patterns to the F1-score (as illustrated in Figure 4), we refrain from further discussion on them. In summary, the results demonstrate that our model outperforms the state-of-the-art baselines in identifying higher-quality communities.

Performance of NCSAC with varying parameters. This experiment investigates how the parameters affect the performance of our *NCSAC*. Figure 10 only reports the F1-score, with analogous trends observed on NMI and JAC. Specifically, we have the following conclusions: (1) Figure 10 (a) presents the results obtained by varying β, which strikes a balance between topology-based conductance and attribute-based conductance in Equation 6. As depicted, on five out of six datasets (with Amazon being an exception showing no significant change with β), the performance of *NCSAC* diminishes as β rises. This trend can be intuitively understood as follows: as β increases, the weight assigned to topological factors grows, thereby overshadowing the influence of attribute-based factors and subsequently resulting in a decline in quality. This finding highlights the pivotal role of attribute similarity and validates the effectiveness of our proposed attribute-augmented conductance. (2) Figure 10 (b) illustrates the performance of *NCSAC* with varying α, which serves to balance the triplet loss and contrastive loss in Equation 11. As can be seen, The F1-score remains relatively stable as α increases, indicating that the impact of the triplet loss significantly exceeds that of the contrastive loss. Subsequent ablation studies provide further support for this observation. (3) Figure 10 (c) presents the influence of the number of episodes τ on the F1-score. As observed, the F1-score experiences a rapid increase (by 0.2-0.3) within the initial 800 episodes, followed by a gradual slowdown (of less than 0.05) in the subsequent 800-2000 episodes. This finding suggests that a relatively small number of episodes are sufficient to achieve a high F1-score. (4) Let η be the fixed number of nodes refined during the refinement phase (section 5.2) and $RFS = \frac{F1\text{-}score\ obtained\ by\ FixNCSAC}{F1\text{-}score\ obtained\ by\ NCSAC}$ be the relative F1-score, where FixNCSAC is the NCSAC with the fixed number of nodes refined. This experiment studies the effectiveness of the proposed flexible termination strategy (details in Section 5.2). Figure 10 (d) shows the results. As illustrated, *RFS* consistently remains below 1 and exhibits no discernible pattern of variation across all datasets. Thus, this result provides compelling evidence that our proposed flexible termination strategy consistently outperforms the fixed-node refinement strategy.

Case studies. We conduct case studies on Facebook [50] with the ground-truth community to further evaluate the utility of the proposed NCSAC. Specifically, we choose node 2108 as the query node (marked in red), and the communities identified by different models are visualized in Figure 11. As can be seen, the communities identified by both CommunityAF and TransZero include nodes (depicted in green) that do not belong to the ground-truth community. Besides, they also miss many nodes within the ground-truth community. Consequently, their F1-scores are 0.8 and 0.77, respectively. In contrast, our NCSCA exclusively identifies nodes from the ground-truth community, with only three nodes from the ground-truth community being undetected. Thus, NCSCA achieves the highest congruence with the ground-truth community, boasting an impressive F1-score of 0.92. This result can be explained as follows: TransZero employs a stricter termination condition, halting the search once the community quality starts to diminish. While this prevents over-expansion, it also restricts the exploration of potentially relevant nodes. CommunityAF, on the other hand, utilizes a sliding window mechanism, enabling it to explore a broader range of promising nodes but at the expense of including more irrelevant ones. However, our NCSCA adopts the more flexible termination strategy (details in Section 6.2), which allows it to align closely with the ground-truth community.

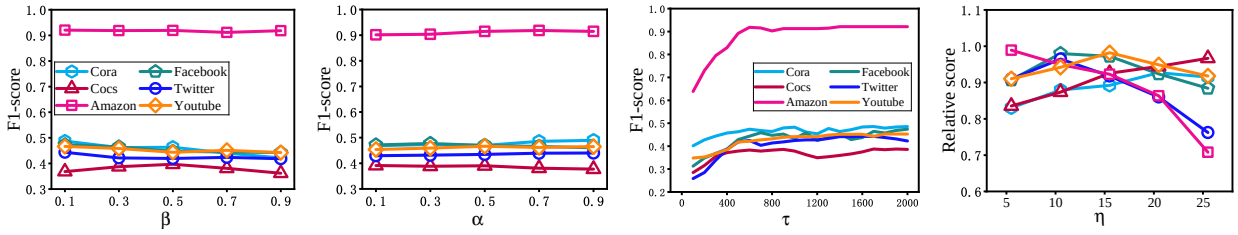


(a) NMI of different algorithms



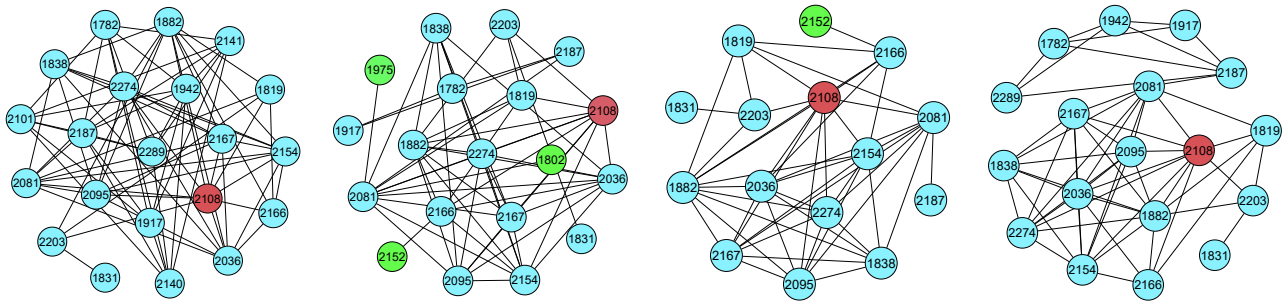
(b) JAC of different algorithms

Figure 9: Comparison of our algorithm NCSAC with baselines. Since QD-GNN runs out of memory on Twitter and YouTube, we only report their results on the four remaining datasets (best view in color).



(a) F1-score with varying β (b) F1-score with varying α (c) F1-score with varying τ (d) Relative F1-score with varying η

Figure 10: Performance of NCSAC with varying parameters.



(a) the ground-truth community (b) CommunityAF (c) TransZero (d) NCSAC

Figure 11: Case studies. The red node represents the query node, while the green node belongs to the ground-truth community and the blue node does not.