Streaming Local Community Detection Through Approximate Conductance

Meng Wang, Yanhao Yang, David Bindel, and Kun He, Senior Member, IEEE

Abstract—Community is a universal structure in various complex networks, and community detection is a fundamental task for network analysis. With the rapid growth of network scale, networks are massive, changing rapidly, and could naturally be modeled as graph streams. Due to the limited memory and access constraint in graph streams, existing non-streaming community detection methods are no longer applicable. This raises an emerging need for online approaches. In this work, we consider the problem of uncovering the local community containing a few query nodes in graph streams, termed streaming local community detection. This new problem raised recently is more challenging for community detection, and only a few works address this online setting. Correspondingly, we design an online single-pass streaming local community detection approach. Inspired by the local property of communities, our method samples the local structure around the query nodes in graph streams and extracts the target community on the sampled subgraph using our proposed metric called approximate conductance. Comprehensive experiments show that our method remarkably outperforms the streaming baseline on both effectiveness and efficiency, and even achieves similar accuracy compared to the state-of-the-art non-streaming local community detection methods that use static and complete graphs.

Index Terms—Network analysis, graph stream, local community detection, approximate conductance.

I. INTRODUCTION

NETWORKS exhibit natural structures representing entities and their relationships for complex systems in various domains, e.g., society, biology, communication, and the World Wide Web, in which vast amounts of data are constantly being generated. As a fundamental task of network analysis, community detection aims to uncover groups of densely connected nodes, termed communities, and has attracted extensive attention. Much effort has been devoted to global community detection for decades [1], [2], [3], [4], [5]. In recent years, there has been a growing interest in exploring the local community containing a few query nodes [6], [7], [8], [9]. From the perspective of computational cost, the problem of local community detection is more suitable for uncovering the community structure for nodes of interest on large-scale networks, considering the real-world scenarios like recommendation systems and political activism study [10].

Existing community detection methods, either globally or locally, focus on mining static and complete networks, which take a considerable overhead on the memory. With the rapid growth of the network scale, the memory space required for handling such massive networks becomes unacceptable for practical applications. For example, in 2020, Twitter had 340 million users, and each user had an average of 707 followers; Facebook had 1.79 billion daily active users, and each user had an average of 338 friends. When these websites are constructed as networks, we will obtain two massive networks with billions of nodes and billions to trillions of edges, which occupy petabytes of storage space if we represent them via adjacency matrices. Loading such networks entirely into the main memory is unrealistic, even in distributed systems [11]. Moreover, under the real-world scenarios, networks are usually changing rapidly, and relationships are only established instantaneously, e.g., in an email exchange network or product co-purchase network. Such networks generated by transient online interactions are naturally streaming data [12] and can be modeled as graph streams.

We consider the problem of local community detection in graph streams, termed streaming local community detection. Graph stream is a streaming model in which the graph data, e.g., edges of the graph, are arranged as a stream based on their generation order. The stream is sometimes infinite, and the newly generated data will be added in the end. Given a massive and natural streaming network $G = (V, E)$, the graph stream can organize the data with limited memory. However, there is another side to this coin, data in the stream can only be accessed once in the order it arrives, and methods for graph stream mining should only occupy limited memory. Furthermore, the global information about the network is unknown before the stream is completely processed.

Recently, there have been several works in graph streams. Researchers mainly focus on mining relatively simple structures in graph streams, such as counting triangles [13], [14], [15] and wedges [14], sampling motifs [12] and densest subgraphs [16]. Compared to these structures, the community is much more complex, and hence it is more challenging to uncover community in graph streams. And for existing local community detection
methods working on static and complete networks [7], [17], [18], [19], the limited memory and access constraint in graph streams makes these methods no longer applicable.

In this work, we propose an online single-pass streaming local community detection method called SCDAC (Streaming Community Detection via Approximate Conductance), for uncovering the target local community containing the query nodes in graph streams. Inspired by the “local” property of communities [20], we first sample a subgraph covering the local community structure around the query nodes in the arriving graph stream. Then following the spirit of seed set expansion [7], [19], [21], [22], we regard the query nodes as a seed-set and expand it to a comparatively large node set that exhibits high community quality on the sampled subgraph.

However, as the entire graph cannot be cached in graph streams, a number of popular metrics evaluating community quality are unavailable, like conductance [23] and modularity [24]. Moreover, the subgraph sampled in the graph streams is usually incomplete for the exact local structure around the query nodes, and the local metrics adopted by existing local community detection methods on complete subgraph [19], [22] are inaccurate on such subgraphs. To address this issue, we propose a metric called approximate conductance to measure the community quality, which aims to approximate the exact conductance of the community on the incomplete subgraph sampled in graph streams. The computational cost of our method, including the time and space complexity, is thoroughly analyzed. Extensive experimental results verify the effectiveness and efficiency of the method.

Our main contributions are summarized as follows:

- We design an online single-pass streaming local community detection method, which uncovers the target local community for query nodes in graph streams without prior knowledge.
- We propose a new metric called approximate conductance to measure the community quality on the subgraph sampled in graph streams.
- Extensive experiments on real-world networks from various domains demonstrate the superiority of our method in terms of effectiveness and efficiency.

II. BACKGROUND

A. Problem Formulation

Consider a network modeled as an undirected and unweighted graph \( G = (V, E) \), where \( V = \{v_i | i = 1, 2, \ldots\} \) represents the node set and \( E = \{e_i | i = 1, 2, \ldots\} \) represents the edge set. A given subset of nodes \( T \) (called query nodes or query node set) is contained in a target community \( C^* \). Generally, \( |T| \ll |C^*| \ll |V| \). The task of local community detection is to uncover the target community \( C^* \) containing the query node set \( T \) on the static complete graph \( G \).

In view of the massive and natural streaming networks, we introduce streaming local community detection as follows.

**Definition 1 (Streaming Local Community Detection):** Consider a stream \( S \) as an arbitrarily ordered sequence of all the edges in network \( G = (V, E) \), denoted as \( \langle e_1, e_2, \ldots, e_i, \ldots \rangle \).

Given a query node set \( T \), the task of streaming local community detection is to uncover the target community \( C^* \) containing the query node set \( T \) using the stream \( S \) under the limit of \( O(|V|) \) work memory.

This definition is a general description of streaming local community detection. In this work, we consider a more demanding streaming model, where the edges in the graph stream are only available once, rather than a sliding-window model [25]. In more detail, when processing an edge \( e_i \) in stream \( S \), the past edges cannot be reaccessed, and the subsequent edges in the stream are unknown, i.e., \( \forall j \neq i, e_j \) is not accessible when \( e_i \) arrives.

B. Related Work

1) Local Community Detection: In contrast to the global community detection [1], [2], [3], [4], [26], which aims to find all communities in a network, the local community detection [6], [7], [9], [18], [21] is a query-oriented problem to find the target community containing the query node(s).

The phase of community extraction in our method falls into the category of seed-set expansion, a popular category of methods for local community detection [7], [19], [21], [22]. The general idea of seed-set expansion is to expand the seed-set along the network topology and obtain a community by optimizing a scoring function. Common diffusion methods include random walk [21], heat kernel [7] and local spectral approximation [19], [22]. Metrics measuring the community quality are usually adopted as the scoring function, such as modularity [24] and conductance [23].

Especially, PRN [21] is a pioneer of seed-set expansion, which adopts lazy random walk as the diffusion method and uncovers the community with local minimum conductance. Kloster and Gleich [7] adopt heat kernel as the expansion method and present a deterministic local algorithm to compute the heat kernel diffusion by coordinate relaxation. LEMON [22] and LOSP [19] perform short random walks and span an approximate invariant subspace termed the local spectral subspace, and obtain the membership vector in the subspace.

All the methods discussed above run on static and complete graphs, which must be loaded into the main memory entirely. This limits the application of local community detection on massive and streaming networks.

2) Streaming Community Detection: As a feasible approach to handling massive networks, graph stream has become increasingly popular in recent years [25]. A considerable amount of literature on graph stream model has been proposed, covering many areas of network and graph analysis, including estimating connectivity [27], [28], sampling subgraph [12], [13], [14], [15], outlier detection [29], link prediction [30], and maintaining the projection of graphs [31], [32], [33].

The problem of community detection in graph stream (also called streaming community detection) has also attracted attention. Yun et al. [34] propose a method to detect communities in a stream composed of the columns of the adjacency matrix. Hollocou et al. [35] consider a stream consisting of edges and propose a streaming community detection method based on the
assumption that an edge randomly selected in the stream is more likely to be inside the community. Wu et al. [36] develop a streaming belief-propagation approach based on a streaming stochastic block model to handle the community detection problem for networks growing over time. However, all these methods focus on detecting all communities of a network and belong to the streaming global community detection category. While very little work has studied streaming local community detection. To our knowledge, only one work aims to detect a local community containing the query nodes in graph streams, and proposes a method called CoEuS [37]. CoEuS expands communities in a greedy way, and addresses a periodical pruning operation simply by sorting each node’s ratio of its connection inside a community to its total degree, which cannot be counted accurately in a graph stream. In this paper, we sample a subgraph from the streaming model, recording the node’s degrees that are consistent with the original graph, then use this information to seek an optimal community by the newly designed metric approximate conductance. Extensive experiments show that our method significantly outperforms CoEuS in both effectiveness and efficiency.

III. ALGORITHM

This section introduces the proposed algorithm in detail. As nodes in the same community are densely connected, the distance between each pair of nodes inside a community is usually short. In the local community detection task, the target community $C^*$ always lurks in the neighborhood of the query nodes. Therefore, we first sample the local structure around the query node set $T$ in graph stream $\mathcal{S}$, then extract a high-quality community containing these query nodes from the local structure. Fig. 1 provides an overview of our method.

A. Stream Sampling

We first provide several definitions on undirected and unweighted graphs. The distance from node $u$ to node $v$, termed $\text{dist}(u, v)$, is the length of the shortest path between the two nodes. The distance from node $u$ to a node set $T$, termed $\text{dist}(u, T)$, is the minimum distance from $u$ to nodes in $T$, i.e., $\text{dist}(u, T) = \min_{v \in T} \text{dist}(u, v)$. The $k$-hop neighborhood of node $v$ is a set of nodes whose distance to $v$ is no greater than $k$, i.e., $\{u | \text{dist}(u, v) \leq k\}$. Similarly, the $k$-hop neighborhood of a node set $T$ is a set of nodes whose distance to $T$ is no greater than $k$, i.e., $\{u | \text{dist}(u, T) \leq k\}$.

The local structure around the query node set $T$ is a subgraph induced by nodes in $T$ and their $k$-hop neighbors ($k$ is a small positive integer) [20]. However, as we can only access each edge in the stream once and have no prior information on the coming edges before they arrive, it is tough to sample this local structure precisely, even for $k = 1$. And the greater the value of $k$ is, the less accurate the sampled subgraph is. Thus, the goal of this phase is to sample subgraph $G_s = (V_s, E_s)$, which can approximately cover the $k$-hop neighborhood of the query nodes $T$.

Also, the subgraph size, denoted as $|V_s|$, is related to the power of node degrees. The sampled subgraph will be expanded too large in some networks with relatively high average node degrees. For instance, in Youtube network, a subgraph covering the 4-hop neighborhood of a few query nodes could reach millions of nodes, but we are only interested in the nodes that are densely connected with the query nodes. To address this issue, we only keep a certain number of nodes closest to the query nodes in the sampled subgraph. Furthermore, we consider the situation that some nodes are far away from $T$ when they first arrive in the stream but actually close to the query nodes on the complete graph. Then we give these nodes a “probation period” not to be expelled immediately, i.e., we prune the subgraph to $ps$ nodes only when every $pc$ streaming edges are processed, where $pc$ and $ps$ are termed the pruning cycle and pruning size, respectively. Details of the stream sampling are presented in Algorithm 1. In the end, we output the sampled subgraph $G_s$ and an array $D$, recording the accumulated degree of all nodes in the stream.

In Algorithm 1, we need to calculate $\text{dist}(v, T)$ many times for a node $v$ and a node set $T$. This is a typical single source shortest path problem in the graph, and a well-known solution is based on Breath-First Search with the time complexity of
Algorithm 1: Stream Sampling.

Input: Graph stream $S$, query nodes $T$.
Output: Sampled subgraph $G_s = (V_s, E_s)$, degree array $D$.

Parameters: Number of hops $k$, pruning cycle $pc$, pruning size $ps$.

1: $D \leftarrow 0$
2: $G_s \leftarrow (T, \emptyset)$
3: for each $e_i = (u, v) \in S$ do
4:      $D[u] \leftarrow D[u] + 1$
5:      $D[v] \leftarrow D[v] + 1$
6:          if $dist(u, T) \leq k$ end $dist(v, T) \leq k$ in
7:              $G'_s = (V_s, E_s \cup \{(u, v)\})$ then
8:      $V_s \leftarrow V_s \cup \{u, v\}$
9:          $E_s \leftarrow E_s \cup \{(u, v)\}$
10: end if
11: if $i \mod pc = 0$ then
12:      sort $V_s$ in ascending order by $dist(v, T)$
13:      $V_s \leftarrow \{v_j | v_j \in V_s \land j \leq ps\}$
14:      $E_s \leftarrow \{(u, v) | u \in V_s \land v \in V_s\}$
15: end for

B. Community Extraction

After we sample a subgraph covering the local structure around the query nodes from the graph stream, the second phase is to find the target community $C^*$ containing the query nodes on the sampled subgraph $G_s$. Considering that nodes within the same community are closely connected, we adopt the seed-set expansion strategy from the query nodes in $T$. For each node in $G_s$, we evaluate its probability of belonging to the target community by the closeness of connection to the seed-set, and add nodes to the seed-set in descending order based on the probability by optimizing the community quality.

In the following, we will introduce in detail the probability diffusion method and the proposed approximate conductance metric for measuring the community quality, as well as the algorithm flow.

1) Diffusion Method: To measure the closeness of the connection between seed-set $T$ and the remaining nodes in subgraph $G_s$, we employ a lazy random walk for probability diffusion. For each node $u \in T$, we start a lazy random walk from $u$, with 0.5 probability staying at the current node and 0.5 probability following an adjacent edge chosen uniformly and randomly. Repeating this operation several times, we will obtain a probability distribution covering all nodes in subgraph $G_s$, and regard the probability on each node as its closeness to seed-set $T$.

The transition matrix $N_{rw}$ of the lazy random walk is given by:

$$N_{rw} = (I + D_s^{-1}A_s)/2,$$

where $A_s$ is the adjacent matrix of $G_s$, $D_s$ is the diagonal matrix of degrees on $G_s$, and $I$ is an identity matrix of order $|V_s|$. 

Fig. 2. Example of organizing the node set of the sampled subgraph in the distance tree, where $\{1, 2, 3\}$ are the query nodes, and edge $(2, 6)$ is the coming edge in the graph stream.

$O(|V_s| + |E_s|)$. Inspired by the idea of path compression in the union-find set, we consider maintaining node set $V_s$ in a distance tree to speed up the calculation on $dist(\cdot, T)$. Consider the example in Fig. 2, node 0 is a dummy root node, the query nodes in $T$ are with depth 1, and other nodes are the children of their adjacent node which are in their shortest paths to the query nodes, i.e., for node $v$,

$$parent(v) = \arg \min_{u \in N(v)} dist(u, T),$$

where $N(v)$ denotes the set of nodes adjacent to $v$ in the sampled subgraph $G_s$. In this way, the problem of calculating the distance from node $v$ to the query node set $T$ can be converted into the problem of calculating the depth of node $u$ in the tree. Formally, we have

$$dist(v, T) = \begin{cases} depth(u) - 1 & \text{if } v \in V_s, \\ \infty & \text{otherwise}. \end{cases}$$

The tree of node set $V_s$ can be implemented as an array, then the time of calculating $dist(v, T)$ is reduced from $O(|V_s| + |E_s|)$ to $O(k)$.

When a new edge $(u, v)$ is added into the subgraph $G_s$, the distance between the query node set $T$ and other nodes in the subgraph may change, then the distance tree needs to be updated accordingly. There are two cases in which the distance from $T$ to other nodes will be updated. Without loss of generality, we assume $dist(u, T) < dist(v, T)$. One case is that $v$ is not in $V_s$ before, and when $(u, v)$ is added, $dist(v, T)$ should become $dist(u, T) + 1$. Correspondingly, the distance tree needs to add a new node $v$ as a child of node $u$. The other is that both $u$ and $v$ are already in $V_s$ and $dist(v, T) - dist(u, T) \geq 2$ before. When $(u, v)$ is added, $dist(v, T)$ should become $dist(u, T) + 1$ and $dist(\cdot, T)$ of the nodes whose shortest paths to $T$ containing $(u, v)$ should also be updated. For this complex situation, the distance tree only needs to change the parent node of node $v$ to $u$, and the time complexity of this operation is $O(1)$. As the example shown in Fig. 2, when a new edge $(2, 6)$ is added into the subgraph, the shortest path from node 6 to the query nodes becomes $6 \rightarrow 2$. Thus the parent of node 6 changes from node 7 to node 2, and we only need to modify one element in the array storing the parent node.
The initial probability $p \in [0, 1]^{V_s}$ is concentrated evenly at the seed-set, defined formally as:
\[
p_i = \begin{cases} 
\frac{1}{|T|} & \text{if } v_i \in T, \\
0 & \text{if } v_i \in V_s \setminus T.
\end{cases}
\]
Since the sampled subgraph $G_s$ only covers the $k$-hop neighborhood of seed-set $T$, a $k$-step lazy random walk is capable of diffusing the probability to the entire subgraph $G_s$.

2) Scoring Function: When the size of the target community (ground truth size) is known, the community can be obtained by cutting the probability distribution vector $p$ with the size as a budget. Unfortunately, the target size is unavailable in many real-world scenarios, especially in naturally streaming networks. To address this issue, we determine the community boundary automatically by optimizing a scoring function.

Modularity [24], the metric for evaluating graph partitions, is usually adopted by global community detection methods. Conductance [23] is also a community quality metric, highly correlated with the degree of tight internal connections and sparse external connections. Different from modularity, conductance is usually used to evaluate a single community and employed in various local community detection methods as the scoring function [19], [21], [22].

Definition 2 (Conductance): Given a graph $G = (V, E)$, for a node set $C$ and $C \subset V$, the conductance of the induced subgraph of $C$ is:
\[
\Phi(C) = \frac{\text{cut}(C, V \setminus C)}{\min\{\text{Vol}(C), \text{Vol}(V \setminus C)\}},
\]
where $\text{cut}(\cdot, \cdot)$ denotes the number of edges between two node sets and $\text{Vol}(\cdot)$ denotes the total degree of a node set.

For a large-scale graph, the cost of calculating conductance by (5) is unacceptable. If there is a subgraph covering the local structure around community $C$ (i.e., community $C$ and its adjacent nodes), calculating $\text{Vol}(C)$ on the complete graph can be simplified to calculate locally on the subgraph [19], [22]. When $|C| \ll |V|$, the denominator of $\Phi(C)$ can be simplified to $\text{Vol}(C)$, and this condition is usually satisfied for community detection tasks. According to existing practices, the conductance on the entire graph can be simplified to the conductance on the subgraph:
\[
\Phi_s(C) = \frac{\text{cut}(C, V_s \setminus C)}{\text{Vol}_s(C)},
\]
where $\text{Vol}_s(\cdot)$ denotes the total degree of a node set on the subgraph.

As the entire graph cannot be cached in a graph stream, the exact conductance can only be calculated based on limited information about the sampled subgraph in the stream, which is usually incomplete compared to a subgraph sampled on a static complete graph. Since edges come in arbitrary order and can only be accessed once, we will miss some useful nodes and edges that are not connected to nodes sampled into the subgraph before. As illustrated in the example in Fig. 3, neither $\text{cut}(C, V_s \setminus C)$ nor $\text{Vol}_s(C)$ on this subgraph can be guaranteed to be accurate.

Consequently, the conductance calculated by (6) differs from its original value on the subgraph sampled in the graph stream.

Though we cannot cache the complete graph from the graph stream, we observe that some information can still be recorded accurately during the sampling. Specifically, we count the times that each node appears when processing the graph stream so that we can obtain the degree of each node in $V_s$, which is stored in the degree array $D$ in Algorithm 1. And $\text{cut}(C, V \setminus C)$, the number of edges between $C$ and $V \setminus C$, is equivalent to the value that the total degrees of nodes in $C$ minus twice the number of edges within $C$. Hence, we propose an approximate conductance on the sampled subgraph as follows:
\[
\Phi_s(C) = \frac{\text{Vol}(C) - 2|E_s(C, C)|}{\text{Vol}(C)},
\]
where $E_s(C, C)$ denotes the set of edges within the node set $C$ on the subgraph $G_s$.

Compared to the existing conductance calculated locally on the sampled subgraph, the approximate conductance is more suitable for graph streams. There is still a deviation between the approximate conductance and the exact conductance, which is due to the difference between $E_s(C, C)$ and $E(C, C)$. However, the closer an edge to the query nodes, the less likely it will be missed during the stream sampling. The nodes in the community $C$ are usually closely connected to the query nodes, indicating that the approximate conductance can be considered a qualified deputy for the exact conductance.

In Fig. 3, we regard the nodes within the light blue area as the members of $C$. For the exact conductance $\Phi(C)$, $\text{cut}(C, V \setminus C) = 8$, $\min\{\text{Vol}(C), \text{Vol}(V \setminus C)\} = \text{Vol}(C) = 40$, and the metric value is 0.20. For the existing conductance calculation locally on the sampled subgraph $\Phi_s(C)$, $\text{cut}(C, V_s \setminus C) = 3$, $\text{Vol}_s(C) = 33$, and the metric value is 0.09. For the approximate conductance $\Phi_s(C)$, $|E_s(C, C)| = 15$, $\text{Vol}(C) = 40$, which is derived from $D$, and the metric value is 0.25, apparently closer than the previous calculation to the exact conductance value on the complete graph.

Actually, when the ratio of the number of the missed edges inside the target community to that of all missed edges is small enough, the proposed approximate conductance must be more accurate than the locally-calculated conductance. Assume that at
least one edge within or connected with $C$ is abandoned during the sampling process, and let $\Delta \text{Vol}(C) = \text{Vol}(C) - \text{Vol}_s(C) \neq 0$ and $\Delta |E(C, C)| = |E(C, C)| - |E_s(C, C)|$, then we have the following theorem.

**Theorem 1:** For the target community $C$, when $|C| \ll |V|$, the value of approximate conductance $\Phi_s(C)$ is closer to that of the exact conductance $\Phi(C)$ than the locally-calculated conductance $\Phi_s(C)$ if and only if

$$\frac{2\Delta |E(C, C)|}{\Delta \text{Vol}(C)} < \frac{|E_s(C, C)|}{\text{Vol}_s(C)}.$$  

(8)

**Proof:** When $|C| \ll |V|$, the calculations of the three metrics are respectively

$$\Phi(C) = \frac{\text{Vol}(C) - 2|E(C, C)|}{\text{Vol}(C)},$$

$$\Phi_s(C) = \frac{\text{Vol}_s(C) - 2|E_s(C, C)|}{\text{Vol}_s(C)},$$

$$\Phi_s(C) = \frac{\text{Vol}(C) - 2|E_s(C, C)|}{\text{Vol}(C)}.$$  

Because $\text{Vol}(C) - \text{Vol}_s(C) > 0$, $\Phi_s(C) > \Phi_s(C)$. Then we have

$$(\Phi_s(C) - \Phi(C)) - (\Phi(C) - \Phi_s(C))$$

$$= \left( \frac{\text{Vol}(C) - 2|E(C, C)|}{\text{Vol}(C)} \right) - \left( \frac{\text{Vol}(C) - 2|E_s(C, C)|}{\text{Vol}(C)} \right)$$

$$- \left( \frac{\text{Vol}_s(C) - 2|E_s(C, C)|}{\text{Vol}_s(C)} \right)$$

$$= 4|E(C, C)|\text{Vol}_s(C) - 2|E_s(C, C)| \left( \frac{\text{Vol}(C) + \text{Vol}_s(C)}{\text{Vol}(C)} \right).$$

To make $\Phi_s(C)$ closer to $\Phi(C)$ than $\Phi_s(C)$, the value of the above expression needs to be less than 0, which is equivalent to

$$2|E(C, C)|\text{Vol}_s(C) < |E_s(C, C)| \left( \frac{\text{Vol}(C) + \text{Vol}_s(C)}{\text{Vol}(C)} \right)$$

$$\iff 2|E(C, C)| < \frac{|E_s(C, C)| (\Delta \text{Vol}(C) + 2\text{Vol}_s(C))}{\text{Vol}_s(C)}$$

$$\iff \frac{2\Delta |E(C, C)|}{\Delta \text{Vol}(C)} < \frac{|E_s(C, C)|}{\text{Vol}_s(C)}.$$  

$\square$

3) **Community Size Determination:** The probability distribution vector $p$, obtained by a $k$-step lazy random walk, indicates the closeness between nodes in $V_s$ and the seed-set $T$, e.g., $p_i > p_j$ means that $v_i$ is closer to $T$ than $v_j$ and is more likely to be a member of the target community. To determine the community size, we sort all nodes in $V_s$ by $p$ in descending order, adding the nodes successively from the beginning of the sorted $V_s$ into the seed-set, and the set with the minimum approximate conductance is regarded as the detected community $C$. In addition, we set an upper bound $b$ for the community size, so we only need to consider candidate communities consisting of $T$ and the first $i \in [1, b]$ nodes in the sorted $V_s$.

**Algorithm 2:** Community Extraction on Sampled Subgraph.

**Input:** Sample subgraph $G_s = (V_s, E_s)$, query nodes $T$, degree array $D$.

**Output:** Extracted community $\hat{C}$.

**Parameters:** Number of hops $k$, community upper bound $b$.

1: $A_s \leftarrow$ adjacency matrix of $G_s$

2: $D_s \leftarrow$ diagonal degree matrix of $A_s$

3: $N_{rw} \leftarrow (I + D_s^{-1}A_s)/2$

4: initialize $p$ by $N_{rw}$

5: sort $p$ by $N_{rw}$ in descending order as $\{v_1, v_2, \ldots, v_{|V_s|}\}$

7: for $i \leftarrow 1$ to $b$ do

8: $\hat{C}_i \leftarrow \{v_j | j \leq i \} \cup T$

9: calculate $\Phi_s(\hat{C}_i)$ by (7)

10: end for

11: $\hat{C} \leftarrow \hat{C}_i$ with the minimum $\Phi_s(\hat{C}_i)$

Details of extracting community on the sampled subgraph $G_s$ are presented in Algorithm 2.

C. **Complexity Analysis**

In this section, we analyze the computational time and space complexity of the proposed algorithm.

1) **Time Complexity:** Our method mainly includes two phases: sampling subgraph in the graph stream and extracting community on the sampled subgraph. According to Algorithm 1, the computational time of each coming edge is bounded by the time of calculating $\text{dist}(\cdot, T)$, and the time complexity of $\text{dist}(\cdot, T)$ is $O(k)$ in the worst case (as discussed in Section III-A). In addition, we need to prune the sampled subgraph after processing every $pc$ edges, and calculate $\text{dist}(\cdot, T)$ $O(|V_s| \log |V_s|)$ times for each pruning operation. So we can sample the subgraph in $O(|E|k + (|E|/pc)(|V_s| \log |V_s|)k)$ for the query nodes. According to Algorithm 2, the computation time is bounded by line 5 and line 7 - 9. In line 5, we need to multiply the transition matrix $N_{rw} \in \mathbb{R}^{|V_s| \times |V_s|}$ and probability vector $p \in [0, 1]^{|V_s|}$ $k$ times, whose time complexity is $O(k|V_s|^2)$. Line 7 - 9 calculate the approximate conductance $b$ times, and the time of each calculation is bounded by $O(b^2)$, so the total is $O(b^3)$.

In summary, the time complexity of our method is $O(k|E|(1 + (|V_s| \log |V_s|)/pc) + k|V_s|^2 + b^3)$. As $k$ is a small integer and $|V_s| < |E|$, $b < |E|$, the time complexity is linear to the stream size $|E|$.

2) **Space Complexity:** In the sampling process shown in Algorithm 1, the degree array $D$ requires $O(V)$ space, and the subgraph requires $O(|V_s| + |E_s|)$ space. During the community extraction on the sampled subgraph, as shown in Algorithm 2, the edge set $E_s$ is replaced by the adjacency matrix $A_s$ with $O(|V_s|^2)$. The probability vector $p$ and storing $b$ approximate conductance values occupy $O(|V_s|)$ and $O(b)$, respectively.
In summary, the space complexity of the proposed method is $O(|V|)$, which satisfies the work memory limitation of graph streams described in Section II-A.

IV. EXPERIMENTS

In this section, we conduct comprehensive experiments to answer the following questions:

- How effective is the approximate conductance?
- What are the effectiveness and efficiency of the proposed method when compared to existing streaming local community detection methods?
- Can the proposed method in graph streams have comparable performance to the non-streaming local community detection methods that run on the static complete graph?
- What about the scalability and preference of our method?
- How to choose appropriate hyper-parameters?

A. Experimental Setup

1) Datasets: To verify the effectiveness and efficiency of our algorithm in real-world scenarios, we adopt five real-world networks with ground truth communities from the Stanford Network Analysis Project (SNAP).

| Network | # Nodes | # Edges | # Communities |
|---------|---------|---------|---------------|
| Amazon  | 334,863 | 925,872 | 936           |
| DBLP    | 317,080 | 1,049,866 | 283          |
| Youtube | 1,134,890 | 2,987,624 | 652          |
| LiveJournal | 3,997,962 | 34,681,189 | 2,159        |
| Orkut   | 3,072,441 | 117,185,083 | 4,530        |

Unless otherwise specified, for all the experiments in this paper, we randomly select 500 ground truth communities for each network as the test cases. For each test case, we randomly select three nodes from the community as the query nodes. If the total number of ground truth communities in a particular network is less than 500, we select all the communities as the test cases.

2) Baselines: To give a well-rounded comparison, we consider two types of local community detection methods.
- **Streaming methods** work in graph streams. To the best of our knowledge, there is only one existing streaming local community detection method called CoEuS [37], which we use to compare with our method.
- **Non-streaming methods** run on static and complete graphs. We select three state-of-the-art local community detection algorithms LEMON [22], LOSP [19] and MWC [39] as the references.

3) Settings: There are four hyper-parameters in our proposed method SCDAC, including the number of hops $k = 4$, pruning cycle $pc = 100,000$, pruning size $ps = 3,000$, and the upper bound of community size $b = 500$. For all the baselines, we keep their default parameter settings, except for changing the minimal communities size of LOSP from 3 to 20, as we discard communities with size less than 20 in the datasets. Following the existing local community detection works [8], [19], [22], [37], we adopt the F1-score between the detected community and the target community to evaluate the detection quality.

The data files from SNAP are organized by edges of the networks in lines. And the graph stream is generated by the edges read line by line from the files. To ensure the validity of the experimental results, each experiment in this work has been repeated five times. Each time we randomly shuffle the order of edges in the graph stream and randomly select the test cases and the query nodes again. And the experimental results are reported in the mean ± standard error of the average of the test cases. All the experiments are conducted on a machine with 2 Intel Xeon CPUs at 2.3GHZ and 256 GB main memory.

B. Evaluation on Approximate Conductance

To verify the effectiveness of approximate conductance, we compare the performance of community extraction using approximate conductance with existing local conductance and ground truth size. Fig. 4 shows the comparison of F1-score on real-world networks, where “g.t.”, “a.c.” and “l.c.” denote the results of detecting communities by cutting the probability distribution vector $p$ with the ground truth size, adopting approximate conductance and adopting locally-calculated conductance as the scoring function, respectively.

The F1-score of adopting the local conductance is pretty low across all real-world networks, implying that this metric, defined on the sampled subgraph from the streaming data, cannot accurately measure the community quality, neither is it suitable as the scoring function for seed-set expansion methods for graph streams. In contrast, the F1-score of adopting approximate conductance is much higher than that of the local conductance and even close to the F1-score of extracting communities by

---

1https://snap.stanford.edu/data/#communities
cutting $p$ with ground truth sizes. Specifically, for all real-world networks, the gap between the two results is no more than 0.09, and the average value of the gap is only 0.06. The experimental results indicate that our proposed approximate conductance can be a good measurement of community quality on the subgraph sampled from the graph streams. It also suggests that our community extraction method can be applied to reveal the target community containing the query nodes in graph streams without the supervision of ground truth size.

C. Comparison With Streaming Method

We conduct experiments on five real-world networks and comprehensively compare our method with a streaming local community detection method CoEuS [37] without the supervision of ground truth size for both effectiveness and efficiency.

1) Effectiveness: Fig. 5 shows the comparative results evaluated by F1-score. It can be observed that our SCDAC method outperforms CoEuS across all networks. Especially on DBLP network, the F1-score of SCDAC exceeds that of CoEuS by 0.10, and on Youtube and Orkut networks, the F1-score of CoEuS is only about three-quarters of that of SCDAC. Compared to CoEuS, SCDAC can reveal the target local community in graph streams more accurately.

2) Efficiency: We report the average running time of detecting a single community in a test case as the efficiency metric. Table II illustrates the comparison between the two methods. We can see that SCDAC is considerably faster than CoEuS on most networks. Specifically, on Amazon network, SCDAC is twice as fast as CoEuS, and on LiveJournal and Orkut networks, SCDAC only takes two-thirds and three-quarters of the corresponding running time of CoEuS, respectively. CoEuS is only faster than SCDAC on the Youtube network, and this is because the Youtube network is sparsely connected (the average degree is only 2, refer to Table I) and the expansion of CoEuS in a greedy way only needs to consider a very limited number of edges.

In summary, experimental results imply that SCDAC can solve the streaming local community detection problem for massive real-world networks and remarkably outperforms the existing streaming local community detection method in terms of effectiveness and efficiency.

D. Comparison With Non-Streaming Method

To further test the proposed method, we compare the performance of SCDAC with three state-of-the-art non-streaming local community detection methods LEMON [22], LOSP [19], and MWC [39] on the same networks. Note that SCDAC works in graph streams, while the other three methods run on static and complete graphs. Due to the strict restrictions in graph streams, SCDAC faces tougher challenges than the methods under the non-streaming scenarios.

Fig. 6 illustrates the comparison of F1-score between SCDAC and the non-streaming methods. SCDAC, LEMON, LOSP and MWC show similar detection accuracy on Amazon, DBLP, and LiveJournal networks. MWC performs best among the non-streaming methods on the other two datasets, and SCDAC achieves almost the same F1-score as MWC on Orkut network, significantly outperforming LEMON and LOSP. On every network, SCDAC is superior to at least one comparison method. These results verify that SCDAC, working in graph streams with access constraints and pretty low memory, can still achieve similar performance to the state-of-the-art non-streaming local community detection method. Therefore, the proposed method is competent for the task of local community detection on massive and natural streaming networks.

E. Scalability Testing

We further analyze the scalability of SCDAC. To this end, we generate several synthetic datasets and analyze the performance of our method in different configurations. For synthetic datasets, we employ the LFR benchmark networks [40], which simulates the properties of real-world networks, including the heterogeneity of node degree and community size distribution.

1) Scalability on Network Scale and Density: To investigate the scalability of SCDAC on the network scale and density, we compare the running time of SCDAC on networks with different numbers of nodes or average degrees. Specifically, for
generating the networks with different network scales, we vary the number of nodes from 200,000 to 2,000,000. For generating the networks with various average degrees, we vary the average degree from 12 to 30. Other parameters follow the basic settings: the number of nodes of 1,000,000, the average degree of 20, the maximum degree of 100, the mixing parameter of 0.1, and the community size is within $[20, 500]$. Fig. 7 shows the average running time of detecting a single community in a test case on networks with different numbers of nodes or different average degrees. With the increase of the network scale, the time cost grows almost linearly, and the trend is similar regarding the change of the average degree. The results are consistent with the time complexity analysis in Section III-C1. In conclusion, SCDAC exhibits good scalability from the perspective of network scale and density.

2) Scalability on Community Size and Density: To investigate the preference of SCDAC for different communities, we compare the F1-scores for detecting communities of different sizes or densities. The experiments are still carried out on the LFR networks, in which the communities with different sizes or densities are selected as the test cases correspondingly. The other parameters remain the same as the previous setting. Fig. 8(a) demonstrates the distribution of F1-scores with respect to the number of nodes in the target community. We can observe that with the growth of the number of nodes, the F1-score shows a slightly downward trend. For communities with less than 100 nodes, the F1-score of SCDAC is around 0.9, while for communities with 400 to 500 nodes, the F1-score of SCDAC decays to about 0.8. The results are acceptable because it is usually harder for algorithms to detect large communities accurately.

The distribution of F1-scores concerning the density of communities is shown in Fig. 8(b), where a community’s density refers to the ratio between the number of existing edges and the number of the node pairs (possible edges) within the community. SCDAC performs better on denser communities. The method samples subgraphs in the stream based on the distance between the query nodes and the others. The closer a community’s members are to the query nodes, the more likely they will be sampled into the subgraph, and the community is easier to be detected. Hence, SCDAC is more accurate in detecting dense communities.

F. Parameter Study

We conduct a study on the four hyper-parameters of our method, as mentioned in Algorithm 1 and Algorithm 2.

1) Number of Hops $k$: The number of hops $k$ directly determines the neighborhood range of the query nodes we consider, but it does not mean that a larger $k$ is better, as the computational cost will also increase when $k$ goes up. As long as $k$ is equal to the diameter of the community (i.e., the maximum pairwise distance among nodes in the community), the sampled subgraph can cover the entire community. And as discussed in [20], the diameter of communities is usually tiny. Thus, a small positive integer is suitable.

To choose an appropriate value for $k$, we vary $k$ from 1 to 8 and evaluate performance by the coverage of the target community. Fig. 9(a) demonstrates the results on Amazon network. When $k \leq 4$, the coverage increases rapidly, and the growth starts to flatten when $k > 4$. To balance the performance and computational cost, $k = 4$ is an appropriate option.

2) Pruning Size $ps$ and Pruning Cycle $pc$: The pruning size $ps$ controls the size of the sampled subgraph for community extraction, and we set $ps = 3,000$ which has been verified and applied in prior works [8], [19]. When SCDAC is used to detect large-scale communities with thousands of members, it is advisable to turn up $ps$ so that the sampled subgraph can cover as many members of the target community as possible. Certainly, as $ps$ increases, the time and space overheads of subgraph sampling will increase as well.

The pruning cycle $pc$ determines the frequency of the pruning process. It is a trade-off for accuracy and efficiency. A larger $pc$ brings a more accurate local structure around the query nodes in the subgraph. In the meantime, when $pc$ increases, the number of pruning operations decreases, while the subgraph grows larger...
before pruning, leading to greater computational cost of each pruning process. We conduct experiments on DBLP network by varying $pc$ from 10,000 to 500,000, and the results are shown in Fig. 9(b). We can observe that SCDAC exhibits the lowest running time when the pruning cycle is in the range of 50,000 to 100,000, and the value of $pc = 100,000$ is slightly lower.

Moreover, $pc$ also affects the memory overhead and accuracy. When SCDAC runs on a machine with very limited memory, it is advisable to turn down $pc$ to reduce the memory cost. And when memory is relatively sufficient, a larger $pc$ can be chosen to obtain more accurate results.

3) Upper Bound of Community Size $b$: Leskovec et al. [41] observe that communities in real-world networks with high quality are small and usually contain no more than 100 nodes. Thus the value of $b$ does not need to be very large, or it will cause unnecessary computational overheads (see line 7 – 10 in Algorithm 2). We statistically analyze the distribution of community sizes in real-world networks, as shown in Table III. There are at least 97% of communities whose sizes do not exceed 100, except for the DBLP network (92%). And there are at least 97% of communities whose sizes do not exceed 500 in all networks. Therefore, we set the upper bound of community size to 500.

V. CONCLUSION

We propose a novel online, single-pass method for addressing the local community detection in graph streams. The key issue is how we could sample as many edges close to the query nodes as possible in the subgraph under the memory limit and access constraint. We use a distance tree that could be re-organized quickly when a coming edge is added to the subgraph, and address subgraph pruning periodically to have a good trade-off for accuracy and efficiency during the sampling process. As edges are coming in arbitrary order, another challenge is how to design a metric that could measure community quality. We record the node degree in the stream, and introduce a new metric called approximate conductance to determine the community boundary on the subgraph. Extensive experiments on real-world datasets show that our method outperforms the existing streaming local community detection method with higher accuracy and shorter running time, and even achieves similar performance compared to the state-of-the-art non-streaming local community detection methods that work on static and complete graphs. Experimental results also demonstrate that the proposed method is scalable on the network scale and density, and is robust in the detection accuracy for different community sizes and densities. Further studies show that the proposed method is not sensitive to the parameters and exhibits good performance on a suitable range for the parameter values.

Streaming local community detection is a new research direction for massive graph data, and there are still many problems to be explored, e.g., the phenomenon that some nodes belong to multiple communities simultaneously. In future work, we will extend our method to solve more tasks in graph streams.

**REFERENCES**

[1] M. E. Newman and M. Girvan, “Finding and evaluating community structure in networks,” *Phys. Rev. E*, vol. 69, no. 2, 2004, Art. no. 026113.

[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *J. Statist. Mechanics: Theory Experiment*, vol. 2008, no. 10, 2008, Art. no. P10008.

[3] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, “Finding statistically significant communities in networks,” *PLoS One*, vol. 6, no. 4, 2011, Art. no. e18961.

[4] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi, “DEMON: A local-first discovery method for overlapping communities,” in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, Beijing, China, 2012, pp. 615–623.

[5] D. He et al., “A joint community detection model: Integrating directed and undirected probabilistic graphical models via factor graph with attention mechanism,” *IEEE Trans. Big Data*, vol. 8, no. 4, pp. 994–1006, Aug. 2022.

[6] W. Cui, Y. Xiao, H. Wang, Y. Lu, and W. Wang, “Online search of overlapping communities,” in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, New York, NY, USA, 2013, pp. 277–288.

[7] K. Kloster and D. F. Gleich, “Heat kernel based community detection,” in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, New York, NY, USA, 2014, pp. 1386–1395.

[8] K. He, Y. Sun, D. Bindel, J. Hopcroft, and Y. Li, “Detecting overlapping communities from local spectral subspaces,” in *Proc. IEEE Int. Conf. Data Mining*, Atlantic City, NJ, USA, 2015, pp. 769–774.

[9] D. Luo, Y. Bian, Y. Yan, X. Liu, J. Huan, and X. Zhang, “Local community detection in multiple networks,” in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 266–274.

[10] I. Weber, V. R. K. Garimella, and A. Batayneh, “Secular vs. Islamist polarization in Egypt on Twitter,” in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, Niagara, ON, Canada, 2013, pp. 290–297.

[11] P. Liakos, K. Papakonstantinoupolou, and A. Delis, “Realizing memory-optimized distributed graph processing,” *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 4, pp. 743–756, Apr. 2018.

[12] N. Ahmed and N. Duffield, “Adaptive shrinkage estimation for streaming graphs,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 10595–10600.

[13] Z. Bar-Yossef, R. Kumar, and D. Sivakumar, “Reductions in streaming algorithms, with an application to counting triangles in graphs,” in *Proc. 13th Annu. ACM-SIAM Symp. Discrete Algorithms*, San Francisco, CA, USA, 2002, pp. 623–632.

[14] N. K. Ahmed, N. Duffield, T. Willke, and R. A. Rossi, “On sampling from massive graph streams,” in *Proc. VLDB Endowment*, vol. 10, no. 11, pp. 1430–1441, 2017.

[15] R. Etanemadi and J. Lu, “PES: Priority edge sampling in streaming triangle estimation,” *IEEE Trans. Big Data*, vol. 8, no. 2, pp. 470–481, Apr. 2022.

[16] B. Bahanini, R. Kumar, and S. Vassilvitskii, “Densest subgraph in streaming and MapReduce,” in *Proc. VLDB Endowment*, vol. 5, no. 5, pp. 454–465, 2012.

[17] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu, “Querying k-true community in large and dynamic graphs,” in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Snowbird, UT, USA, 2014, pp. 1311–1322.

[18] N. Veldt, D. Gleich, and M. Mahoney, “A simple and strongly-local flow-based method for cut improvement,” in *Proc. 33rd Int. Conf. Mach. Learn.*, New York City, NY, USA, 2016, pp. 1938–1947.

[19] K. He, P. Shi, D. Bindel, and J. E. Hopcroft, “Krylov subspace approximation for local community detection in large networks,” *ACM Trans. Knowl. Discov. Data*, vol. 13, no. 5, pp. 1–30, 2019.

[20] A. Conte, T. De Matteis, D. De Sensi, R. Grossi, A. Marino, and L. Versari, “D2K: Scalable community detection in massive networks via small-diameter k-plexes,” in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, London, U.K., 2018, pp. 1272–1281.

---

**TABLE III**

| Network | $[0, 100)$ | $[100, 500)$ | $[500, 1000)$ | $> 1000$ |
|---------|------------|--------------|---------------|---------|
| Amazon  | 97.77%     | 1.56%        | 0.28%         | 0.39%   |
| DBLP    | 92.88%     | 4.68%        | 1.33%         | 1.11%   |
| Youtube | 99.17%     | 0.74%        | 0.07%         | 0.02%   |
| LiveJournal | 98.59% | 1.19%        | 0.14%         | 0.08%   |
| Orkut   | 99.41%     | 0.47%        | 0.06%         | 0.06%   |
[21] R. Andersen, F. Chung, and K. Lang, “Local graph partitioning using PageRank vectors,” in Proc. 47th Annu. IEEE Symp. Found. Comput. Sci., Berkeley, CA, USA, 2006, pp. 475–486.

[22] Y. Li, K. He, D. Bindel, and J. E. Hopcroft, “Uncovering the small community structure in large networks: A local spectral approach,” in Proc. 24th Int. Conf. World Wide Web, Florence, Italy, 2015, pp. 658–668.

[23] R. Kannan, S. Vempala, and A. Vetta, “On clusterings: Good, bad and spectral,” J. ACM, vol. 51, no. 3, pp. 497–515, 2004.

[24] M. E. Newman, “Modularity and community structure in networks,” in Proc. Nat. Acad. Sci. USA, vol. 103, no. 23, pp. 8577–8582, 2006.

[25] A. McGregor, “Graph stream algorithms: A survey,” ACM SIGMOD Rec., vol. 43, no. 1, pp. 9–20, 2014.

[26] P.-Z. Li, L. Huang, C.-D. Wang, and J.-H. Lai, “EdMot: An edge enhancement approach for motif-aware community detection,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, Anchorage, AK, USA, 2019, pp. 479–487.

[27] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang, “On graph problems in a semi-streaming model,” Theor. Comput. Sci., vol. 348, no. 2/3, pp. 207–216, 2005.

[28] K. J. Ahn, S. Guha, and A. McGregor, “Analyzing graph structure via linear measurements,” in Proc. Annu. ACM-SIAM Symp. Discrete Algorithms, Kyoto, Japan, 2012, pp. 459–467.

[29] C. C. Aggarwal, Y. Zhao, and S. Y. Philip, “Outlier detection in graph streams,” in Proc. IEEE 27th Int. Conf. Data Eng., Hannover, Germany, 2011, pp. 399–409.

[30] P. Zhao, C. Aggarwal, and G. He, “Link prediction in graph streams,” in Proc. IEEE 32nd Int. Conf. Data Eng., Helsinki, Finland, 2016, pp. 553–564.

[31] N. Tang, Q. Chen, and P. Mitra, “Graph stream summarization: From big bang to big crunch,” in Proc. Int. Conf. Manage. Data, San Francisco, CA, USA, 2016, pp. 1481–1496.

[32] A. Khan and C. Aggarwal, “Query-friendly compression of graph streams,” in Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining, San Francisco, CA, USA, 2016, pp. 130–137.

[33] X. Gou, L. Zou, C. Zhao, and T. Yang, “Fast and accurate edge graph stream summarization,” in Proc. IEEE 35th Int. Conf. Data Eng., Macao, China, 2019, pp. 1118–1129.

[34] S.-Y. Yun, M. Lelarge, and A. Proutiere, “Streaming, memory limited algorithms for community detection,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2014, pp. 3167–3175.

[35] A. Hollocou, J. Maudet, T. Bonald, and M. Lelarge, “A linear streaming algorithm for community detection in very large networks,” 2017, arXiv: 1703.02955.

[36] Y. Wu et al., “Streaming belief propagation for community detection,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2021, pp. 26 976–26 988.

[37] P. Liakos, K. Papakonstantinopoulou, A. Noulas, and A. Delis, “Rapid detection of local communities in graph streams,” IEEE Trans. Knowl. Data Eng., vol. 34, no. 5, pp. 2375–2386, May 2022.

[38] J. Yang and J. Leskovec, “Defining and evaluating network communities based on ground-truth,” Knowl. Inf. Syst., vol. 42, no. 1, pp. 181–213, 2015.

[39] Y. Bian, J. Ni, W. Cheng, and X. Zhang, “The multi-walker chain and its application in local community detection,” Knowl. Inf. Syst., vol. 60, no. 3, pp. 1663–1691, 2019.

[40] A. Lanchinietti, S. Fortunato, and F. Radicchi, “Benchmark graphs for testing community detection algorithms,” Phys. Rev. E, vol. 78, no. 4, 2008, Art. no. 046110.

[41] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, “Statistical properties of community structure in large social and information networks,” in Proc. 17th Int. Conf. World Wide Web, Beijing, China, 2008, pp. 695–704.

Meng Wang received the BS degree in applied mathematics from the Wuhan University of Technology, Wuhan, China, in 2019. He is currently working toward the PhD degree from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan. His research interests include machine learning and social networks.

Yan Hao Yang received the master’s degree in computer science from the Huazhong University of Science and Technology, Wuhan, China, in 2022. His research interests include machine learning and social networks.

David Bindel received the PhD degree in computer science from the University of California at Berkeley, in 2006. He is currently an associate professor of computer science and the director of Center for Applied Math, Cornell University. His research interests include numerical linear algebra, scientific computing, high-performance computing, spectral network analysis methods, optimization via surrogate models, and finite element analysis.

Kun He (Senior Member, IEEE) received the PhD degree in system engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2006. She is currently a professor with the School of Computer Science and Technology, Huazhong University of Science and Technology. She had been with the Department of Management Science and Engineering, Stanford University in 2011–2012 as a visiting researcher. She had been with the Department of Computer Science, Cornell University, New York in 2013–2015 as a visiting associate professor, in 2016 as a visiting professor, and in 2018 as a visiting professor. She was honored as a Mary Shepard B. Upson visiting professor for the 2016–2017 Academic year in engineering, Cornell University. Her research interests include adversarial learning, representation learning, social network analysis, and combinatorial optimization.