LMSC: Local Sketch Modularity Optimisation for Size-Constrained Community Search in Networks

DAHEE KIM, Ulsan National Institute of Science and Technology, South Korea
TAEJOON HAN, Ulsan National Institute of Science and Technology, South Korea
KAIYU FENG, Beijing Institute of Science and Technology, China
JUNGHOON KIM*, Ulsan National Institute of Science and Technology, South Korea
SUSIK YOON, Korea University, South Korea

With the proliferation of social networks, identifying meaningful community structures efficiently is essential for analysing complex interactions. This paper introduces Local Sketch Modularity (LSM), a novel modularity that measures community quality without relying on the entire structural information of the network, enabling a more targeted and practical approach to find the query-centric community. We validate the efficacy of the proposed modularity LSM through theoretical analyses, showing robustness against the free-rider effect. We further formulate the Local Modularity Optimisation for Size-Constrained Community Search (LMSC) problem, which leverages LSM to identify the query-centric community without requiring knowledge of the entire graph. We prove that LMSC is NP-hard and propose two efficient and effective algorithms. Extensive experiments on real-world networks demonstrate both the effectiveness and efficiency of the proposed method, confirming its applicability for large-scale network analysis.

CCS Concepts: • **Information systems** → **Clustering**; • **Theory of computation** → *Graph algorithms analysis.*

Additional Key Words and Phrases: Community Search, Cohesive Subgraph Discovery, Modularity

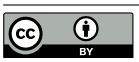
ACM Reference Format:
Dahee Kim, Taejoon Han, Kaiyu Feng, Junghoon Kim, and Susik Yoon. 2026. LMSC: Local Sketch Modularity Optimisation for Size-Constrained Community Search in Networks. *Proc. ACM Manag. Data* 4, 1 (SIGMOD), Article 52 (February 2026), 26 pages. https://doi.org/10.1145/3786666

1 INTRODUCTION

Community search has emerged as a core task in graph mining, with applications spanning social network analysis, biological network interpretation, and fraud detection [7, 20, 35]. In contrast to community detection [11], which aims to enumerate all communities in a graph, community search is query-centric: given a set of query nodes, the objective is to extract a densely connected subgraph that contains the query and satisfies application-specific constraints [9, 16, 17, 35]. Community search is often more practical in real applications, since users are typically interested in the communities containing specific query nodes rather than a full enumeration [9].

*Corresponding author.

Authors' Contact Information: Dahee Kim, Ulsan National Institute of Science and Technology, Ulsan, South Korea, dahee@unist.ac.kr; Taejoon Han, Ulsan National Institute of Science and Technology, Ulsan, South Korea, cheld7132@unist.ac.kr; Kaiyu Feng, Beijing Institute of Science and Technology, Beijing, China, fengky@bit.edu.cn; Junghoon Kim, Ulsan National Institute of Science and Technology, Ulsan, South Korea, junghoon.kim@unist.ac.kr; Susik Yoon, Korea University, Seoul, South Korea, susik@korea.ac.kr.



ACM 2836-6573/2026/2-ART52
https://doi.org/10.1145/3786666

A common requirement in real-world scenarios is to impose explicit constraints on the size of communities [13, 28]. In practice, community size is often dictated by factors such as physical capacity, cost, or functional scope. For example, in team formation, venue size or budget limits the number of participants [23]; in biological systems, functional modules typically comprise a small set of closely interacting proteins [36]. These needs motivate the development of community search methods that accommodate size constraints.

To solve this problem, a number of approaches have been proposed. Most of the approaches are based on classical structural patterns such as k-core [43] and k-truss [23, 45], which aim to extract dense subgraphs by enforcing specific topological constraints. While effective in identifying dense subgraphs, these methods suffer from several limitations [18].

(1) *Inflexibility.* These structural approaches rely on pre-defined topological constraints (e.g., minimum degree or triangle participation) that real-world communities may not satisfy [21, 38]. As a result, they are often too inflexible to adapt to diverse networks or tasks [18]; and
(2) *Global information dependency.* Many of these methods require complete access to the global graph structure, limiting scalability in large networks. Furthermore, in real-world settings, privacy constraints often result in partial or missing data [1], which poses challenges for topology-based optimisation methods in accurately detecting communities [37]; and
(3) *Neglect of external separability.* Most existing methods prioritise internal cohesion but do not consider the separation of the community from the rest of the graph—an essential property for identifying meaningful communities.

These limitations suggest an alternative direction: modularity-based approaches, which are widely used to evaluate the quality of communities. Compared to structural models, modularity offers a more adaptive measure that better aligns with the diverse real-world networks. Classical modularity [30], a measure designed for community detection, compares observed edge density against a null model. However, when applied to community search, its global optimisation nature often leads to the *free-rider effect* [40], where loosely connected nodes are unnecessarily included in the community. To alleviate this, density modularity [20] was introduced to favour smaller and denser communities, and more recently, generalised versions such as density sketch modularity [38] have attempted to refine the balance between community size and density. However, these modularities still face two fundamental limitations: (1) *Global Information Dependency.* These measures fundamentally rely on global graph statistics to compute expected edge patterns. (2) *Free-rider effect.* Since these measures rely on global network information, they become inherently susceptible to the detection of small-sized community [12] as the graph size increases.

In addressing this issue, we revisit Luo modularity [25], a locally computable objective designed for query-dependent community search. Luo modularity is defined as the ratio between the number of internal edges and the number of external edges of the community, which allows it to avoid global structure dependency and operate on local information. While this approach avoids global dependency, it has a critical flaw in real-world settings: as the community grows, the ratio of internal to external edges also increases, causing the method to favour larger communities. This size bias becomes a serious problem when strict size limits are required—leading to communities that are too large or loosely connected, thereby reducing interpretability and practical utility. Luo et al. [25] attempted to mitigate this bias at the algorithmic level by employing early termination strategies to extract smaller communities, but this only addresses the symptom rather than the root cause. Therefore, we aim to resolve this limitation intrinsically by redesigning the objective itself.

We propose Local Sketch Modularity (LSM), a principled extension of Luo modularity that introduces a size penalisation term to directly suppress the large-community bias while preserving its local computation advantages. By incorporating size-awareness into the objective function,

the proposed method identifies compact and interpretable communities. Specifically, our measure provides the following key benefits:

- **Structural flexibility:** Unlike models such as k-core or k-truss, our modularity-based formulation avoids rigid structural assumptions, enabling broader applicability in diverse scenarios.
- **Local computation:** Local Sketch Modularity operates on the local structure by not requiring the entire graph. This property facilitates scalability in large-scale or partially accessible networks.
- **Mitigating the free-rider effect:** Compared to Luo modularity, Local Sketch Modularity is less affected by the free-rider issue.

The following example illustrates both the practical motivation for our formulation and the shortcomings of existing methods.

Example 1 (Motivating Scenario). *Consider a practical scenario where Evan and Joy plan to organise a private Christmas gathering at a BBQ restaurant. To reserve the venue, they must ensure a minimum of* 10 *participants, while the space allows at most* 20*. For a successful and enjoyable event, they aim to invite a group of friends who are socially close and interact comfortably with one another. Since the gathering is intended to bring together a close group of friends, participants should be more closely connected within the group than with people outside the group. At the same time, they want to avoid inviting people who are only loosely connected to the group (i.e., free-riders)—for example, acquaintances who know just one or two others. Including such individuals might lead to awkward conversations and weaken the overall bond of the group.*

This scenario demonstrates that a community search method must satisfy several essential criteria: (1) **strong internal cohesion** where members are closely connected to each other, (2) **external separation** where the community is well-separated from the outside, (3) **avoidance of loosely connected individuals** who might act as free-riders.

Existing structural methods (e.g., k-core [33], k-truss [6]) capture internal density well but often ignore external separability. Modularity-based methods tend to favour large communities, making them prone to free-rider inclusion. And both methods typically may require full graph access, which is impractical in many real-world settings. In contrast, our *Local Sketch Modularity* operates locally, balances cohesion and separation, and suppresses loosely connected free-riders through a size-aware penalty term.

In this paper, we formalise this framework as the *Local Sketch Modularity Optimisation for Size-Constrained Community Search (LMSC)* problem. Given a graph $G = (V, E)$, a query node set $Q \subseteq V$, and a size constraint $[l, h]$, the goal is to find a community C such that $Q \subseteq C$, $l \leq |C| \leq h$, and the Local Sketch Modularity of C is maximised. We prove that the problem is NP-hard.

Contributions. Our main contributions are summarised as follows:

(1) We propose Local Sketch Modularity, a new local objective function for size-constrained community search, and formalise the LMSC problem. Theoretical results include a proof of NP-hardness and an analysis of robustness to free-rider effects.
(2) We develop two scalable algorithms: Incremental Greedy Algorithm (IGA) and Sub-chain Merge Algorithm (SMA) for efficient optimisation of LMSC.
(3) We empirically validate our approach on real-world networks, demonstrating that our methods achieve F1 and ARI scores up to 1.39 and 5.95 times higher, respectively, than the best performing baseline, while maintaining strong computational efficiency.

2 PRELIMINARIES

This section presents basic notations and concepts used in this paper. We represent $G = (V, E)$ as an undirected weighted graph, where V is the set of nodes and E is the set of edges. Each edge

Table 1. Summary of community measures. CD and CS denote community detection / search, respectively.

Measure	Bias to large comm.	Entire graph	Target
NM [29]	High	Required	CD
DM [21]	Medium	Required	CS
DSM [38]	Adjustable	Required	CS
LM [25]	High	Not Required	CS
LSM	Adjustable	Not Required	CS

$(v, u) \in E$ is associated with a non-negative weight $w(v, u)$ representing the strength of connection between v and u. A community, denoted by C, corresponds to a set of nodes of V. Let l_C represent the sum of internal edge weights within C; o_C denotes the sum of edge weights between C and $V \setminus C$. We define the node strength $s(v)$ of a node v as the sum of the weights of edges incident to v, i.e., $s(v) = \sum_{u \in N(v)} w(v, u)$. Accordingly, the total strength of C is given by $d_C = o_C + 2l_C$.

For a subset of nodes $H \subset V$, we define $N(H, G)$ as the set of neighbour nodes of H in G. Formally, $N(H, G) = \bigcup_{u \in H} N(u, G)$ where $N(u, G)$ represents the set of neighbours of a node u in G.

We now review modularity-based objectives commonly used in community detection and community search. Newman modularity [29] compares observed edge density within communities to random expectations:

Definition 1. *(Newman Modularity [29]). Given a graph $G = (V, E)$ and a set of nodes $C \subseteq V$, $NM(G, C) = \frac{1}{2|E|}\left(2l_C - \frac{d_C^2}{2|E|}\right)$, where l_C denotes the sum of internal edge weights, d_C is the sum of all node weights, and $|E|$ is the total number of edges.*

While foundational, maximising Newman modularity (NM) is NP-hard [3] and suffers from the well-known resolution limit [12], which causes small communities to be merged into larger ones as the graph size increases. To address the detection of smaller and denser communities, Kim et al. [21] proposed density modularity (DM):

Definition 2. *(Density Modularity [21]). Given a graph $G = (V, E)$ and a set of nodes $C \subseteq V$, $DM(G, C) = \frac{1}{2|C|}\left(2l_C - \frac{d_C^2}{2|E|}\right)$, where $|C|$ is the number of nodes in C.*

Density modularity extends classical modularity for the community search problem by explicitly incorporating community size. Wang et al. [38] further refined DM with a cohesiveness factor τ which enables adjustable control over the preference for a smaller or denser community:

Definition 3. *(Density Sketch Modularity [38]). Given a graph $G = (V, E)$, a set of nodes $C \subseteq V$, and $\tau \in \mathbb{R}_{\geq 0}$, $DSM(G, C) = \frac{1}{2|C|^\tau}\left(2l_C - \frac{d_C^2}{2|E|}\right)$.*

Despite the efforts to adapt modularity to community search, these measures fundamentally depend on global graph statistics, such as the total number of edges or the degrees of all nodes. This reliance restricts their scalability to large-scale or partially observable networks. Moreover, as the graph size increases, the score becomes increasingly dominated by the internal edge term l_C, making it difficult to detect small communities [12].

Luo Modularity (LM). Feng et al. [25] proposed Luo modularity specifically for query-dependent community search:

Definition 4. *(Luo Modularity [25]). Given a graph $G = (V, E)$ and a set of nodes $C \subseteq V$, $LM(G, C) = \frac{l_C}{o_C}$, where o_C denotes the sum of external edge weights.*

Unlike modularity-based measures that require global graph information and null models, LM operates purely on local structure, relying only on the internal and external edge weights of C. This locality enables scalable community search, even when only partial graph information is available. However, LM suffers from a strong bias towards large communities. As C grows, the number of outgoing edges o_C usually decreases sharply as previously external edges are absorbed into the community, causing the score to grow artificially high—even for overly large or trivial solutions (e.g., $C = V$). As a result, loosely connected or irrelevant nodes are frequently included, which prevents the identification of compact and meaningful communities. To handle this limitation, we propose a principled extension (Section 3.1) that incorporates a size-aware penalty while preserving the desirable locality of LM.

Table 1 shows that unlike NM and LM, our proposed LSM avoids full graph access and allows adjustable control over community size, making it suitable for practical community search. The next section formally introduces LSM and its optimisation problem, along with a motivating example.

3 PROBLEM STATEMENT

We propose a new local modularity measure, Local Sketch Modularity (LSM), addressing existing limitations, and formulate a corresponding problem: Local Modularity Optimisation for Size-Constrained Community Search (LMSC).

3.1 Local Sketch Modularity

Most modularity measures, including Newman [30], Density [21], and Density Sketch modularity [38], require full graph access, which is often impractical. In contrast, Luo modularity (LM) operates locally but tends to favour large communities (see Section 4). While Luo et al. [25] proposed fixes to reduce this bias, such adjustments conflict with the original objective. We introduce a new measure that addresses these limitations while retaining the local, query-focused nature of LM. The proposed modularity yields more cohesive communities using only partial graph information.

Definition 5. *(Local Sketch Modularity). Given a graph $G = (V, E)$, a sketch threshold $\tau \in \mathbb{R}_{\geq 0}$, and a set of nodes $C \subset V$, the local sketch modularity (LSM) is defined as follows:*

$$LSM(G, C, \tau) = \frac{l_C}{o_C} \cdot \frac{1}{|C|^\tau}$$

Observe that the minimum value of LSM is 0 when there are no internal edges within the community. Conversely, the maximum value of LSM is achieved when $\tau = 0$ and a community contains all the nodes except the one with the lowest sum of edge weights. In Section 4, we compare the properties of LSM with those of LM.

A naive regularisation of Luo modularity may penalise large communities indiscriminately, disrupting the internal-to-external structural balance already encoded in the original objective. In contrast, LSM incorporates a size-aware adjustment that preserves local density while effectively discouraging loosely connected nodes (i.e., free-riders). As shown in Section 4, LSM offers stronger robustness to the free-rider effect, and we derive a sketch threshold that guarantees resilience under a certain condition.

3.2 Local Sketch Modularity Optimisation for Size-Constrained Community Search

To address the challenge of identifying a query-centric community, we propose the Local Sketch Modularity Optimisation for Size-Constrained Community Search (LMSC) problem as follows.

Problem definition 1. *Given a graph $G = (V, E)$, a set of query nodes $Q \subseteq V$, a size constraint $[l, h]$, and a sketch threshold $\tau \in \mathbb{R}_{\geq 0}$, the Local Sketch Modularity Optimisation for Size-Constrained*

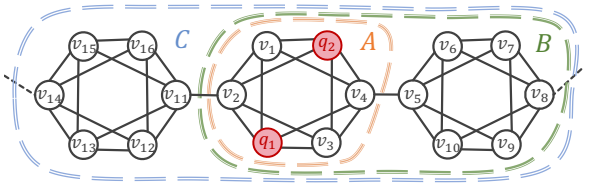


Fig. 1. A simple graph

Community Search (LMSC) problem aims to find a query-centric community C that satisfies the following constraints:

(1) ***Connectivity****: $G[C]$ is connected;*
(2) ***Query containment****: $Q \subseteq C$;*
(3) ***Size constraint****: $l \leq |C| \leq h$;*
(4) ***Maximality****: $LSM(G, C, \tau)$ is maximised.*

The properties and hardness of the problem are specified in Section 4. In the following, we present a simple example to illustrate the differences between LM and LSM.

Table 2. Comparison of LM and LSM

	LM	LSM			
		$\tau = 0.4$	$\tau = 0.8$	$\tau = 1.2$	$\tau = 1.6$
Community A	6.00	2.93	1.43	0.69	0.34
Community B	12.50	4.62	1.71	0.63	0.23
Community C	19.00	5.97	1.88	0.59	0.18

Example 2. *Figure 1 compares Luo modularity (LM) and Local Sketch Modularity (LSM) for three communities A, B, and C. As shown in Table 2, LM consistently assigns high scores to larger communities, illustrating its bias toward size, which leads to the free-rider effect (see Section 4). In contrast, LSM incorporates a tunable parameter τ that penalises overly large communities. As τ increases, smaller, denser communities (e.g., A) are relatively preferred. Notably, LM is a special case of LSM with $\tau = 0$.*

4 THEORETICAL ANALYSIS

4.1 Hardness of LMSC Problem

In this section, we discuss the hardness of the LMSC problem.

Theorem 1. *The LMSC problem is NP-hard.*

We establish NP-hardness by a polynomial-time reduction from Set Cover. Given a Set Cover instance with a set of items and a collection of itemsets, we construct a weighted graph in which nodes represent items and itemsets, with edges encoding set membership. Additional nodes and edge weights are introduced so that minimising the number of selected sets in Set Cover corresponds to maximising the LSM objective when $\tau = 0$. Thus, any optimal solution to the LMSC instance can be mapped to a minimum set cover. Details of the reduction are provided in Appendix A of [19].

4.2 Free-rider Effect

The free-rider effect refers to the inclusion of nodes that do not meaningfully contribute to the query-focused community [21]. It can be classified into *global* and *local* variants [40], with the latter being more relevant to community search tasks, where the presence of query nodes is central. This section focuses on analysing and mitigating the *local free-rider effect*.

Definition 6. *(Local free-rider effect [17, 40]). Given a graph $G = (V, E)$, query nodes Q, the goodness function f. Let $G[H]$ denote the identified community based on the function f and $G[H']$ represent a query-independent local optimum subgraph. The function f suffers from the local free-rider effect if $f(H \cup H') \geq f(H)$.*

4.2.1 Comparison Between LM and LSM. In the following, we theoretically analyse the local free-rider effects of LM and LSM.

Theorem 2. *LSM is less prone to the local free-rider effect than LM.*

Proof. Let H be the set of nodes of an identified subgraph and H' be the set of nodes the optimal subgraph. Let $l_{H,H'}$ be the sum of edge weights between $H \setminus H'$ and $H' \setminus H$, l_{dup} denote the sum of duplicated internal edge weights between H and H', H_{dup} represent the set of duplicated nodes between H and H'. We further define $l_{H_{dup},H'\setminus H}$ and $l_{H_{dup},H\setminus H'}$ as the sum of edge weights between the duplicated nodes H_{dup} and the non-overlapping nodes in H' and H, respectively. To simplify the notation, we denote $l_{cross} = l_{H_{dup},H'\setminus H} + l_{H_{dup},H\setminus H'}$. We now formulate the equations of $LM(H)$, $LM(H \cup H')$, $LSM(H)$, and $LSM(H \cup H')$.

- $\mathsf{LM}(H) = \frac{l_H}{o_H}$
- $\mathsf{LM}(H \cup H') = \frac{l_H + l_{H'} - l_{dup} + l_{H,H'}}{o_H + o_{H'} - l_{cross} - 2l_{H,H'}}$
- $\mathsf{LSM}(H) = \frac{l_H}{o_H} \cdot \frac{1}{|H|^\tau}$
- $\mathsf{LSM}(H \cup H_g^*) = \frac{l_H + l_{H'} - l_{dup} + l_{H,H'}}{o_H + o_{H'} - l_{cross} - 2l_{H,H'}} \cdot \frac{1}{(|H|+|H'|-|H_{dup}|)^\tau}$

We compare the difference between the values of LM and LSM for the free-rider effect. We first consider the case of LM, the corresponding equation is as follows:

$$\mathsf{LM}(H \cup H') - \mathsf{LM}(H) = \frac{l_H + l_{H'} - l_{dup} + l_{H,H'}}{o_H + o_{H'} - l_{cross} - 2l_{H,H'}} - \frac{l_H}{o_H}$$

Let $k = o_H \cdot (o_H + o_{H'} - l_{cross} - 2l_{H,H'})$. Multiplying both sides by k, we obtain the equation:

$$k \cdot (\mathsf{LM}(H \cup H') - \mathsf{LM}(H)) = o_H \cdot (l_H + l_{H'} - l_{dup} + l_{H,H'}) - l_H \cdot (o_H + o_{H'} - l_{cross} - 2l_{H,H'})$$

Note that $o_H \geq l_{H_{dup},H'\setminus H} + l_{H,H'} \geq 0$ and $o_{H'} \geq l_{H_{dup},H\setminus H'} + l_{H,H'} \geq 0$, implying that the term $o_H + o_{H'} - l_{cross} - 2l_{H,H'}$ is always positive. Similarly, for LSM, we derive the following result:

$$\mathsf{LSM}(H \cup H') - \mathsf{LSM}(H) = \frac{l_H + l_{H'} - l_{dup} + l_{H,H'}}{o_H + o_{H'} - l_{cross} - 2l_{H,H'}} \cdot \frac{1}{(|H| + |H'| - |H_{dup}|)^\tau} - \frac{l_H}{o_H} \cdot \frac{1}{|H|^\tau}$$

$$k \cdot (\mathsf{LSM}(H \cup H') - \mathsf{LSM}(H)) = \frac{o_H \cdot (l_H + l_{H'} - l_{dup} + l_{H,H'})}{(|H| + |H'| - |H_{dup}|)^\tau} - \frac{(o_H + o_{H'} - l_{cross} - 2l_{H,H'}) \cdot l_H}{|H|^\tau}$$

To simplify, let $t_1 = o_H \cdot (l_H + l_{H'} - l_{dup} + l_{H,H'})$ and $t_2 = l_H \cdot (o_H + o_{H'} - l_{cross} - 2l_{H,H'})$. Hence, we get the following equations.

$$k \cdot (\mathsf{LSM}(H \cup H') - \mathsf{LSM}(H)) = \frac{t_1}{(|H| + |H'| - |H_{dup}|)^\tau} - \frac{t_2}{|H|^\tau}$$
$$k \cdot (\mathsf{LM}(H \cup H') - \mathsf{LM}(H)) = t_1 - t_2$$

Let $X = t_1 - t_2$ and $Y = \frac{t_1}{(|H|+|H'|-|H_{dup}|)^\tau} - \frac{t_2}{|H|^\tau}$. Showing $X > Y$ indicates that LSM is less prone to the free-rider effect.

- $X > 0$ and $Y > 0$: Both X and Y experience the free-rider effect. $X > Y$ implies that Y is less affected by the free-rider effect than X because small positive value indicates that it is close to overcoming the free-rider effect.
- $X < 0$ and $Y < 0$: Neither X nor Y suffers from the free-rider effect. However, if $X > Y$, it means that LSM imposes a larger penalty for the free-riders.
- $X > 0$ and $Y < 0$: LSM overcomes the free-rider effect, while LM suffers from it. Thus, we do not need to check it.
- $X < 0$ and $Y > 0$: LM does not suffer from the free-rider effect, while Y does. In the expression for Y, the denominator of the positive value (t_1) is larger than that of the negative value (t_2), indicating the positive value is significantly reduced compared to the negative one. This implies that if X is negative, Y must also be negative. Thus, we do not need to consider this case.

To prove $X > Y$, we multiply X by $\frac{1}{(|H|+|H'|-|H_{dup}|)^\tau}$, giving $X' = \frac{t_1-t_2}{(|H|+|H'|-|H_{dup}|)^\tau}$. Subtracting Y from X' yields:

$$X' - Y = \frac{-t_2}{(|H| + |H'| - |H_{dup}|)^\tau} + \frac{t_2}{|H|^\tau}$$

Since $\frac{1}{|H|^\tau} > \frac{1}{(|H|+|H'|-|H_{dup}|)^\tau}$, the positive term dominates the negative one, making $X' - Y > 0$. Thus, as $X > X'$, it follows that $X > Y$. Note that $X = (\mathsf{LM}(H \cup H') - \mathsf{LM}(H)) \cdot k$, $Y = (\mathsf{LSM}(H \cup H') - \mathsf{LSM}(H)) \cdot k$. Since $k > 0$, the inequality $X > Y$ implies that $\mathsf{LM}(H \cup H') - \mathsf{LM}(H) > \mathsf{LSM}(H \cup H') - \mathsf{LSM}(H)$. □

4.2.2 Parameter Threshold Under Weak Sense Assumption. As we have discussed in the previous proof, both LM and LSM suffer from local free-rider effect, though LSM is more robust due to the parameter τ. As τ increases, it favours smaller communities, thereby reducing the degree of suffering from the free-rider effect. We theoretically demonstrate that when $\tau > \log_{1+\frac{1}{h}}(\frac{h(h-1)}{2})$ and the identified community is a weak sense community, LSM no longer suffers from the local free-rider effect. Before proving this theorem, we introduce the concept of a weak sense community, which is central to ensuring the robustness of the identified community.

Definition 7. *(Weak sense community [31]). Given a graph $G = (V, E)$ and a set of nodes $S \subseteq V$, the induced subgraph $G[S]$ is called a **weak sense community** if the total weight of the internal edges is greater than or equal to the total weight of external edges. Formally, $G[S]$ is a weak sense community if $l_S \geq o_S$.*

Theorem 3. *LSM does not suffer from local free-rider effect when the identified community is a weak sense community and $\tau > \log_{1+\frac{1}{h}}(\frac{h(h-1)}{2})$.*

Proof. To guarantee that LSM does not suffer from the free-rider effect, it must satisfy the following inequality.

$$\mathsf{LSM}(H) > \mathsf{LSM}(H \cup H') \Rightarrow 0 > \frac{o_H \cdot (l_H + l_{H'} - l_{dup} + l_{H,H'})}{(|H| + |H'| - |H_{dup}|)^\tau} - \frac{(o_H + o_{H'} - l_{cross} - 2l_{H,H'}) \cdot l_H}{|H|^\tau}$$

To derive a lower bound on τ, we rewrite the following inequality in terms of τ.

$$\tau > \frac{\log o_H + \log l_{H\cup H'} - \log l_H - \log o_{H\cup H'}}{\log(|H| + |H'| - |H_{dup}|) - \log|H|} = \frac{-\log\frac{l_H}{o_H} + \log\frac{l_{H\cup H'}}{o_{H\cup H'}}}{\log(1 + \frac{|H'|-|H_{dup}|}{|H|})}$$

The inequality above provides an exact expression for the lower bound of the sketch threshold τ required to eliminate the free-rider effect in LSM. To make this condition more interpretable and

analytically tractable, we derive a simplified lower bound by separately bounding the numerator and the denominator separately.

To derive an upper bound on the numerator, we consider when the internal-to-external edge ratio of $H \cup H'$ significantly exceeds that of H. Since all terms $l_H, o_H, l_{H\cup H'}$, and $o_{H\cup H'}$ are always positive, the expression $\log(l_{H\cup H'}/o_{H\cup H'})$ achieves its maximum when the ratio $l_{H\cup H'}/o_{H\cup H'}$ is maximised, and $\log(l_H/o_H)$ reaches its minimum when l_H/o_H is minimised. The smallest possible value of l_H/o_H is 1, as the weak sense condition requires $l_H \geq o_H$. The maximum value for $l_{H\cup H'}/o_{H\cup H'}$ approaches $\frac{h(h-1)}{2}$. Since the LMSC problem enforces a size constraint such that $l \leq |H \cup H'| \leq h$, the maximum number of internal edges is $\frac{h(h-1)}{2}$. Therefore, we obtain the following equation :

$$\log(\frac{l_{H\cup H'}}{o_{H\cup H'}}) - \log(\frac{l_H}{o_H}) \leq \log(\frac{h(h-1)}{2}) - \log(1) \leq \log(\frac{h(h-1)}{2})$$

Next, we consider the minimum possible value of the denominator. Since $\frac{1}{h} \leq \frac{1}{|H|} \leq \frac{|H'|-|H_{dup}|}{|H|}$, it follows that :

$$\log(1 + \frac{1}{h}) \leq \log(1 + \frac{|H'| - |H_{dup}|}{|H|})$$

Thus, combining the bounds, we obtain:

$$\tau > \frac{\log(\frac{h(h-1)}{2})}{\log(1+\frac{1}{h})} > \frac{-\log\frac{l_H}{o_H} + \log\frac{l_{H\cup H'}}{o_{H\cup H'}}}{\log\,(1+\frac{|H'|-|H_{dup}|}{|H|})}$$

As a result, to ensure that it does not suffer from the local free-rider effect, τ must satisfy the lower bound:

$$\tau > \log_{1+\frac{1}{h}}(\frac{h(h-1)}{2})$$

□

4.3 Properties of LSM

We now discuss two structural properties of the LSM : (1) monotonicity and (2) submodularity.

Theorem 4. *The LSM is non-monotone; that is, there exists a set $A \subseteq V$ and a node $v \notin A$ such that $LSM(A \cup \{v\}) < LSM(A)$.*

Proof. We prove by counterexample. Suppose LSM is monotone, i.e., $\mathrm{LSM}(A \cup \{v\}) \geq \mathrm{LSM}(A)$. Consider a graph $G = (V, E)$ where every edge weights are 1, consisting of two cliques: a 4-node clique and a 3-node clique. One node v in 3-node clique is connected to a single node u in 4-node clique via an inter-clique edge. Let A denote the node set of the 4-node clique, and let v be the node in 3-node clique that is connected to $u \in A$. Assume $\tau = 1$. In this setting, the internal edge weight of A is $l_A = 6$, and the outgoing edge weight is $o_A = 1$. Therefore, $\mathrm{LSM}(A) = \frac{l_A}{o_A} \cdot \frac{1}{|A|^\tau} = \frac{6}{1} \cdot \frac{1}{4^1} = 1.5$. Now consider adding v to A. The internal edge weight becomes $l_{A\cup\{v\}} = 7$, and the outgoing edge weight becomes $o_{A\cup\{v\}} = 2$. Hence, $\mathrm{LSM}(A') = \frac{l_{A'}}{o_{A'}} \cdot \frac{1}{|A'|^\tau} = \frac{7}{2} \cdot \frac{1}{5^1} = 0.7$. Since $\mathrm{LSM}(A \cup \{v\}) < \mathrm{LSM}(A)$, adding v to A decreases the LSM value, which contradicts the assumption that LSM is monotone. Therefore, LSM is non-monotone. □

Theorem 5. *The LSM is non-submodular; that is, there exist sets $A \subseteq B \subseteq V$ and a node $v \notin B$ such that $LSM(A \cup \{v\}) - LSM(A) < LSM(B \cup \{v\}) - LSM(B)$.*

Proof. Suppose LSM is submodular. Consider a graph with all edge weights set to 1, consisting of a 5-node clique and two additional nodes $\{x, y\}$. Let A be a set of four nodes from the 5-clique, and let B is the 5-clique node set. Suppose x is connected to one node in A and one node in $B \setminus A$,

Algorithm 1: IGA(G, h, l, Q, τ)

$i \leftarrow 1, C_1 \leftarrow \texttt{getSeed}(G, Q)$;
for $|C_i| \leq h - 1$ **do**
  $v \leftarrow \arg\max_{w \in N(C_i, G)} \Delta\text{LSM}(G, C \cup \{w\}, \tau)$;
  $C_{i+1} \leftarrow C_i \cup \{v\}$;
  $i \leftarrow i + 1$;
return $\arg\max_{\substack{C \in \{C_1, C_2, \cdots, C_i\}, \\ l \leq |C| \leq h}} \text{LSM}(G, C, \tau)$

while y is connected only to a node in A. Let $\tau = 1$. For set A, the internal edge weight and external connectivity are $l_A = 6$ and $o_A = 6$, respectively, with $|A| = 4$. Hence, $\text{LSM}(A) = 0.25$. Then for a set B, $l_B = 10$, $o_B = 3$, and $|B| = 5$. Therefore, $\text{LSM}(B) = \frac{2}{3}$. Now consider adding node x to each set. In the case of $A \cup \{x\}$, $l_{A\cup\{x\}} = 7$, $o_{A\cup\{x\}} = 6$, and $|A \cup \{x\}| = 5$, leading to $\text{LSM}(A \cup \{x\}) = \frac{7}{30}$. The corresponding marginal gain is therefore $\Delta_A = \text{LSM}(A \cup \{x\}) - \text{LSM}(A) = \frac{7}{30} - 0.25 \approx -0.017$. For $B \cup \{x\}$, $l_{B\cup\{x\}} = 12$, $o_{B\cup\{x\}} = 1$, and $|B \cup \{x\}| = 6$. Thus, $\text{LSM}(B \cup \{x\}) = 2.0$, and $\Delta_B = \text{LSM}(B \cup \{x\}) - \text{LSM}(B) = 2.0 - \frac{2}{3} \approx 1.334$. Consequently, $\Delta_A < \Delta_B$, suggesting that the submodular property does not hold. Therefore, LSM is non-submodular. □

5 INCREMENTAL GREEDY ALGORITHM(IGA)

Overview. The Incremental Greedy Algorithm (IGA) aims to identify a query-centric community by iteratively expanding from the query nodes according to the Local Sketch Modularity (LSM) value. It serves as a baseline local optimiser for the LSM objective, providing an intuitive greedy expansion procedure that incrementally improves community cohesiveness under the size constraint.

Procedure. Algorithm 1 describes the procedure of IGA. The algorithm starts with a seed community C_1, comprising the query node (for a single query) or a connected component containing all query nodes (see below for multiple queries). At each step, the algorithm considers the neighbours of the current community C_i and selects the node that maximises the gain in LSM (ΔLSM). This process repeats until the size constraint is reached. Finally, among all candidate communities generated during the process, the one with the highest LSM (within $[l, h]$) is returned.

Time complexity. The overall time complexity of IGA is $O(S + h(|V| + d_{\max}))$, which in the worst case simplifies to $O(S + h|V|)$, where S is the cost of obtaining the seed containing the query nodes and $d_{\max}$ is the maximum degree. Each iteration scans the current boundary $N(C)$ (at most $|V| - 1$ candidates) and computes ΔLSM in $O(1)$ per candidate, as $\deg(v)$ and and $l_{v,C}$, the total weight of edges between v and C, are maintained for each candidate v. After adding a node u, we update $l_{v,C}$ only for $v \in N(u)$ in $O(\deg(u))$ time.

Limitation of IGA. Since IGA greedily adds nodes maximising ΔLSM at each step, it may become trapped in local optima, failing to improve LSM further. To address this, we introduce a group-wise extension in the next section. Empirical analysis of this limitation is provided in Section 7.

Handling multiple query nodes. Handling multiple query nodes in community search is challenging, especially for bottom-up approaches [21]. Prior work has used modularity-aware shortest paths [21] or Steiner-tree methods [2]. A central challenge is forming an initial structure that connects all query nodes while achieving a strong LSM score—balancing connectivity with optimising internal and external edges. We adopt Mehlhorn's Steiner-tree algorithm [27], which provides a 2-approximation in $O(|E| + |V| \log |V|)$ time. Since a tree of n nodes has only $n-1$ internal edges, enhancing internal connectivity is limited. We therefore focus on reducing external edges to enhance the LSM score. To this end, we reweight each edge as $w(u, v) = \max(\deg(u), \deg(v))$, penalising high-degree nodes, which typically increase o_c. This guides the tree toward lower-degree paths and

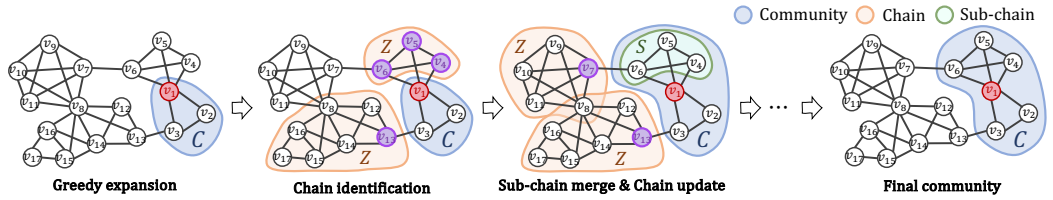


Fig. 2. Overall SMA framework

improves community quality. Empirically, this degree-based re-weighting yields boundary-robust seeds by avoiding border nodes that often act as cross-community bridges [34].

6 SUB-CHAIN MERGE ALGORITHM(SMA)

6.1 Overview

The Sub-chain Merge Algorithm (SMA) is a bottom-up framework designed to overcome the local-optimality limitation of IGA. While IGA expands a community by adding one node at a time, SMA performs group-wise expansion by merging cohesive structures, termed *chains*. Each chain is a locally connected sequence of nodes adjacent to the current community and is evaluated using Chain-based Local Sketch Modularity (CLSM), which captures internal connectivity and boundary strength without global graph information. The overall pseudocode of SMA is provided in Algorithm 2. In [19], the detailed procedures of the SMA subroutines—greedy expansion, chain identification, sub-chain merging, and chain updating—are presented in Appendices B, C, D, and E.

Figure 2 illustrates the workflow of SMA. Starting from a seed community, the algorithm forms an initial core via local greedy expansion, then iteratively identifies candidate chains, merges candidate sub-chains that maximise LSM improvement, and updates only the affected chains. This process continues until the size constraint $[l, h]$ is satisfied, returning the community with the highest LSM value encountered. This strategy alleviates local-optimum traps, improves efficiency through localised updates, and ensures that the resulting community is cohesive and size-constrained.

6.2 Greedy Expansion Procedure

In the first step, SMA identifies a "core" community[1] starting from the query nodes, using a strategy similar to the Incremental Greedy Algorithm in Section 5. The key difference is as follows: while IGA adds a node at a time based on maximum ΔLSM even when this gain is negative, SMA expands the community by adding a node only when it provides positive and maximal gain in each iteration. This expansion continues until no further gain is possible or the size constraint is reached.

6.3 Chain Identification Procedure

After forming the core community, the algorithm proceeds to explore neighbouring cohesive structures, referred to as "chains". Unlike IGA, which expands the community by adding individual nodes, this approach seeks to merge sets of nodes that together yield a greater increase in the LSM objective. A chain is defined as a sequence of nodes adjacent to the community, identified using a relaxed criterion termed Chain-based LSM (CLSM), which evaluates each chain based on its internal structure and connections to the current community. The formal definition is as follows.

Definition 8. *(Chain.) Given a graph $G = (V, E)$, a current community $C \subset V$, and a maximum community size constraint h, a sequence of nodes $Z = \{v_0, v_1, \cdots, v_j\}$ is defined as a chain if it satisfies the following conditions:*

[1] a connected subgraph containing all the query nodes and formed by greedy expansion from the initial seed.

(1) ***Connectivity:*** *Any subset* $S = \{v_0, v_1, \ldots, v_i\}$ *(*$0 \le i \le j$*) forms a connected subgraph in* G*;*
(2) ***Adjacency:*** $v_0 \in Z$ *is connected to at least one node in* C*, i.e.,* $v_0 \in N(C, G) \setminus C$*;*
(3) ***Disjointness:*** $Z \cap C = \emptyset$*;*
(4) ***Size constraint:*** $|Z \cup C| \le h$*.*

Note that identifying and updating chains directly based on the LSM measure is challenging, as it requires tracking not only the internal connectivity of the chain (l_Z, o_Z), but also the state of the community (l_C, o_C, $|C|^\tau$, $l_{C,Z}$). Since any modification affects all the chains simultaneously, we design CLSM, a relaxed objective that evaluates each chain solely on its internal structure and its boundary with the community.

Definition 9. *(Chain-based LSM) Given a graph* $G = (V, E)$*, a current community* C*, and a chain* Z *of the current community, the Chain-based LSM (CLSM) is defined as follows:*

$$CLSM(C, Z) = \frac{l_Z + l_{C,Z}}{o_Z - l_{C,Z}} \tag{1}$$

where $l_{C,Z}$ *denotes the total weight of edges between* C *and* Z*.*

This formulation encourages chains with strong internal connectivity (via l_Z) and significant interaction with the current community (via $l_{C,Z}$), while penalising those with many external connections to the rest of the graph (via o_Z). Thus, CLSM enables efficient and independent updates of chains as it relies only on local information.

Next, we describe how to get a set of chains given a current community. Starting from a neighbouring node—referred to as "pivot"—we repeatedly add nodes that yield the maximum increase in CLSM. If multiple candidates have equal improvements, the node closer to the pivot is selected. This expansion continues until no further improvements are possible or the maximum size is reached.

Each chain maintains its insertion order, which is preserved to extract valid sub-chains for merge operation. Note that chain identification proceeds only as long as each incremental addition results in a non-negative marginal gain in CLSM, ensuring that only chains with positive improvement are considered as valid.

6.4 Sub-chain Merge Procedure

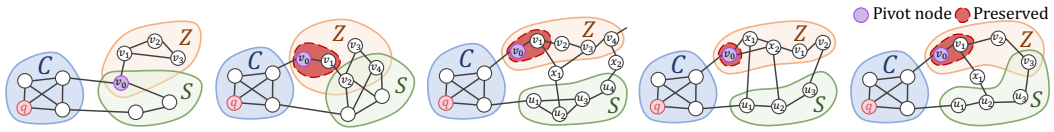
After identifying candidate chains, we initiate the merge process. Although CLSM is a variant of LSM, merging a chain with the current community does not always increase the LSM score. To ensure that each merge step is beneficial, we examine only consecutive prefixes of each chain, referred to as sub-chains. This restriction guarantees connectivity and order preservation, and reduces unnecessary computation by excluding disconnected or unordered subsets.

Definition 10. *(Sub-chain.) Given a chain* $Z = \{v_0, v_1, \cdots, v_j\}$*, a sub-chain of* S *is a subset of the chain* Z *such that* $S = \{v_0, v_1, \cdots, v_i\}$ *where* $0 \le i \le j$*.*

Thus, each chain with $|Z|$ nodes yields at most $|Z|$ valid sub-chains. We scan all such sub-chains to find the one whose addition results in the largest increase in LSM. This sub-chain is merged with the community. If multiple candidate sub-chains have the same gain, this process prefers the one closer to the query node, by selecting the shorter sub-chain to maximise flexibility for future expansion. After merging, only chains affected by the update are recomputed, enabling efficient iteration until the size constraint is satisfied.

6.5 Chain Update Procedure

Merging the current community with the optimal sub-chain may disrupt the structure of neighbouring chains, potentially violating the constraints stipulated in Definition 8. Recomputing all



(a) Case #1 (b) Case #2 (c) Case #3 (d) Case #4 (e) Case #5

Fig. 3. Affected chains

chains after each merge is computationally prohibitive, particularly in large-scale networks. We therefore introduce a selective update strategy that updates only those chains whose composition may be directly affected by the merge operation, improving efficiency significantly.

We describe a set of rules to selectively update chains following a community merge. Each rule addresses a structural constraint or neighbourhood change introduced during the merge process.

Update for condition #1 (Size constraint exceeded). After merging the community C with a sub-chain S, the size of each chain Z, with respect to the updated community, may violate the constraint $|C' \cup Z| \leq h$, i.e., $|Z| > h - |C'|$. Then, the chain must be truncated.

Update Rule 1. *If the chain $|Z| > h - |C'|$, it is truncated by retaining only the first $h - |C'|$ nodes: $Z = \{v_0, v_1, \ldots, v_{h-|C'|-1}\}$.*

Example 3. *Suppose $h = 8$. In Figure 3e, merging the current community C with sub-chain S results in an updated community of size $|C'| = 7$. Since the maximum number of additional nodes that can be merged is $h - |C'| = 1$, chain $Z = \{v_0, v_1, v_2, v_3, v_4\}$ is truncated to $\{v_0\}$.*

Update for condition #2 (Disjointness violated). A chain must be disjoint from the current community. After merging, however, some nodes previously in chains may now belong to C', potentially breaching the disjointness constraint. We distinguish two sub-cases:

(1) Pivot node included in S: If the pivot node v_0 of chain Z is included in the merged sub-chain S, the chain Z is no longer valid and must be removed.
(2) Non-pivot node included in S: If $v_0 \notin S$, but some other nodes in chain are included in S, then Z is truncated.

The update rule and example for sub-case 1 are as follows:

Update Rule 2. *If the pivot node v_0 of chain Z is included in the merged sub-chains S (i.e, $v_0 \in S$), remove the chain.*

Example 4. *As depicted in Figure 3a, if pivot node v_0 is included in merged sub-chain S, the chain $Z = \{v_0, v_1, v_2, v_3\}$ is removed in its entirety.*

For sub-case 2, we present the following update rule and example:

Update Rule 3. *Let $Z = \{v_0, v_1, \ldots, v_k\}$ be a chain that the pivot node $v_0 \notin S$, and $Z \cap S \neq \emptyset$. Let i be the smallest index such that $v_i \in S$. Then, truncate the chain to $Z = \{v_0, \ldots, v_{i-1}\}$.*

Example 5. *Figure 3b shows a scenario where nodes v_2 and v_4 in Z are also in S. The smallest index at which such an overlap occurs is $i = 2$, so only $\{v_0, v_1\}$ are kept in Z.*

Update for Condition #3 (CLSM scores changed). After merging, the updated community may alter the CLSM scores of neighbouring nodes, potentially requiring the order of nodes within a chain to be updated or rearranged. Specifically:

(1) If there exists a node x—not belonging to Z or C', but adjacent to both Z and S—that offers a higher CLSM gain than the node originally selected during the construction of Z, then the chain Z must be updated to reflect the new optimal ordering.

(2) If a node $v \in Z$ is directly connected to S, the chain Z must be updated, as this connection may change the optimal sequence or composition of nodes in Z.

These two sub-cases are addressed by the following update rule.

Update Rule 4. *Let $Z = \{v_0, v_1, \ldots, v_k\}$. If there exists a node $x \in (N(Z, G) \cap N(S, G)) \setminus C'$, let i be the smallest index such that v_i is adjacent to x. Then, truncate the chain to $Z = \{v_0, \ldots, v_i\}$.*

The following examples illustrate each case handled by the update rule 4. We first show sub-case 1, and the rule is applied as follows.

Example 6. *Figure 3c illustrates this rule. Suppose $Z = \{v_0, v_1, \ldots, v_4\}$ and, after merging, an external node x_1 (not in Z) is adjacent to both S and v_1. Since x_1 may provide a higher CLSM gain than v_2, the chain is truncated at v_1, and only $\{v_0, v_1\}$ are preserved. Similarly, Figure 3d shows the case where $x \in Z$. As v_2 may offer a higher CLSM gain than v_1, so the chain is truncated to only $\{v_0\}$.*

The next example corresponds to sub-case 2.

Example 7. *Figure 3d demonstrates this case: x_1, x_2 in Z becomes newly adjacent to S, affecting the next node choice after v_0. As a result, the chain is truncated to $\{v_0\}$.*

After applying these rules, the preserved portion of each chain is rebuilt using the original greedy strategy of Chain identification. Starting from the last preserved node, each next node is selected based on the CLSM gain among neighbours of the current chain.

Update for New Pivots. Following the merge, nodes that become newly adjacent to the updated community C' are used as pivots to construct new chains.

Update Rule 5. *Let $\mathcal{P}$ denote the set of pivot nodes from previously constructed chains. For each $v \in N(S, G) \setminus (C' \cup \mathcal{P})$, initiate a new chain with v as the pivot.*

Appendix G in [19] formally proves that the local update rules suffice and yield the same canonical chain state as a full global reconstruction, thus ensuring the correctness of the update procedure.

6.6 Priority of Chain Update Rules

A chain may simultaneously satisfy multiple update conditions. In such cases, we apply the update rules in the following priority order to ensure consistency and correctness:

- *Priority 1. Removal (Update Rule 2)*: If the pivot node is included in S, the chain is removed—this takes highest priority.
- *Priority 2. Size constraint truncation (Update Rule 1):* If the chain exceeds the size constraint, it is truncated accordingly.
- *Priority 3. Other truncations (Update Rules 3, and 4):* For all remaining cases, the chain is preserved only up to the smallest index specified by the applicable rule(s), ensuring all structural constraints are satisfied.

Example 8. *In Figure 3e, chain Z satisfies both Update Rule 2 and 4. Rule 2 would retain $\{v_0, v_1, v_2\}$, but Rule 4 dictates that only $\{v_0, v_1\}$ should be kept. Thus, we preserve only $\{v_0, v_1\}$.*

Appendix H, as detailed in [19], further proves that the prescribed priority order of the update rules is sufficient to ensure correctness when multiple conditions hold simultaneously.

In summary, only chains directly affected by the merge—through overlap or a change in their optimal order—require updates; all other chains retain their structure and CLSM scores unchanged. This selective update ensures computational efficiency by avoiding unnecessary recomputation.

Algorithm 2: SMA(G, h, l, Q, τ)

```
1  i ← 1, Z ← {}, S, P ← ∅;
2  C_i ← getSeed(G, Q) ;                                   // Step #1
3  C_i ← greedyExpansion(G, C_i, τ, h);                    // Step #2
4  Z, P ← findChains(G, C_i, h);                           // Step #3
5  while |C_i| < h do
6  |   S ← getSubchain(G, C_i, Z, h,τ) ;                  // Step #4
7  |   C_{i+1} ← C_i ∪ S;
8  |   i ← i + 1;
9  |   Z, P ←update(G, C_i, S, Z, P, h) ;                 // Step #5
10 return arg max_{C∈{C_1,C_2,···,C_i}, l≤|C|≤h} LSM(G, C, τ)
```

6.7 Overall Procedure

Algorithm 2 presents the full SMA framework, comprising five main stages. The procedure begins by identifying a seed community containing the query nodes (Step 1) and incrementally expanding it with a greedy strategy (Step 2). Chains are then constructed in the neighbouring region of the current community (Step 3). At each iteration, the most beneficial sub-chain—i.e., the one yielding the greatest improvement in CLSM score—is selected and merged (Step 4), after which only the affected chains are locally updated (Step 5). This process continues until the community size reaches the upper limit h, at which point the algorithm returns the highest-scoring community found within the specified constraints.

Example 9. *Consider the example illustrated in Figure 2. Let $\tau = 1$, the query node be v_1, and the size constraint be $[l, h] = [6, 15]$. The algorithm initiates greedy expansion from v_1, adding v_2 and v_3 to form $C = \{v_1, v_2, v_3\}$. In the chain identification step, four pivots adjacent to C—v_4, v_5, v_6, and v_{13}—are used to construct valid chains. Among the sub-chains of $Z = \{v_4, v_5, v_6\}$, the full chain $\{v_4, v_5, v_6\}$ provides the largest LSM gain and is merged, yielding $C = \{v_1, v_2, v_3, v_4, v_5, v_6\}$. The chain update then removes all chains containing v_5 or v_6, and a new pivot v_7 allows the construction of a further chain. This process repeats until the community size reaches $h = 15$, at which point the community with the highest LSM score within the constraints is returned.*

Time complexity. The computational complexity of SMA stems from the following components:

- **Seed community (Step #1):** $O(|E| + |V| \log |V|)$ - to obtain seed community, Mehlhorn's Steiner-tree algorithm [27] is applied.
- **Greedy expansion (Step #2):** $O(h|V|)$ — up to h iterations, each involving $O(|V|)$ time to compute ΔLSM for all neighbours.
- **Chain identification (Step #3):** $O(h|V|^2)$ — considers $O(|V|)$ pivot candidates, each requiring $O(h|V|)$ time to construct a chain.
- **Sub-chain merging (Step #4):** $O(h|V|^2)$ — for each of the $O(|V|)$ chains, up to h sub-chains, $O(|V|)$ time to compute LSM.
- **Chain updating (Step #5):** $O(h|V|^2)$ — up to $O(|V|)$ chains may require updating, with each involving up to $O(h|V|)$ work.

Thus, the time complexity of SMA is $O(h^2|V|^2)$, where h is the upper limit on community size.

7 EXPERIMENTS

In this section, we conduct a comprehensive evaluation of our algorithms on a variety of real-world networks. We aim to evaluate both the effectiveness and scalability of the proposed methods under

Table 3. Real-world datasets with ground-truth

Type	Dataset	$\|V\|$	$\|E\|$	$\|C\|$
Small-scaled	Karate	34	78	2
	Dolphin	62	159	2
	Polbooks	105	441	3
	Railway	301	1224	18
Large-scaled	Amazon*	334, 863	925, 872	75, 149
	DBLP*	317, 080	1, 049, 866	13, 477
	YouTube*	1, 134, 890	2, 987, 624	8, 385
	LiveJournal*	3, 977, 962	34, 681, 189	287, 512

* Dataset with overlapping ground-truth communities.

diverse conditions. All experiments were performed on a dedicated server running Ubuntu 22.04, equipped with 265GB of RAM and an Intel Xeon Gold 6342 CPU (2.80GHz), ensuring a consistent and controlled environment for fair comparison.

Datasets. Table 3 presents the key statistics of the real-world datasets used in our experiments, where $|C|$ denotes the number of ground-truth communities, d is the average degree, and $d_{\max}$ is the maximum degree. All datasets are publicly available [4, 26, 32, 41, 44].

Baseline. We compare our algorithms with existing size-constrained community search models, as detailed in Section 8. We report results only if the algorithms successfully return a solution within 24 hours. The compared algorithms are summarised as follows:

- GreedyD [35]: a heuristic algorithm that maximises the minimum degree within a subgraph satisfying the size constraint h and the distance constraint d from the query nodes.
- SCS [43]: an exact k-core based size-constrained community search algorithm using a branch-reduce-and-bound framework.
- STCS [45]: an exact size-constrained community search algorithm based on maximising the minimum node support, utilising a branch-and-bound framework with truss-based cohesiveness.

As our problem explicitly considers size constraints, we compare it with models that also handle size-bounded communities, including two-sided and one-sided approaches. Following the recommendations of [5], we omit the SCkT [23] baseline from our main comparison, as it requires dataset-dependent manual tuning of the k parameter, which may result in inconsistent or unfair evaluation. Instead, we include STCS [45] as a representative truss-based baseline. For GreedyD, we set d to the network diameter so that the distance constraint does not restrict the search space. GreedyD failed to return results within 24 hours on large-sized networks due to its computational cost, and was therefore excluded from those cases.

Parameter setting. Each algorithm is evaluated using its standard query type; to ensure fair comparison, all results reported in this section correspond to the single-query scenario. Experiments on multiple-query settings are presented separately. Unless otherwise specified, query nodes are sampled uniformly at random, and the size constraints l and h are fixed to $[1, 20]$. The sketch threshold parameter τ is set to 1 throughout (see Appendix F in [19] for details). Baseline algorithms are configured identically for direct comparability. To mitigate the effects of query node selection, all results are averaged over 20 independent trials unless noted otherwise.

Evaluation metrics. We evaluate the quality of identified communities using F1-score and Adjusted Rand Index (ARI), with higher values indicating closer alignment to the ground-truth. Following [21], community search is cast as a binary classification problem in which nodes within the retrieved community are treated as positives. Accuracy is computed by comparing the returned community

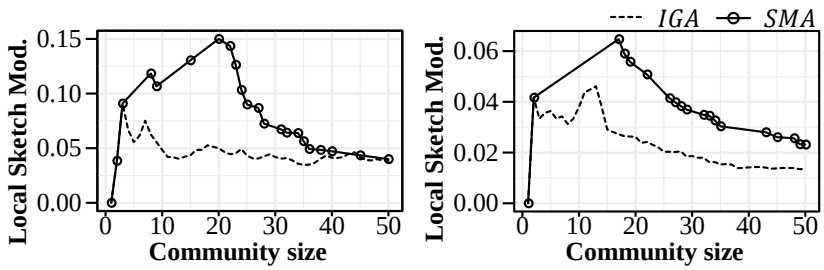
with the corresponding ground-truth; in cases of overlapping ground-truths, we take the maximum score across candidates.

7.1 Experimental Setup

We conduct a comprehensive set of experiments to evaluate the impact of our theoretical findings and the performance of the proposed methods. Our study addresses the following key questions:

(1) **EQ1. LSM trend comparative analysis:** How does the value of LSM change for the IGA and SMA algorithms?
(2) **EQ2. Accuracy of identified community:** How accurately do the proposed algorithms identify ground-truth communities?
(3) **EQ3. Effect of the user parameter τ:** How does varying the parameter τ affect the accuracy and the size of the community?
(4) **EQ4. Multiple query nodes:** How efficiently does our algorithm perform when handling multiple query nodes?
(5) **EQ5. Efficiency of the chain updating procedure:** How effective is the chain updating procedure in computation time?
(6) **EQ6. Efficiency with different query types:** How efficiently does our algorithm perform across various types of query nodes?
(7) **EQ7. Scalability:** How does our algorithm scale with network sizes and size constraints?
(8) **EQ8. Limitation of Greedy Expansion in IGA:** Why node-wise selection is insufficient for cohesive community recovery?
(9) **EQ9. Case study:** We conduct a case study on the American football dataset by identifying a size-constrained community centred around a target team.

7.2 Experimental Result

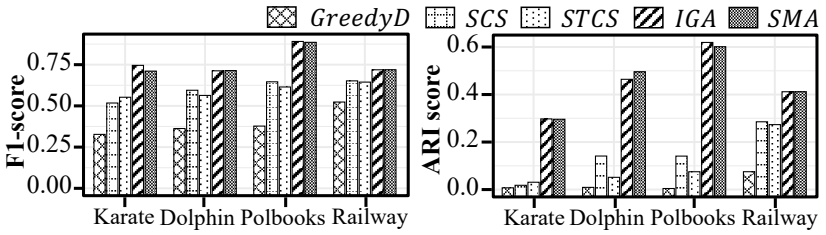


(a) DBLP (b) YouTube

Fig. 4. EQ1. LSM trend comparative analysis

EQ1. LSM trend comparative analysis. We examine how LSM scores evolve as nodes are incrementally added to the community under two algorithms: IGA and SMA. Both algorithms are executed with the default sketch threshold τ, and a query node is selected at random. To better capture the changes in LSM with increasing community size, we set the size constraint to $[1, 50]$ instead of the default. The LSM trends observed on the DBLP and YouTube datasets are shown in Figures 4a and 4b, respectively. Across both datasets, the initial LSM scores increase sharply as nodes are added. Subsequently, IGA exhibits a gradual decline in LSM, suggesting that it becomes trapped in a local optimum. In contrast, SMA incorporates multiple nodes in a single iteration, resulting in a more rapid increase in LSM scores. After reaching a peak, the LSM scores decrease, which aligns with the expected behaviour of the algorithms.

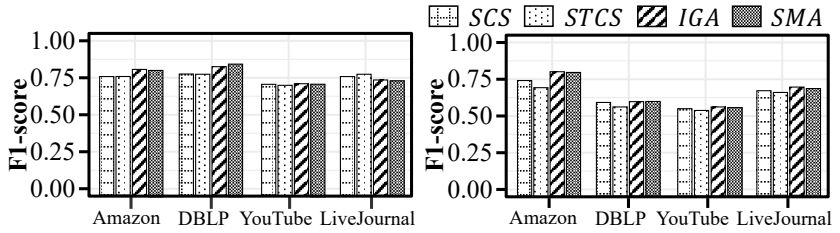
EQ2. Accuracy of identified community. We evaluate the accuracy of the algorithms in comparison to SOTA algorithms. To control for size-scale confounding, we set the size constraint based on the size of the ground-truth community. In detail, the size constraint is set as $[l, h] = [\frac{|C|}{2}, \min(\frac{3|C|}{2}, |V|)]$, where $|C|$ denotes the size of the ground-truth community.



(a) F1-score (b) ARI score

Fig. 5. EQ2-1. Accuracy with small-scaled datasets

EQ2-1. Small-scaled networks. Figure 5a and Figure 5b show the accuracy on small-scale networks in terms of F1-score and ARI, respectively. Query nodes were randomly selected for each case. These datasets provide distinct ground-truth communities. Both IGA and SMA consistently yield high-quality communities, significantly outperforming state-of-the-art baselines. This advantage arises from the fact that many existing methods are tightly bound to rigid structural constraints (e.g., k-core or k-truss), which often fail to reflect the semantics of real-world communities.

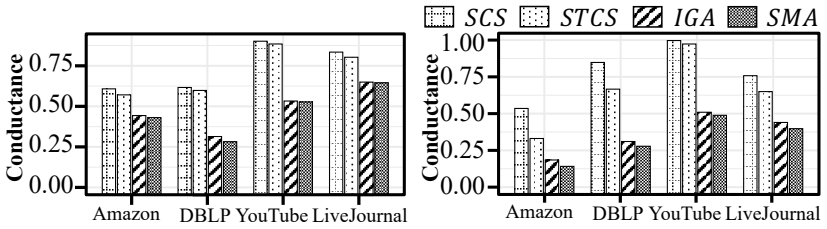


(a) Small-sized communities (b) Large-sized communities

Fig. 6. EQ2-2. Accuracy with large-scaled datasets

EQ2-2. Large-scaled networks. In large-scaled networks, we categorise the ground-truth communities into two groups based on their size: small (between 2 and 25 nodes) and large (between 26 and 50 nodes). We randomly pick a community belongs to that size, then randomly pick a set of query nodes. As shown in Figure 6, both IGA and SMA yield reasonable performance across datasets. However, in the YouTube and LiveJournal datasets, the difference in accuracy among competing methods is marginal. We consider that this is due to the overlapping nature of communities in these networks. In these large-sized networks, many nodes belong to multiple communities, making it difficult to isolate a single cohesive subgraph. Moreover, the presence of numerous inter-community connections hinders the identification of a structurally distinct community, highlighting the inherent limitations of query-centric approaches in overlapping environments.

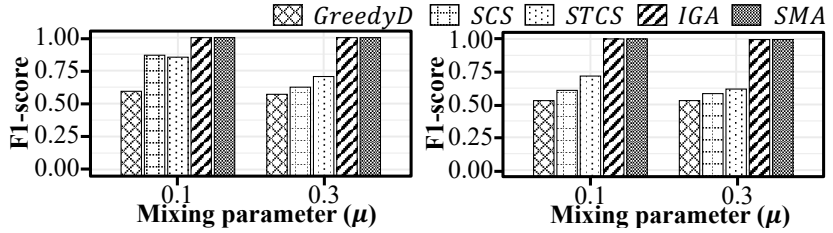
Although the accuracy differences among algorithms are small in overlapping networks such as YouTube and LiveJournal, we further examined the structural quality of the identified subgraphs using conductance, which captures both internal cohesion and external separability. As shown in Figure 7, SMA achieves the lowest conductance, followed by IGA, since both optimise LSM,



(a) Small-sized communities (b) Large-sized communities

Fig. 7. EQ2-2. Conductance with large-scaled datasets

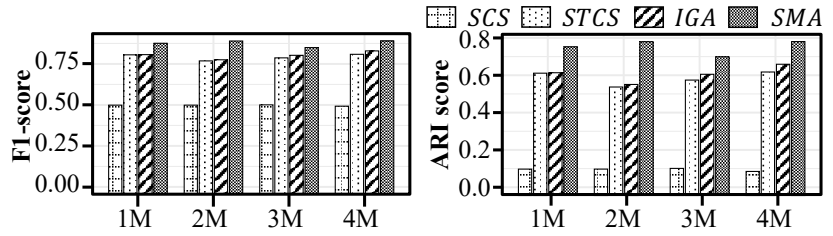
which jointly considers internal cohesion and external separation. In contrast, SCS and STCS yield higher conductance despite similar accuracy, as both methods rely primarily on internal structural constraints and do not account for external connections. Overall, SMA and IGA produce high-quality communities with dense internal and sparse external connections.



(a) Small-sized community (b) Large-sized community

Fig. 8. EQ2-3. Accuracy with synthetic datasets

EQ2-3. Synthetic networks with varying mixing parameter. We further validate our methods on synthetic datasets generated using the LFR benchmark [22], which is designed to simulate networks with well-separated community structure. In this experiment, we evaluate performance under two mixing parameter settings: 0.1 and 0.3. We exclude the case of 0.5, since it is known to significantly interfere with community identification due to excessive inter-community connections [42]. Figure 8 presents the results separately for small-sized and large-sized communities. Consistent with our findings (e.g., EQ2-1), IGA and SMA achieve the highest accuracy across all settings. These results confirm that our proposed methods are effective when the community structure is distinct and well-defined.

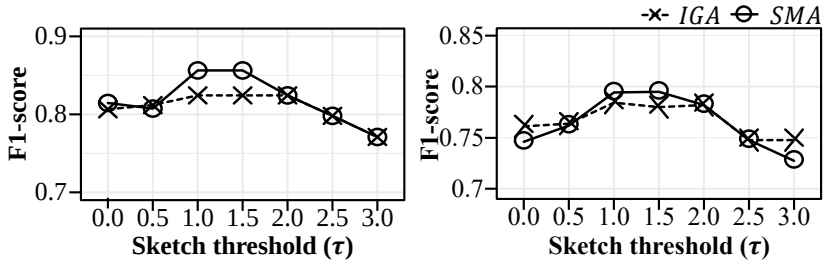


(a) F1-score (b) ARI score

Fig. 9. EQ2-4. Accuracy with large-scale synthetic datasets.

EQ2-4. Large-scale synthetic networks. As observed in **EQ2-2**, overlapping communities in real-world networks exhibit less notable performance differences among algorithms. To further evaluate the algorithms, we employ large-scale non-overlapping synthetic networks generated by the LFR benchmark, ranging from 1M to 4M nodes. Figure 9 reports the F1 and ARI scores on large-scale non-overlapping synthetic networks. It shows that SMA consistently achieves the highest accuracy, followed by IGA, demonstrating that both algorithms remain effective even in million-scale non-overlapping settings.

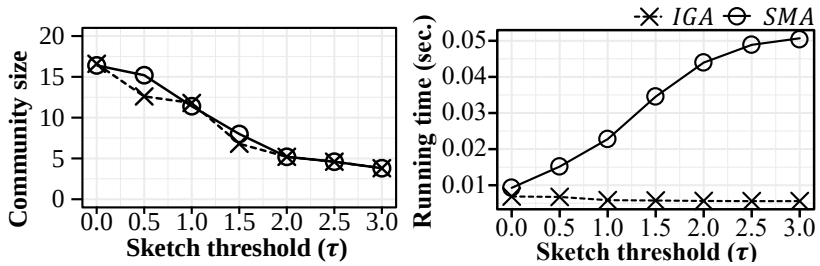
EQ3. Effect on user parameter τ. We evaluate the impact of the parameter τ on the quality and efficiency of community search results. Specifically, we examine how variations in τ influence the F1-score, the size of the resultant community, and the overall running time.



(a) Small-sized community (b) Large-sized community

Fig. 10. EQ3-1. Accuracy under varying τ (Amazon dataset)

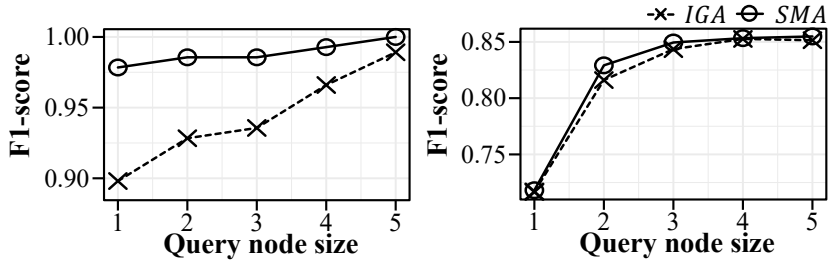
Figure 10 presents the F1-score performance across different values of τ for both small and large-sized communities in the Amazon dataset. The results indicate that the F1-score tends to increase initially with τ, reaching its peak around $\tau = 1.0$ to 1.5, and then declines as τ grows further. This trend is consistently observed for both small and moderate-sized community settings, suggesting that a moderate sketch threshold achieves the best balance between internal density and external separability. Excessively low values of τ lead to overly inclusive communities, while high values impose overly strict penalties, potentially omitting relevant nodes.



(a) Resultant subgraph size (b) Running time by τ

Fig. 11. EQ3-2. Size and time varying τ (Amazon dataset)

We examine the effect of the sketch threshold τ on both community size and running time under the default size constraint. As shown in Figure 11a, increasing τ leads to smaller communities and more compact solutions. SMA is thus sensitive to τ, with higher values yielding smaller, but potentially more numerous, sub-chains and longer running times (see Figure 11b). Overall, τ is a key parameter for balancing accuracy, compactness, and efficiency, with values in the range 1.0–1.5 working well in practice.



(a) Small-scaled graph (Dolphin) (b) Large-scaled graph (Amazon)

Fig. 12. EQ4. Multiple query nodes

EQ4. Multiple query nodes. We examine the impact of varying the number of query nodes on community search performance in both small- and large-scaled networks. For this experiment, we randomly sample ground-truth communities and vary the number of query nodes from 1 to 5. Figure 12a and Figure 12b present the F1-scores on the Dolphin and Amazon datasets, respectively. As the number of query nodes increases, both IGA and SMA show consistent improvements in accuracy, indicating that richer query input helps guide the search process more effectively. In particular, SMA achieves near-perfect F1-scores on the Dolphin graph, while both methods yield steady gains on the Amazon dataset. These results highlight that providing multiple query nodes enhances the structural cues, leading to more accurate community identification.

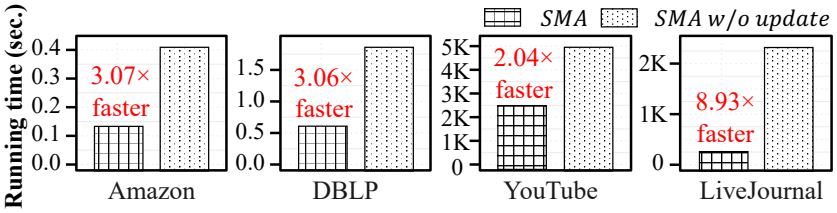


Fig. 13. EQ5. Efficiency of the updating chain procedure

EQ5. Efficiency of the chain updating procedure. To evaluate the efficiency of our chain updating strategy, we compare SMA with SMA w/o update. Here, SMA w/o update refers to the variant without the chain updating procedure, which recomputes all chains from scratch after each merge operation. The size constraint is set to $[1, 50]$, and τ is set to default. The running times across large-scaled datasets are shown in Figure 13. It can be observed that SMA with the chain updating technique consistently achieves lower running times across all datasets. Specifically, on the Amazon and LiveJournal datasets, SMA achieves 3.07× and 8.93× faster running time, respectively, compared to SMA w/o update. This efficiency gain becomes more remarkable as the graph size increases, highlighting the scalability of the selective update mechanism. These results demonstrate that local chain updates are highly effective in reducing redundant computations in large-scale scenarios.

EQ6. Different query types. We evaluate the running time of the SMA algorithm under different types of query nodes: sparse, average, dense, and random. Dense queries correspond to nodes whose coreness values are within the top 10% of the graph, whereas sparse queries refer to those in the bottom 10%. Average queries are chosen from nodes whose coreness values fall between the top 45% and 55%, while random queries are uniformly sampled from all nodes. A single-query setting is adopted, where queries are randomly selected for each type, and the average running time is

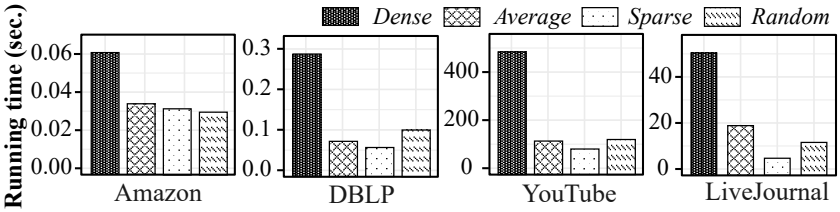
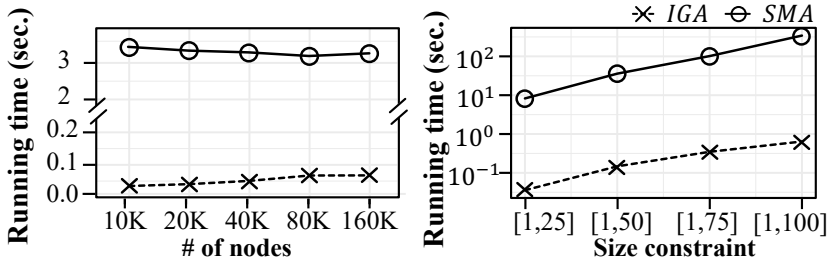


Fig. 14. EQ6. Different query types

reported. Figure 14 illustrates the running time results for each query type. It can be observed that dense queries require more time than sparse ones. This phenomenon results from the chain identification and greedy expansion processes. Specifically, a query node with high coreness has a larger number of neighbours, thereby increasing both the number of nodes considered during the expansion phase and the number of chains constructed. Average and random queries have similar behaviour, with running times lying between those of the sparse and dense cases. Across all datasets, dense queries are 1.9–10.8× slower than sparse ones. Although YouTube and LiveJournal incur higher costs for dense queries, such cases are uncommon in practice, and most queries (average or random) are processed efficiently.

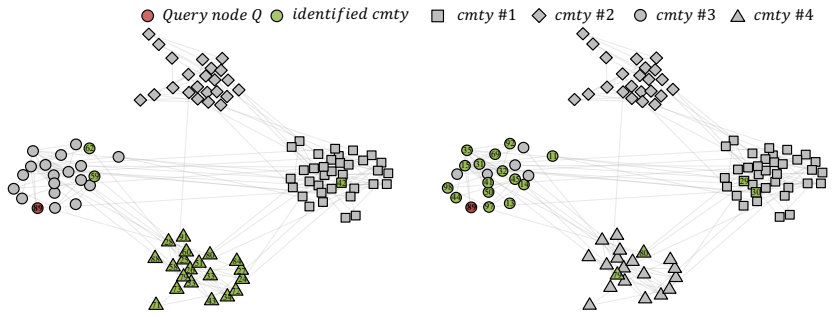


(a) Varying node size (b) Varying size constraint

Fig. 15. EQ7. Scalability

EQ7. Scalability. To evaluate scalability, we use the LFR benchmark [22] under two settings: (1) varying the number of nodes while fixing the size constraints l and h to default values, and (2) fixing the network size while increasing the size constraints from $[1, 25]$ to $[1, 100]$. Figure 15a shows that both IGA and SMA maintain stable performance as network size increases. In contrast, Figure 15b shows that runtime grows linearly with increasing size constraints. These results suggest that the size constraint has a greater influence on runtime than the network size itself.

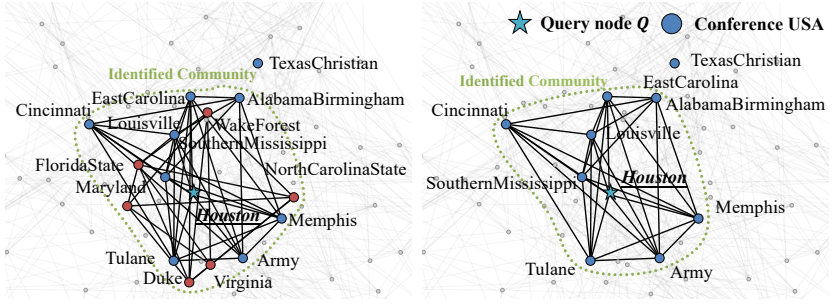
EQ8. Limitation of Greedy Expansion in IGA. We investigate the intrinsic limitations of greedy node-wise expansion in IGA through an experiment on an LFR benchmark network, where node 89 is chosen as the query with a size constraint of $[10, 31]$ (Figure 16). Although IGA and SMA both select the same initial core $\{89, 80, 79\}$, IGA approach—adding one node at a time based on marginal LSM gain—often leads to suboptimal results, as it is prone to local optima and can incorporate weakly connected or even topologically irrelevant nodes such as 26 and 24. This strategy causes IGA to overlook the broader structure of the ground-truth community, ultimately producing a fragmented result with a low F1-score of 0.4408. In contrast, SMA overcomes this limitation by identifying and merging cohesive chains of nodes, such as $\{15, 35, 41, 69, 92, 32, 31, 98, 13, 14, 97, 45, 44, 50\}$, all belonging to the true community. By considering group-level connectivity, SMA recovers the correct community structure and achieves a substantially higher F1-score of 0.8619. This result



(a) IGA (b) SMA

Fig. 16. EQ8. Comparison of IGA and SMA

presents the limitation of naive greedy expansion and demonstrates the importance of group-wise structural optimisation to avoid being trapped in local optima.



(a) STCS (b) SMA

Fig. 17. EQ9. Case study

EQ9. Case study on football network. We conduct a case study using the American college football network [14], which represents regular-season games played between Division I-A colleges during Fall 2000. Each node corresponds to a team, and each edge indicates a game played between two teams. The network comprises 115 nodes and 613 edges, with ground-truth communities defined by 12 major conferences (e.g., Atlantic Coast, Big Ten, Pacific Ten). To evaluate the effectiveness of our method in identifying size-constrained and semantically coherent communities, we select the team 'Houston' as the query node from the *Conference USA*, applying a size constraint of $[5, 15]$ based on the actual size of the corresponding ground-truth community, and set the sketch threshold to 1. The quality of the resulting community is evaluated using the F1-score. We compare our SMA algorithm against STCS, which demonstrated superior accuracy over SCS in EQ2. As illustrated in Figure 17, the community produced by STCS includes several unrelated nodes (coloured in red), yielding an F1-score of 0.84. In contrast, our method achieves an F1-score of 0.97, reflecting improved performance in extracting compact and topically coherent communities.

8 RELATED WORK

Existing community search methods can be grouped in various ways. In this study, we categorise them according to whether they enforce both, one, or no size constraints, as this directly affects their applicability to the *two-sided* size-constrained setting of our problem.

Two-sided size-constrained models. These approaches explicitly restrict both lower and upper bounds $[l, h]$ of community size. Yao et al. [43] formalised the size-bounded community search (SCS) problem, which aims to find a connected subgraph containing a query node q that maximises the minimum degree under a size constraint. They proposed SC-BRB, an exact algorithm for narrow bounds ($h - l \leq 3$, $h < 20$), leveraging reduction rules within a branch-and-bound framework. Zhang et al. [45] extended this line by introducing STCS, which replaces degree with minimum edge support and integrates budget-cost constraints. While effective, these methods require full access to the graph, limiting their scalability in large or dynamic networks. Both explicitly optimise under bounded size intervals, aligning closely with our formulation.

One-sided size-constrained models. These methods impose only an upper bound on community size. Sozio and Gionis [35] introduced the GreedyDist algorithm, which searches for a subgraph maximising the minimum degree while satisfying a distance constraint d and an upper size limit h. If no feasible solution is found, the algorithm reduces d via binary search and re-executes the search. Liu et al. [23] proposed SCkT, which finds a triangle-connected k-truss containing the query node under an upper bound h, using a goodness-based pruning–expansion strategy. However, as reported in [45], the result is highly sensitive to the choice of k and often produces trivial or empty communities when k is mis-specified.

Methods without explicit size constraints. Beyond size-constrained models, a large body of community search focus on structural cohesiveness without bounding community size. Luo and Wang [25] analysed local modularity to evaluate how sharply a subgraph is separated from the rest of the network, and Kim et al. [21] extended this notion that jointly captures internal density and external separability. Sozio et al. [35] formalised the community search problem as finding a connected subgraph maximising minimum degree with a query constraint. Cui et al. [7] further developed this framework by proposing both global and local expansion strategies. Wu et al. [40] proposed the query-biased density model, which reweights nodes according to their proximity to the query node. These frameworks have also been generalised to directed networks [10, 24], bipartite networks [39], attributed networks [8, 15]. Recent surveys [9] highlight that unconstrained community search remains a central and extensible paradigm across diverse graph settings.

9 CONCLUSION

We introduced the Local Sketch Modularity for Size-Constrained Community Search (LMSC) problem, which aims to find query-centric community under size constraints using local graph information. To address this, we proposed LSM, a measure that penalises large communities to reduce free-rider effects. Our framework enables scalable search without full graph access. We developed two algorithms: the Incremental Greedy Algorithm (IGA), which greedily expands the community, and the Sub-chain Merge Algorithm (SMA), which expands the community by merging cohesive structures. Theoretical and empirical results on real-world and synthetic networks demonstrate the effectiveness and efficiency of our approaches. Future work includes extending our framework to dynamic or attributed graphs and further improving computational scalability.

Acknowledgments

This research was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT)(No.2020-0-01336, Artificial Intelligence graduate school support(UNIST)), the National Research Foundation of Korea(NRF) grant funded by MSIT(No.RS-2025-00523578), grant No. 62394333 from the National Natural Science Foundation of China (NSFC), the IITP-ICT Creative Consilience Program grant (IITP-2026-RS-2020-II201819), and the artificial intelligence star fellowship support program to nurture the best talents (IITP-2026-RS-2025-02304828) funded by the Korea government(MSIT).

References

[1] Alessandro Acquisti, Laura Brandimarte, and George Loewenstein. 2015. Privacy and human behavior in the age of information. *Science* 347, 6221 (2015), 509–514.

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

[3] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. 2006. Maximizing modularity is hard. *arXiv preprint physics/0608255* (2006).

[4] Tanmoy Chakraborty, Sriram Srinivasan, Niloy Ganguly, Animesh Mukherjee, and Sanjukta Bhowmick. 2014. On the permanence of vertices in network communities. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, New York, NY, USA, 1396–1405.

[5] Deming Chu, Fan Zhang, Xuemin Lin, Wenjie Zhang, Ying Zhang, Yinglong Xia, and Chenyi Zhang. 2020. Finding the best k in core decomposition: A time and space optimal solution. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 685–696.

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

[7] Wanyun Cui, Yanghua Xiao, Haixun Wang, and Wei Wang. 2014. Local search of communities in large graphs. In *SIGMOD*. ACM, New York, NY, USA, 991–1002.

[8] Yixiang Fang, Reynold Cheng, Siqiang Luo, and Jiafeng Hu. 2016. Effective community search for large attributed graphs. *Proceedings of the VLDB Endowment* 9, 12 (2016), 1233–1244.

[9] 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.

[10] 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.

[11] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[12] Santo Fortunato and Marc Barthelemy. 2007. Resolution limit in community detection. *Proceedings of the national academy of sciences* 104, 1 (2007), 36–41.

[13] Mohadeseh Ganji, James Bailey, and Peter J Stuckey. 2017. A declarative approach to constrained community detection. In *Principles and Practice of Constraint Programming: 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28–September 1, 2017, Proceedings 23*. Springer, 477–494.

[14] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *PNAS* 99, 12 (2002), 7821–7826.

[15] Xin Huang and Laks VS Lakshmanan. 2017. Attribute-driven community search. *Proceedings of the VLDB Endowment* 10, 9 (2017), 949–960.

[16] Xin Huang, Laks VS Lakshmanan, and Jianliang Xu. 2019. *Community search over big graphs*. Morgan & Claypool Publishers.

[17] Xin Huang, Laks VS Lakshmanan, Jeffrey Xu Yu, and Hong Cheng. 2015. Approximate Closest Community Search in Networks. *Proceedings of the VLDB Endowment* 9, 4 (2015), 276–287.

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

[19] Dahee Kim, Taejoon Han, Kaiyu Feng, Junghoon Kim, and Susik Yoon. Online appendix of LMSC: Local Sketch Modularity Optimisation for Size-Constrained Community Search in Networks. Available at: https://github.com/dahee-e/LMSC/blob/main/Appendix.pdf.

[20] Junghoon Kim, Tao Guo, Kaiyu Feng, Gao Cong, Arijit Khan, and Farhana M Choudhury. 2020. Densely connected user community and location cluster search in location-based social networks. In *SIGMOD*. ACM, New York, NY, USA, 2199–2209.

[21] Junghoon Kim, Siqiang Luo, Gao Cong, and Wenyuan Yu. 2022. DMCS : Density Modularity Based Community Search. In *SIGMOD*. ACM, New York, NY, USA, 889–903.

[22] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *PRE* 78, 4 (2008), 046110.

[23] Boge Liu, Fan Zhang, Wenjie Zhang, Xuemin Lin, and Ying Zhang. 2021. Efficient community search with size constraint. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 97–108.

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

[25] Feng Luo, James Z. Wang, and Eric Promislow. 2006. Exploring Local Community Structures in Large Networks. In *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*. IEEE, USA, 233–239.

[26] David Lusseau, Karsten Schneider, Oliver J Boisseau, Patti Haase, Elisabeth Slooten, and Steve M Dawson. 2003. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations: can geographic isolation explain this unique trait? *Behavioral ecology and sociobiology* 54 (2003), 396–405.
[27] Kurt Mehlhorn. 1988. A faster approximation algorithm for the Steiner problem in graphs. *Inform. Process. Lett.* 27, 3 (1988), 125–128.
[28] Henning Meyerhenke, Peter Sanders, and Christian Schulz. 2014. Partitioning complex networks via size-constrained clustering. In *International Symposium on Experimental Algorithms*. Springer-Verlag, Berlin, Heidelberg, 351–363.
[29] Mark EJ Newman. 2006. Modularity and community structure in networks. *Proceedings of the national academy of sciences* 103, 23 (2006), 8577–8582.
[30] Mark EJ Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical review E* 69, 2 (2004), 026113.
[31] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. 2004. Defining and identifying communities in networks. *Proceedings of the national academy of sciences* 101, 9 (2004), 2658–2663.
[32] Ryan Rossi and Nesreen Ahmed. 2015. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 29. AAAI Press.
[33] Stephen B Seidman. 1983. Network structure and minimum degree. *Social networks* 5, 3 (1983), 269–287.
[34] Devang Shah, Hriday Ranka, Lynnette Hui Xian NG, and Swapneel Mehta. 2024. Bridging Nodes and Narrative Flows: Identifying Intervention Targets for Disinformation on Telegram. *arXiv preprint arXiv:2411.05922* (2024).
[35] 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*. ACM, New York, NY, USA, 939–948.
[36] Victor Spirin and Leonid A Mirny. 2003. Protein complexes and functional modules in molecular networks. *Proceedings of the national Academy of sciences* 100, 21 (2003), 12123–12128.
[37] Cong Tran, Won-Yong Shin, and Andreas Spitz. 2021. Community detection in partially observable social networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 16, 2 (2021), 1–24.
[38] 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.
[39] 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 *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 85–96.
[40] Yubao Wu, Ruoming Jin, Jing Li, and Xiang Zhang. 2015. Robust local community detection: on free rider effect and its elimination. *Proceedings of the VLDB Endowment* 8, 7 (2015), 798–809.
[41] Jaewon Yang and Jure Leskovec. 2012. Defining and Evaluating Network Communities Based on Ground-Truth. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*. IEEE Computer Society, USA, 745–754.
[42] Zhao Yang, René Algesheimer, and Claudio J Tessone. 2016. A comparative analysis of community detection algorithms on artificial networks. *Scientific reports* 6, 1 (2016), 30750.
[43] Kai Yao and Lijun Chang. 2021. Efficient size-bounded community search over large networks. *Proceedings of the VLDB Endowment* 14, 8 (2021), 1441–1453.
[44] Wayne W Zachary. 1977. An information flow model for conflict and fission in small groups. *Journal of anthropological research* 33, 4 (1977), 452–473.
[45] Fan Zhang, Haicheng Guo, Dian Ouyang, Shiyu Yang, Xuemin Lin, and Zhihong Tian. 2023. Size-constrained community search on large networks: An effective and efficient solution. *IEEE Transactions on Knowledge and Data Engineering* 36, 1 (2023), 356–371.

Received July 2025; revised October 2025; accepted November 2025