Sketch-based community detection in evolving networks

Andre Beckus[1,*] and George K. Atia[1,2,†]

[1]*Department of Electrical and Computer Engineering, University of Central Florida, Orlando, Florida 32816, USA*

[2]*Department of Computer Science, University of Central Florida, Orlando, Florida 32816, USA*

(Received 10 July 2022; accepted 8 September 2022; published 24 October 2022)

We consider an approach for community detection in time-varying networks. At its core, this approach maintains a small sketch graph to capture the essential community structure found in each snapshot of the full network. We demonstrate how the sketch can be used to explicitly identify six key community events which typically occur during network evolution: growth, shrinkage, merging, splitting, birth, and death. Based on these detection techniques, we formulate a community detection algorithm which can process a network concurrently exhibiting all processes. One advantage afforded by the sketch-based algorithm is the efficient handling of large networks. Whereas detecting events in the full graph may be computationally expensive, the small size of the sketch allows changes to be quickly assessed. A second advantage occurs in networks containing clusters of disproportionate size. The sketch is constructed such that there is equal representation of each cluster, thus reducing the possibility that the small clusters are lost in the estimate. We present a new standardized benchmark based on the stochastic block model which models the addition and deletion of nodes, as well as the birth and death of communities. When coupled with existing benchmarks, this new benchmark provides a comprehensive suite of tests encompassing all six community events. We provide analysis and a set of numerical results demonstrating the advantages of our approach both in runtime and in the handling of small clusters.

DOI: 10.1103/PhysRevE.106.044306

I. INTRODUCTION

The detection of community structure in networks has garnered a great deal of attention, leading to a vast array of algorithms. Much of the focus has been on static networks, where the goal is to identify groups of nodes within which connections are dense and between which connections are relatively sparse. However, it is often the case that networks evolve with time. For example, edges in social media networks appear and disappear to reflect ever-changing friendships, and gene expression networks continuously evolve in response to external stimuli [1,2]. In this dynamic setting, new sequential algorithms are needed to track the community structure underlying each temporal snapshot of the network. Here we propose a sketch-based approach.

Sketching involves the construction of a small synopsis of a full dataset [3]. Notably, this technique has been used in static community detection [4,5], where a sketch subgraph is generated by sampling nodes from the full network. The sketch is clustered using an existing community detection algorithm, and the community membership of the nodes in the full network are inferred based on the estimated communities in the sketch. Here we propose the use of an evolving sketch to detect and handle the six canonical community events observed in dynamic networks [6]: growth, shrinkage, merging, splitting, birth, and death. This dynamic approach addresses two pervasive issues in community detection.

One important concern in community detection is the ability to process large graphs. Many static methods become infeasibly slow when processing a large network, thus motivating a search for efficient algorithms [7]. The extra time dimension inherent to the dynamic setting only makes this search for efficiency more pressing. However, time-evolving networks also offer a distinct advantage not found in the static domain. Specifically, evolving networks often possess temporal smoothness in which the community structure changes gradually [8]. In this case, previous snapshots offer prior information which can aid in the clustering of subsequent snapshots. We present a method which relies on a small sketch to convey information regarding previous snapshots. By using a small sketch, the algorithm can detect the main community events without requiring the full graph to be examined, thus reducing the required computational complexity. If the sketch size and number of clusters are fixed, then the complexity of our algorithm scales linearly in network size.

Another typical issue found in community detection is the detection of small clusters [9]. If a community shrinks too small, then it may become lost, i.e., the community may be absorbed into a larger community in the estimated partition. We show that once a community is captured in the sketch, it can be tracked even if the community becomes very small.

We use dynamic benchmarks as a means for evaluating the proposed algorithm with respect to the canonical network events. The first four events are included in the benchmarks of Ref. [10], which are based on the well-known stochastic block model (SBM) [11]. Here we propose a new dynamic SBM benchmark which captures the last two events of birth and death. An important feature of this proposed benchmark is that the size of the network varies with time, a characteristic

[*]abeckus@knights.ucf.edu

[†]george.atia@ucf.edu

not found in the existing benchmarks. In addition to modeling the birth event, this benchmark incrementally adds new nodes to the network which join existing communities, a feature also not seen in Ref. [10].

This paper is organized as follows. In Sec. II, we summarize existing community detection algorithms for evolving networks. Section III describes the network model, and Sec. IV summarizes SBM benchmarks which capture key evolutionary processes. In Sec. V, we describe the sketch-based approach and formulate techniques by which sketches can detect events and track evolutionary processes. Section VI presents the proposed algorithm based on these tracking techniques. We analyze the algorithm in Sec. VII, present numerical results in Sec. VIII, and conclude in Sec. IX. Appendix A describes the static clustering used as a part of the main algorithm, and Appendix B provides details on the main algorithm itself. Appendix C derives the results found in the analysis of Sec. VII. Appendix D provides additional details regarding the algorithms we compare against in the numerical results.

II. RELATED WORK: COMMUNITY DETECTION IN EVOLVING NETWORKS

A number of algorithms have been proposed for community detection in evolving networks (see Refs. [8,12] for comprehensive surveys). One straightforward approach entails the independent clustering of each snapshot using a static clustering algorithm. The communities in the current snapshot are matched to the previous communities such that there is continuity in the community identities. This category of algorithm contains a number of variants beginning with the classic work of Ref. [13].

More recently, many algorithms take a more sophisticated "dependent" approach, in which previous snapshots are accounted for in the clustering of the current snapshot. These algorithms have the potential to outperform independent community detection algorithms, since they incorporate previous knowledge *directly* in the clustering step.

One approach commonly seen in this category is the representation of each snapshot using a compact graph. In Ref. [14], a small weighted graph is constructed after clustering a given snapshot, with each community represented by a single "supernode." The weights of the edges between supernodes indicate the cumulative number of edges between the corresponding communities. These supernodes are then incorporated into the next snapshot's graph, thus carrying forward information from the previous estimates. A similar idea can be seen in dynamic methods built around the static Louvain algorithm [15], for example as seen in Ref. [16]. The extension of the Louvain algorithm to time-varying networks follows naturally from its reliance on supernodes. Our approach also uses a small representative graph, however, using an altogether different idea of sketching, as described in Sec. V.

The model used in this paper is based on the SBM [11]. Several recent algorithms have been developed based on dynamic SBM-based models. The dynamic models of Refs. [17,18] specify that nodes move between a fixed set of communities according to a stationary transition probability matrix. In addition to allowing the movement of nodes between communities, the models of Refs. [19,20] also allow the edge probabilities of the communities to vary. Nonetheless, these works focus on the case where individual nodes only change community membership, i.e., the communities undergo the grow and shrink processes. Although Ref. [21] is able to track communities which are also merging and splitting, it still does not allow varying numbers of nodes across the snapshots. The algorithm of Ref. [22] allows nodes to join or leave the graph but requires that all snapshots be known when invoking the algorithm. We emphasize that our proposed algorithm is online in nature, i.e., it performs community detection iteratively on one snapshot at a time, while carrying forward the clustering results from previous snapshots.

III. TEMPORAL NETWORK MODEL

At time t, the network snapshot is represented by graph $G(t) = (V(t), E(t))$, where $V(t)$ is the set of nodes in existence at time t, and $E(t)$ is the set of edges between these nodes. Let $\widehat{\mathcal{C}}(t) = \{\widehat{C}_u(t) \mid u \in \{1, \dots, \widehat{q}(t)\}\}$ be the partition at time t, with $\widehat{C}_u(t)$ denoting the set of nodes in community u, and $\widehat{q}(t)$ the number of communities.

In each snapshot, an edge exists between nodes within a community with probability p_{in}. Nodes in community u are connected to nodes in a different community u' with probability $p_{u,u'}(t)$. An evolutionary process may vary the intercommunity edge density $p_{u,u'}(t)$ so long as the resulting graph adheres to the SBM. We discuss one such process in Sec. IV B. A pair of communities u, u' are considered to be merged if

$$p_{\text{in}} - p_{u,u'}(t) < \sqrt{\frac{2[p_{\text{in}} + p_{u,u'}(t)]}{|\widehat{C}_u(t)| + |\widehat{C}_{u'}(t)|}}. \tag{1}$$

When communities are of equal size, this condition corresponds to the asymptotic weak detectability limit (see Ref. [10] for a discussion of this bound in the context of merging and splitting communities). For simplicity, here, we average the community sizes when they are of unequal size.

The following events may occur at time t.

(i) **Node movement between communities** A set of nodes $V_{u\to u'}(t)$ belonging to community u may move to community u'. The edges connected to these nodes are regenerated according to the SBM based on the new community memberships.

(ii) **New nodes and community birth** A set of nodes $V^+(t)$ may join the graph. A subset $V_{\text{birth}}(t) \subseteq V^+(t)$ of these nodes join new communities. The remaining nodes $V^+(t) \setminus V_{\text{birth}}(t)$ join existing communities. The edges of nodes in $V^+(t)$ are generated according to the SBM.

(iii) **Removed nodes and community death** A set of nodes $V^-(t)$ may be removed from the graph. Death occurs when all nodes in a particular community are removed.

(iv) **Merge and split of communities** The merge event occurs for communities u, u' when $p_{u,u'}(t)$ increases such that (1) becomes true. Likewise, the split event occurs when $p_{u,u'}(t)$ decreases such that (1) becomes false.

Note that many of the model variables are functions of time t. Where there is no ambiguity, we omit this time parameter to simplify the exposition.

IV. EVOLUTIONARY PROCESSES: BENCHMARKS

For the purpose of illustrating and analyzing the proposed algorithm, we consider here specific examples of evolutionary processes. These are realized by four benchmark networks, i.e., parameterized sequences of snapshots with known community partitions for validating and comparing community detection algorithms. The grow-shrink and merge-split benchmarks are defined in Ref. [10], whereas we present the birth-death process here for the first time.

Each benchmark consists of an evolving network containing $2n$ total nodes. The underlying process is driven by a periodic triangular waveform

$$x(t) = \begin{cases} 2t^*, & 0 \leqslant t^* < 1/2, \\ 2 - 2t^*, & 1/2 \leqslant t^* < 1, \end{cases} \tag{2}$$

where

$$t^* \equiv (t/\tau + \phi) \bmod 1, \tag{3}$$

τ is the period of the waveform, and ϕ controls the phase of the waveform. We will assume that $\phi = 0$ unless otherwise specified.

A. Grow-shrink benchmark

The grow-shrink benchmark moves nodes between a pair of communities denoted A and B, thus growing and shrinking the communities. At each time step the first community contains

$$n_A = n - nf[2x(t + \tau/4) - 1] \tag{4}$$

nodes, whereas the second community contains $n_B = 2n - n_A$ nodes. Nodes lost from the first community are transferred to the second community, and vice versa. The parameter $f \in [0, 1]$ controls the variation in community sizes. For $t \in \{0, \tau/2, \tau\}$ the sizes of the communities are equal. At time $t = \tau/4$, a fraction f of nodes in community A will have moved to community B, whereas at time $t = 3\tau/4$ the opposite holds.

B. Merge-split benchmark

The merge-split benchmark has two communities denoted A and B, each of size n, with intracommunity edge density p_{in}. Initially, the intercommunity edge density is $p_{A,B}(0) = p_{\text{out}}$. New edges are gradually added between the two communities until they are completely merged at time $t = \tau/2$ with $p_{A,B}(\tau/2) = p_{\text{in}}$. Then the process reverses and the new edges are removed until the communities are completely split again at time $t = \tau$.

The intercommunity edges are placed in the following way. The number of intercommunity edges m_{um} in the unmerged state are drawn according to a binomial distribution with parameters n^2 and p_{out}. The number of edges m_m in the merged state is similarly drawn, except using probability p_{in}. The number of edges at time t is then determined by

$$m^*(t) = [1 - x(t)]m_{\text{um}} + x(t)m_m, \tag{5}$$

where the edges are placed uniformly at random. In this way, the edge density between the two communities is $p_{A,B}(t) = m^*(t)/n^2$. The communities are considered merged at the detectability limit (1).

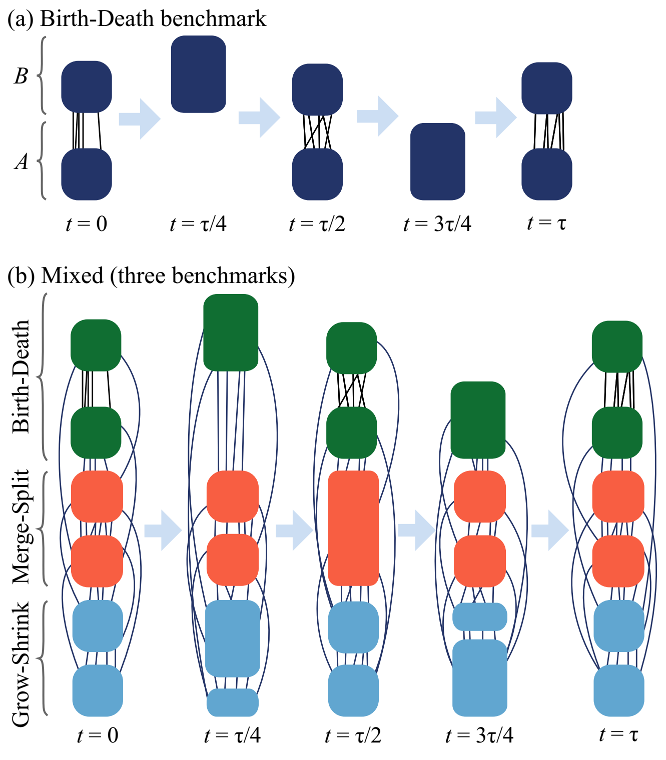

FIG. 1. (a) Schematic representation of the birth-death benchmark, showing the two communities labeled A and B. (b) Schematic representation of the mixed benchmark, which stacks the grow-shrink, merge-split, and birth-death benchmarks.

C. Birth-death benchmark

We now propose a new benchmark which realizes the birth and death of communities, as well as the addition and removal of nodes from the network. A schematic diagram of the birth-death benchmark is shown in Fig. 1(a). The benchmark contains two communities which pass into and out of existence. The size of the first community is

$$n_A = \begin{cases} 0, & x(t + \tau/4) \geqslant 1 - \gamma/2, \\ n[1 - x(t + \tau/4)], & \text{otherwise}, \end{cases} \tag{6}$$

where $n_A = 0$ designates a nonexistent community and parameter $\gamma \in [0, 1]$ controls the minimum size of the community. The community starts at time $t = 0$ with $n/2$ nodes. Nodes are removed from the network, until the community shrinks to size $\gamma n/2$ at time $t = \tau(1 - \gamma)/4$. At this point, the community dies and all of its remaining nodes are deleted from the network. At time $t = \tau(1 + \gamma)/4$, a new set of $\gamma n/2$ nodes is added to the network and used to recreate the community. New nodes are gradually created and added to the community until it reaches size n. At this point, nodes are again removed from the community until it contains $n/2$ nodes, and the process repeats.

The second community is of size

$$n_B = \begin{cases} 0, & x(t + \tau/4) < \gamma/2 \\ n\,x(t + \tau/4), & \text{otherwise}. \end{cases} \tag{7}$$

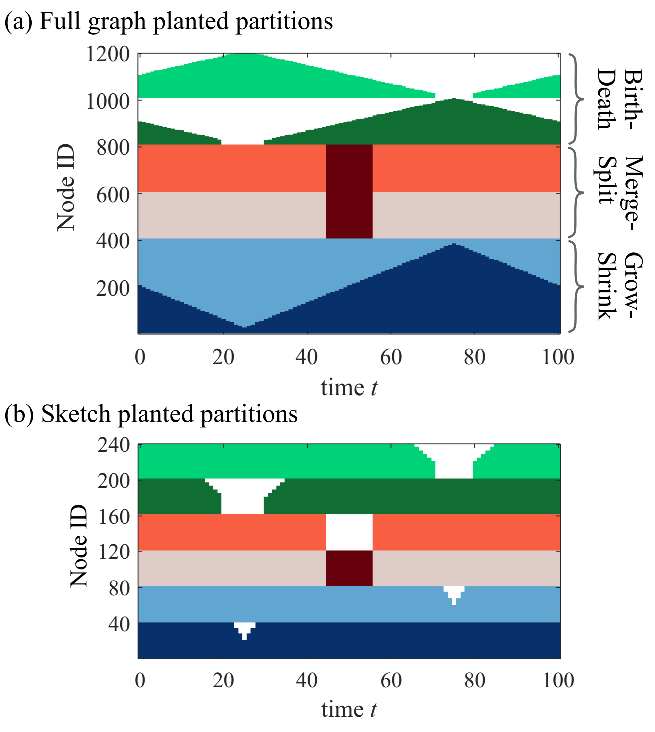

FIG. 2. (a) Planted partitions for full graphs of the mixed benchmark network. Each vertical slice indicates the planted partition at time t. The model parameters are $n = 200$, $\widehat{q} = 6$, $f = 0.9$, and $\gamma = 0.2$. (b) Sketches produced with $n' = 40$. Each vertical slice indicates the planted partitions in sketch $\mathcal{S}(t)$. White regions indicate that the corresponding node does not exist at time t.

This community undergoes essentially the same process as the first community except with a phase shift of $\tau/2$.

D. Mixed benchmark

To model concurrent processes capturing all of the events, we present a mixed benchmark which is created by "stacking" the grow-shrink, merge-split, and birth-death benchmarks. A schematic of this mixed benchmark is shown in Fig. 1(b). The benchmark has a maximum of $6n$ nodes. The first $4n$ nodes contain the grow-shrink and merge-split benchmarks as previously described, whereas the last $2n$ nodes participate in the birth-death process (the actual number of nodes varies with time due to addition and deletion of nodes in the birth-death benchmark). We show an example of this mixed benchmark in Fig. 2(a).

V. SKETCH-BASED TRACKING OF EVOLUTIONARY PROCESSES

Our algorithm relies on a small representative sketch of the full network. The sketch captures important information which can be used to detect network events and track the processes by which the network evolves. Meanwhile, the smaller size of the sketch allows these checks to be performed quickly without requiring a complete assessment of the entire network. The estimates of the communities in snapshot t are $\mathcal{C}(t) = \{C_u(t) \mid u \in \{1, \ldots, q(t)\}\}$, where $q(t)$ is the estimated number of communities at time t. We first describe the sketch, and then describe how this sketch can be used to detect specific events.

The sketch consists of a set of nodes sampled from the full network. At each time step, this set is updated such that it contains an equal number of nodes n' from each community. The set of nodes in the sketch at time t is denoted $\mathcal{S}(t)$, and the subset of these nodes from community u is denoted $C'_u(t) = \mathcal{S}(t) \cap C_u(t)$. We refer to this as sketch community u. An example sketch time series is shown in Fig. 2(b), where nodes have been sampled from the mixed benchmark shown in Fig. 2(a).

For this example, we build the sketches using knowledge of the planted community partitions. The proposed algorithm has no such knowledge, and therefore must build the sketches based on estimates of the true communities. We will present an actual sketch produced by the proposed algorithm in Sec. VIII D.

A. Inferring community membership of nodes

We show in this section how the sketch may be used to infer community membership of any node $i \in V(t)$ in network snapshot $G(t)$. To this end, we calculate

$$s_{i,v}(t) = \frac{|\{(i,j) \in E(t) \mid j \in C'_v(t-1)\}|}{|C'_v(t-1)|} \tag{8}$$

to evaluate the connectivity of node i to each sketch community v. Let u be the true community assignment of node i. Since

$$\mathbb{E}[s_{i,v}(t)] = \begin{cases} p_{\text{in}}, & v = u, \\ p_{u,v}(t), & v \neq u, \end{cases} \tag{9}$$

it follows that $s_{i,v}(t)$ provides a point estimate of the probability that there is an edge between node i and any node $j \in C'_v(t-1)$. Node i can then be assigned to the community u' with which connectivity is greatest, i.e., where

$$u' = \arg\max_{1 \leqslant v \leqslant q(t)} s_{i,v}(t). \tag{10}$$

The proposed algorithm uses (10) to assign communities to new nodes joining the network, as well as to identify nodes which have changed community membership.

We finish this section by noting that the variance in $s_{i,v}(t)$ is

$$\text{Var}(s_{i,v}(t)) = \begin{cases} \frac{p_{\text{in}}(1-p_{\text{in}})}{|C'_v(t-1)|}, & v = u, \\ \frac{p_{u,v}(t)[1-p_{u,v}(t)]}{|C'_v(t-1)|}, & v \neq u. \end{cases} \tag{11}$$

The variance grows as the sketch communities shrink, thus motivating the use of equal-sized communities in the sketch.

B. Detecting the split event

Suppose that community u is undergoing a split into two separate communities u and u'. To detect the emerging clusters we can use the spectrum of the nonbacktracking matrix as described in Ref. [23]. Let G_u be the subgraph of $G(t)$ induced by the latest estimate $C_u(t-1)$, and $\mathbf{A}$ be the adjacency matrix of G_u. Given diagonal matrix $\mathbf{D}$ containing the degrees of nodes in $\mathbf{A}$, and identity matrix $\mathbf{I}$, define

$$\mathbf{B}' = \begin{pmatrix} \mathbf{0} & \mathbf{D}-\mathbf{I} \\ -\mathbf{I} & \mathbf{A} \end{pmatrix}. \tag{12}$$

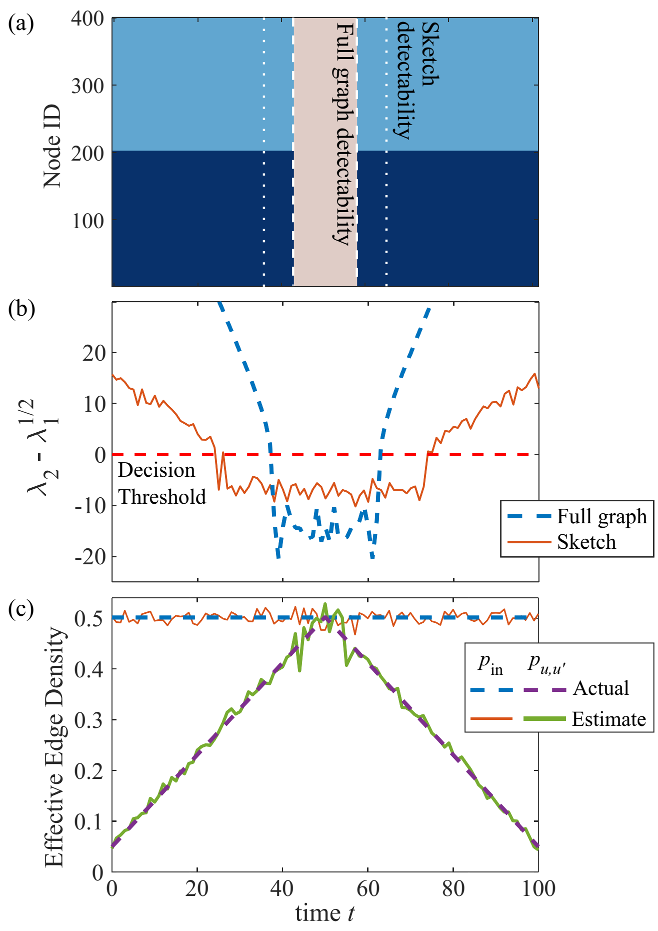

FIG. 3. (a) Planted partitions of the merge-split benchmark. For reference, the detectability limits that formally define the splits in the benchmark are shown as vertical dashed and dotted lines. Network parameters are $\widehat{q} = 2$, $n = 200$, $p_{\text{in}} = 0.5$, and $p_{\text{out}} = 0.05$. (b) Actual and estimated values of $\lambda_2 - \sqrt{\lambda_1}$ at each time step. When the gap is greater than the decision threshold (horizontal dashed red line), the community is considered split. The sketch is constructed using $n' = 50$ nodes sampled uniformly at random from each of the two communities at each time step. (c) Actual and estimated values of $p_{\text{in}}, p_{u,u'}$ at each time step.

Suppose the emerging communities are each of size n, and define λ_1, λ_2 as the largest and second largest eigenvalues of $\mathbf{B}'$, respectively. If

$$n[p_{\text{in}} - p_{u,u'}(t)]^2 > p_{\text{in}} + p_{u,u'}(t), \tag{13}$$

then in the limit as $n \to \infty$ with np_{in} and $np_{u,u'}(t)$ constant, $\lambda_1 \to n[p_{\text{in}} + p_{u,u'}(t)]$ and $\lambda_2 \to n[p_{\text{in}} - p_{u,u'}(t)]$ such that [23]

$$\lambda_2 > \sqrt{\lambda_1}. \tag{14}$$

Although condition (14) is only valid in the limit of infinite sized graphs, it can still serve as a reliable split indicator for a given sequence of network realizations. We show an example of this in Fig. 3. The planted partitions are shown in Fig. 3(a), and the dashed blue line in Fig. 3(b) shows the corresponding gap $\lambda_2 - \sqrt{\lambda_1}$ for each time step. The value of this gap increases as the process moves in either direction away from the fully merged state at $t = 50$, and toward the fully split states at $t \in \{0, \tau\}$. Decision threshold (14) is shown as a horizontal dashed line. As can be seen, the split is detected fairly close to the full graph detectability limit.

Rather than calculating the eigenvalues for the full network (at great computational cost), we propose to instead detect the split using the sketch. We apply the same procedure as described above, but instead substitute $C_u(t-1)$ with $C'_u(t-1)$. The estimate based on the sketch is shown in Fig. 3(b) as a solid orange line. Note that the time of detection in the sketch diverges from that in the full graph as the community sizes in the sketch decrease. We analyze this dependence on sketch size in Sec. VII.

C. Detecting the merge event

Suppose that communities u, u' are merging. To detect the merge event, we exploit the fact that the two communities are already known at time $t-1$. This means that we can estimate p_{in} and $p_{u,u'}(t)$ and use these estimates to directly check condition (1) to detect a merge event. The sketch allows us to quickly calculate point estimates of the edge probabilities using the expressions

$$\widehat{p}_{\text{in}} = \frac{2\sum_{v=1}^{q(t)} |\{(i,j) \in E(t) \mid i,j \in C'_v(t-1)\}|}{\sum_{v=1}^{q(t)} |C'_v(t-1)|(|C'_v(t-1)|-1)}, \tag{15}$$

$$\widehat{p}_{u,u'}(t) = \frac{|\{(i,j) \in E(t) \mid i \in C'_u(t-1),\, j \in C'_{u'}(t-1)\}|}{|C'_u(t-1)||C'_{u'}(t-1)|}. \tag{16}$$

Figure 3(c) shows the actual (dashed blue line) and estimated (solid orange line) values of p_{in} for the example in Fig. 3(a). The actual (dashed purple line) and estimated (solid green line) values of $p_{u,u'}(t)$ are also shown in the sample plot. In both cases, the estimates track the actual values well.

D. Detecting the birth event

Consider a node $i \in V^+(t)$, which is joining the network. If the node joins an existing community u, then $\mathbb{E}[s_{i,u}(t)] = p_{\text{in}}$, and we can detect this occurrence by checking if $s_{i,u}(t) \geqslant \widehat{p}_{\text{in}} - 3\widehat{\sigma}$, where $\widehat{p}_{\text{in}}$ is the estimate from (15), and $\widehat{\sigma} = \sqrt{\widehat{p}_{\text{in}}(1-\widehat{p}_{\text{in}})/n'}$ is an estimate of the standard deviation of $s_{i,u}(t)$. On the other hand, if $i \in V_{\text{birth}}(t)$, then the expectation $\mathbb{E}[s_{i,v}(t)]$ will equal the intercommunity edge density between the new community and *any* existing community v. This suggests that we can identify the set of nodes that are joining newborn communities using the expression

$$\hat{V}_{\text{birth}}(t) = \{i \in V^+(t) \mid s_{i,v}(t) < \widehat{p}_{\text{in}} - 3\widehat{\sigma},\ \forall v \in \{1, \ldots, q(t)\}\}. \tag{17}$$

VI. PROPOSED ALGORITHM

We first discuss preliminaries. The proposed algorithm invokes a function **StaticCluster**(G), which performs clustering of a static graph G to produce community estimates $\mathcal{C} = \{C_1, \ldots, C_q\}$. We implement this function using spectral techniques based on the nonbacktracking matrix [23], along with enhancements to provide more robust estimation of the number of communities (details of the function are given in Appendix A). The computational cost of this function is dominated by the eigendecomposition, which is cubic in the size of graph G. We now summarize the main steps of the proposed algorithm. We provide an assessment of the computational complexity for each step, and comment on the overall complexity at the end. A detailed algorithm listing is provided in Appendix B.

MainAlgorithm

Input: Initial sketch size N'. Sketch community size n'.
Graph snapshots $G(t), t = 0, 1, \ldots$

(1) **Cluster initial snapshot.** Build sketch G' by sampling N' nodes from $G(0)$ uniformly at random. Invoke **StaticCluster**(G') to obtain community estimates $\mathcal{C}'$ for the sketch. Use (10) to infer the community memberships $\mathcal{C}(0)$ of all nodes in $G(0)$ based on community estimates $\mathcal{C}'$.
Complexity: By executing **StaticCluster** solely on the sketch, we reduce the running time of this expensive step to only $O(N'^3 + N'|V(0)|)$. The first term corresponds to clustering of the sketch, and the second term corresponds to inference on the full graph.

(2) **For** each snapshot $G(t), t = 1, 2, \ldots$ **do**

(3)   **Update sketch.** Update the sketch to include n' nodes sampled uniformly at random from each community.

(4)   **Birth detection.** Identify newborn communities by calculating $\hat{V}_{\text{birth}}$ as in (17). Since there may be more than one community born at the same time, we cluster the graph induced by $\hat{V}_{\text{birth}}$ using **StaticCluster**. To keep running time low, we use the same sketch-based approach as in Step 1.
Complexity: The clustering of the nodes in $\hat{V}_{\text{birth}}$ incurs the dominant cost. We use a sketch consisting of n' nodes from $\hat{V}_{\text{birth}}$, and so the clustering will take time $O(q(t)^3 n'^3 + q(t) n' |V(t)|)$

(5)   **Infer community membership of new and moved nodes.** Use the estimator (10) to infer community membership of each node $i \in V(t) \setminus \hat{V}_{\text{birth}}$. Note that this set includes existing nodes, which may have changed community membership, as well as new nodes $V^+(t)$ which are joining existing communities.
Complexity: Calculation of the similarity metric $s_{i,u}(t)$ for a single community u and single node i takes time $O(n')$. Therefore, this step is $O(q(t)\, n' |V(t)|)$ in total.

(6)   **Split detection.** For each community u, build graph G' induced by $C'_u(t)$. From this induced graph, build $\mathbf{B}'$ as defined in (12). Calculate the eigenvalues λ_1, λ_2 of $\mathbf{B}'$. If $\lambda_2 > \sqrt{\lambda_1}$, then a split event is declared. In this case, invoke **StaticCluster**(G') to identify the emerging communities in the sketch, and then use (10) to identify nodes in the full graph belonging to these emerging communities.
Complexity: In the worst case, for each sketch community we must perform an eigendecomposition, estimate the partitions, and infer community membership in the full graph. Thus, this step is $O(q(t) n' (n'^2 + |V(t)|))$ in total.

(7)   **Merge detection.** For each pair of communities u, u', consider the communities merged if

$$\widehat{p}_{\text{in}} - \widehat{p}_{u,u'} < d \sqrt{\frac{2(\widehat{p}_{\text{in}} + \widehat{p}_{u,u'})}{|C_u| + |C_{u'}|}}, \tag{18}$$

where we use estimates of the intracommunity edge density $\widehat{p}_{\text{in}}$, and the intercommunity edge density $\widehat{p}_{u,u'}$. Condition (18) is similar to (1), except with an additional scaling parameter d in the right-hand side. If $d = 1$, then a shrinking density gap $\widehat{p}_{\text{in}} - \widehat{p}_{u,u'}$ causes erratic behavior during node inference, resulting in nodes incorrectly being moved between the pair of merging communities. This in turn corrupts the estimates $\widehat{p}_{\text{in}}, \widehat{p}_{u,u'}$. We set $d = 2$ to trigger the merge earlier and avoid this issue.
Complexity: Constructing the estimates takes $O(n'^2)$ time, whereas checking the merge condition for all pairs takes $O(q(t)^2)$ time.

(8)   **Build estimate** $\mathcal{C}(t)$ using results of Steps 4–7.

Output: Partitions $\mathcal{C}(t), t = 0, 1, \ldots$

Suppose that $\overline{q}$ and $\overline{N}$ are the maximum number of communities and nodes, respectively, in any given snapshot. We furthermore assume that N' is at most $\overline{q}n'$. Then, the computational complexity for estimating a single partition $\mathcal{C}(t)$ at time $t \geqslant 0$ is $O(\overline{q}n'(\overline{q}^2 n'^2 + \overline{N}))$. For the first iteration, this is the time required for executing step 1, whereas for each subsequent iteration, this is the total time required to execute steps (3)–(8). Contrast this with clustering the full snapshot graph, which is $O(\overline{N}^3)$ for each iteration. If $\overline{q} \ll \overline{N}$ and we use a small sketch, then this results in an orderwise improvement in complexity.

VII. ANALYSIS

In this section, we provide performance guarantees for the proposed algorithm, as well as guidelines for setting sketch size. To simplify analysis we take the sketch to be balanced at all time steps, i.e., $|C'_u(t)| = n'$ for each community u and time t. Furthermore, we suppose that the graph at the previous time step has been correctly clustered, i.e., $\mathcal{C}(t-1) = \widehat{\mathcal{C}}(t-1)$. Unless otherwise specified, it is assumed that $p_{u,u'}(t) = p_{\text{out}}$ for any two communities u and u'. The average degree of such a snapshot with $\widehat{q}$ communities is $c = n'(p_{\text{in}} + \widehat{q}\, p_{\text{out}})$. The following approximation is made to provide clearer results.

Assumption 1. Each of $s_{i,u}(t)$, $\widehat{p}_{\text{in}}$, and $\widehat{p}_{u,u'}$ is well approximated by a normal random variable having the same mean and variance.

This assumption follows from the fact that the listed variables are driven by binomial random variables. The underlying distributions of these random variables will generally have enough symmetry to be well approximated by normal distributions [24]. More details are provided in Appendix C.

We now provide definitions used in this section. Denote by $\Phi^{-1}(\cdot)$ the inverse cumulative distribution function of the standard normal distribution. Specifically, given a standard normal random variable Z and probability α, we have $\mathbb{P}(Z \leqslant \Phi^{-1}(\alpha)) = \alpha$. Consider a graph with N nodes and $\widehat{q}$ equal-sized communities $\widehat{\mathcal{C}} = \{\widehat{C}_v \mid v \in \{1, \ldots, \widehat{q}\}\}$. The agreement with an estimated community partition $\mathcal{C} = \{C_v \mid v \in \{1, \ldots, \widehat{q}\}\}$ is defined as [25]

$$A(\widehat{\mathcal{C}}, \mathcal{C}) = \frac{1}{N} \max_{\pi} \sum_{v=1}^{\widehat{q}} \left|\widehat{C}_v \cap C_{\pi(v)}\right|, \tag{19}$$

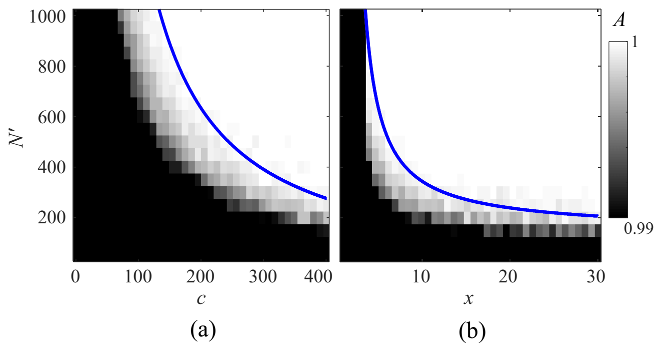

FIG. 4. Plot of agreement for community estimates produced by **StaticCluster**(G'), where sketch graph G' is produced by randomly sampling N' nodes from the full graph. The edge densities are determined from average degree c and ratio $x = p_{\text{in}}/p_{\text{out}}$. Plots are shown for (a) varying c with $x = 5$ and (b) varying x with $c = 200$.

where π ranges over the permutations on $\widehat{q}$ elements (this permutation is necessary since the community indices may be ordered arbitrarily). Exact recovery is solved by an algorithm if it produces community estimates such that $\mathbb{P}(A(\widehat{\mathcal{C}}, \mathcal{C}) = 1) \to 1$ as $N \to \infty$. In this section, we use 20 trials for each experiment. Detailed derivations for the results in this section are deferred to Appendix C.

A. Estimating communities in the initial sketch

This section provides guidelines for choosing initial sketch size. For simplicity, we consider the symmetric case in which every community has n nodes. Suppose that an initial sketch has been constructed by sampling N' nodes from $G(0)$. If $a = \frac{p_{\text{in}}N'}{\ln(N')}$ and $b = \frac{p_{\text{out}}N'}{\ln(N')}$ are held constant, then exact recovery of the planted partition is efficiently solvable in the initial sketch provided $(\sqrt{a}-\sqrt{b})^2 > \widehat{q}(0)$ [25]. Although this bound is only exact in the limit, it can still be used to estimate values of N' for which agreement will remain high. Specifically, for fixed N', p_{in}, p_{out}, this bound becomes

$$\frac{N'}{\ln(N')} > \widehat{q}(0)(\sqrt{p_{\text{in}}} - \sqrt{p_{\text{out}}})^{-2}. \tag{20}$$

Either a small density gap $p_{\text{in}} - p_{\text{out}}$ or a large number of communities $\widehat{q}(0)$ can make the initial estimate unreliable. These issues can be mitigated by increasing the sketch size.

We demonstrate the efficacy of (20) in Fig. 4. We produce a sketch from a graph with two communities of size $n = 2500$, and plot agreement between the estimated and planted communities. The blue line indicates the boundary of (20). Indeed, the agreement remains high (exceeding 0.998) whenever this condition holds.

We note that if the initial snapshot is imbalanced, i.e., with communities of different size, the sampling inversely proportional to node degree (SPIN) sampling method [5] may be used in place of uniform random sampling. This method can improve the success rate by sampling more uniformly across communities.

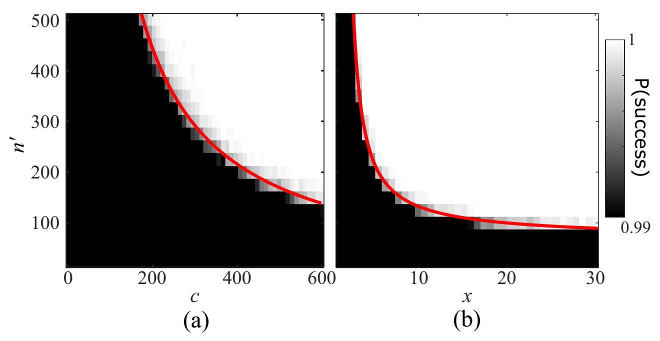

FIG. 5. Empirical estimate of the probability of successfully detecting a node joining a new community. The edge densities are determined from average degree c and ratio $x = p_{\text{in}}/p_{\text{out}}$. Plots are shown for (a) varying average degree c with $x = 5$ and (b) varying x with $c = 200$.

B. Birth detection

Suppose that one or more new communities are born at time t, and take a node $i \in V_{\text{birth}}$ which belongs to one of these communities. Let $\sigma_{\widehat{p}_{\text{in}}} = \sqrt{\frac{2p_{\text{in}}(1-p_{\text{in}})}{\widehat{q}(t)n'(n'-1)}}$ be the standard deviation of estimator $\widehat{p}_{\text{in}}$. Then the probability that node i is correctly identified as belonging to V_{birth} is at least α if

$$n' \geqslant \left(\frac{\Phi^{-1}(1-\beta)\sqrt{p_{\text{out}}(1-p_{\text{out}})} + 3\sqrt{p^{+}(1-p^{-})}}{p^{-} - p_{\text{out}}}\right)^2, \tag{21}$$

where $\beta = (1-\alpha)/[2(\widehat{q}(t)-1)]$ and

$$p^{\pm} = p_{\text{in}} \pm \Phi^{-1}\left(1 - \frac{\beta}{2}\right)\sigma_{\widehat{p}_{\text{in}}}. \tag{22}$$

Note that the sufficient number of samples in (21) is independent of community size in the full graph. This allows for the detection of new communities even when they are of a very small size. This advantage is illustrated further in the numerical results of Sec. VIII B.

Figure 5 shows results in which a new community with 500 nodes joins a graph containing two existing communities of size $n = 2500$. The plots indicate the fraction of nodes in V_{birth} which are correctly identified as belonging to the newborn community. The red line shows the boundary of condition (21) with $\alpha = 0.99$, and shows excellent agreement with the numerical results.

As the density gap $p_{\text{in}} - p_{\text{out}}$ shrinks, a larger sketch will be required to reliably detect which nodes belong to newborn communities. In fact, as $n' \to \infty$, we have $\sigma_{\widehat{p}_{\text{in}}} \to 0$ such that $p^{\pm} \to \widehat{p}_{\text{in}}$, and the right-hand side of (21) converges to

$$\left(\frac{\Phi^{-1}(1-\beta)\sqrt{p_{\text{out}}(1-p_{\text{out}})} + 3\sqrt{p_{\text{in}}(1-p_{\text{in}})}}{p_{\text{in}} - p_{\text{out}}}\right)^2. \tag{23}$$

In this regime, the denominator depends solely on the square of the density gap.

C. Inferring community membership

We next consider the required sketch size to successfully infer community membership of individual nodes using (10).

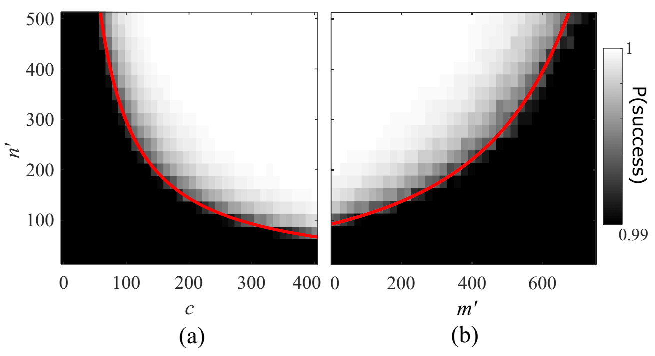

FIG. 6. Plots show empirical estimate of success probability of (10) over 10^5 trials. Sketch graph G' is produced by randomly sampling N' nodes from a network with the same parameters as in Fig. 4. Let $m'_{1\to 2} = m'_{2\to 1} = m'$, i.e., m' nodes move from one community to the other at each time step, and $p_{\text{in}} = 5p_{\text{out}}$. (a) Varying average degree c with $m' = 0$. (b) Varying m' with $c = 300$.

Define $m'_{u\to u'} = |V_{u\to u'}(t) \cap \mathcal{S}(t-1)|$, i.e., the number of nodes in the sketch that are moving from u to u'. The analysis here will use the following simplification.

Assumption 2. In place of random variable $m'_{u\to u'}$, we use its expected value $\mathbb{E}[m'_{u\to u'}] = n'|V_{u\to u'}|/|\widehat{C}_u(t-1)|$.

We denote the minimum community size in a given snapshot by $n_{\min} = \min_{1\leqslant v\leqslant \widehat{q}(t)} |\widehat{C}_v(t)|$.

Suppose that at most $\overline{m}$ nodes move between any two pairs of communities, i.e., $|V_{v\to w}(t)| \leqslant \overline{m}$ for $1 \leqslant v, w \leqslant \widehat{q}(t)$. Then (10) correctly identifies the community of a given node $i \notin \mathcal{S}(t)$ with probability $\geqslant \alpha$ provided that

$$n' \geqslant \overline{x}\left[\frac{\Phi^{-1}\left(1 - \frac{1-\alpha}{\widehat{q}(t)-1}\right)}{\overline{\mu}}\right]^2, \tag{24}$$

where

$$\overline{\mu} = (p_{\text{in}} - p_{\text{out}})\left(1 - \frac{\overline{m}\,\widehat{q}(t)}{n_{\min}}\right), \tag{25}$$

$$\overline{x} = p_{\text{in}}(1-p_{\text{in}}) + p_{\text{out}}(1-p_{\text{out}}) \tag{26}$$

$$+\overline{m}\frac{p_{\text{in}}(1-p_{\text{in}}) - p_{\text{out}}(1-p_{\text{out}})}{n_{\min}}. \tag{27}$$

Variable $\overline{\mu}$ serves as a lower bound on the expected value of $s_{i,u}(t) - s_{i,v}(t)$ for $v \neq u$, whereas $\frac{\overline{x}}{n'}$ serves as an upper bound on the standard deviation. Both a small density gap $p_{\text{in}} - p_{\text{out}}$ and a large number of moving nodes $\overline{m}$ can make inference less reliable. In these situations, an increased sketch size will be required to keep the probability of misclassification low.

Figure 6 shows the inference success rate of (10) for a network containing two communities of size $n = 2500$. In Fig. 6(a), $\overline{m} = 0$ (no nodes move between communities), whereas Fig. 6(b) varies the number of moving nodes. The red lines indicate the boundary of (24) with $\alpha = 0.99$, and show excellent agreement with the numerical results. As $\overline{m} \to 0$, this boundary converges to

$$\left[\Phi^{-1}\left(1 - \frac{1-\alpha}{\hat{q}(t)-1}\right)\right]^2\left(\frac{p_{\text{in}}(1-p_{\text{in}}) + p_{\text{out}}(1-p_{\text{out}})}{(p_{\text{in}} - p_{\text{out}})^2}\right), \tag{28}$$

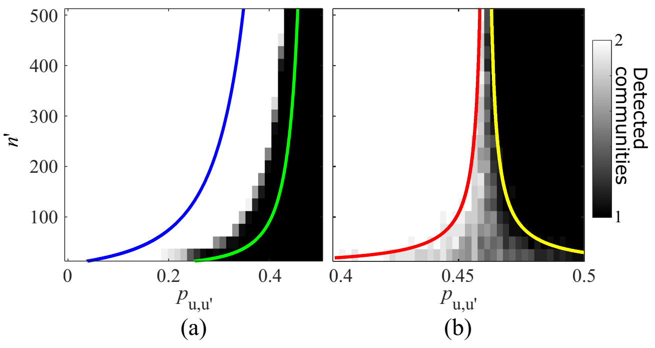

FIG. 7. Detected number of communities for a pair of communities undergoing the merge-split process. Detection is based on (a) split condition (14) and (b) merge condition (18).

which is independent of community size in the full graph. This advantage will be illustrated further in the numerical results of Sec. VIII B. In this case, the primary driver of performance becomes the density gap $p_{\text{in}} - p_{\text{out}}$.

D. Split detection

Consider a network with a single community undergoing a split into two equal-sized communities u and u'. An important consideration is the smallest value of $p_{u,u'}$ at which the communities will be considered split according to (14). Using a similar argument as for the initial sketch, in practice we may use the exact recovery limit to approximate this lower bound. Likewise, we can use the asymptotic detectability threshold as an approximate upper bound. Following this line of reasoning, it is likely that the split will be detected for some $p_{u,u'}$ bounded according to

$$p_{\text{in}} + \frac{1}{2n'} - \sqrt{\frac{2p_{\text{in}}}{n'} + \frac{1}{4n'^2}} > p_{u,u'} > \left(\sqrt{p_{\text{in}}} - \sqrt{\frac{\ln(2n')}{n'}}\right)^2. \tag{29}$$

Increased sketch size will tend to allow earlier detection of the split, i.e., for smaller values of $p_{\text{in}} - p_{u,u'}$.

Figure 7(a) shows a plot of the estimated number of communities from (14) for a sketch with two communities containing n' nodes each (the detected number of communities is 2 if the condition holds, and 1 otherwise). Along the horizontal axis, we vary $p_{u,u'}$ within $[0, p_{\text{in}}]$, where $p_{\text{in}} = 0.5$. The blue and green lines show the lower and upper bounds in (29), respectively. The split is indeed detected for a value of $p_{u,u'}$ within these bounds.

E. Merge detection

Finally, suppose that two equal-sized communities u, u' are merging into one community. We consider the value of $p_{u,u'}$ at which condition (18) detects a merge. This condition will hold with probability $\geqslant \alpha$ if

$$p_{u,u'} \geqslant p_{\text{in}} + \Phi^{-1}\left(1 - \frac{1-\alpha}{2}\right)\sigma_m + \frac{d^2}{2n} - d\sqrt{\frac{2p_{\text{in}}}{n} + \frac{d^2}{n^2}}, \tag{30}$$

where we use the the standard deviation of $\widehat{p}_{\text{in}} \pm \widehat{p}_{u,u'}$,

$$\sigma_m = \sqrt{\frac{2p_{\text{in}}(1-p_{\text{in}})}{\widehat{q}(t)n'(n'-1)} + \frac{p_{u,u'}(1-p_{u,u'})}{(n')^2}}. \tag{31}$$

However, it is also important to consider when (18) reliably identifies the communities as being split. This occurs with probability $\geqslant \alpha$ if

$$p_{u,u'} \leqslant p_{\text{in}} - \Phi^{-1}\left(1 - \frac{1-\alpha}{2}\right)\sigma_m + \frac{d^2}{2n} - d\sqrt{\frac{2p_{\text{in}}}{n} + \frac{d^2}{n^2}}. \tag{32}$$

To illustrate the significance of bounds (30) and (32), Fig. 7(b) shows the detected number of communities for a pair of communities with $n = 2500$ nodes each [the detected number of communities is 1 if (18) holds, and 2 otherwise]. The red line indicates the boundary of (30), and the yellow line indicates the boundary of (32), for $\alpha = 0.9$. When $p_{u,u'}$ falls in the gap between these bounds, the detection tends to be unreliable. However, the size of this gap can be reduced by using larger sketch sizes to drive down the standard deviation σ_m.

VIII. NUMERICAL RESULTS

We compare against four algorithms from the literature, each of which uses a different means for carrying forward information from one snapshot to the next. First, we use the classic Bayesian approach found in Yang *et al.* [17]. Second, we run the algorithm of Dinh *et al.* [14]. This algorithm uses a sketchlike concept by consolidating known communities into "supernodes" within a weighted graph. These supernodes are then incorporated into the next snapshot. Third, we use evolutionary clustering based on structural perturbation and resource allocation similarity (ESPRA), which is based on structural perturbation theory [26]. This algorithm defines an objective function which explicitly balances two similarities: one which encourages temporal smoothness across snapshots and one that takes into account only the community structure in the latest snapshot. Last, we independently cluster each snapshot as described in such works as Ref. [13,27]. This algorithm, referred to here as (Independent), estimates the communities in the current snapshot using **StaticCluster** and then matches the estimates in adjacent snapshots using the Jaccard similarity coefficient [28]. Although **StaticCluster** performs optimally in certain regimes, the main weakness of (Independent) is that it completely ignores information from the previous snapshot when clustering the current snapshot.

Further details regarding these algorithms are provided in Appendix D. Unless otherwise specified, all plots show an average over 20 independent runs. We set the initial sketch size to $N' = \widehat{q}(0)n'$.

A. Performance with small clusters

We first consider the performance of the proposed algorithm in the presence of small communities. We use normalized agreement to compare the planted communities $\widehat{\mathcal{C}} = \{\widehat{C}_v \mid v \in \{1,\dots,\widehat{q}\}\}$ and estimated communities $\mathcal{C} = \{C_v \mid v \in \{1,\dots,\widehat{q}\}\}$. Sets $\widehat{\mathcal{C}}$ and $\mathcal{C}$ are padded with empty

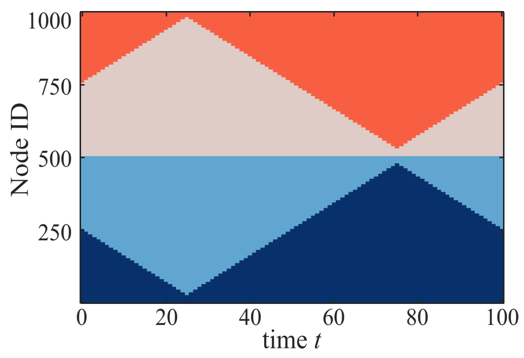

FIG. 8. Planted partitions for a double-stacked version of the grow-shrink benchmark, with $n = 250$, $f = 0.95$, $\widehat{q} = 4$, $p_{\text{in}} = 0.4$, and $p_{\text{out}} = 0.1$.

communities such that $|\widehat{\mathcal{C}}| = |\mathcal{C}|$. Then, normalized agreement is defined as [25]

$$\tilde{A} = \frac{1}{\widehat{q}} \max_{\pi} \sum_{\substack{v=1 \\ |\widehat{C}_v|>0}}^{\widehat{q}} \frac{|\widehat{C}_v \cap C_{\pi(v)}|}{|\widehat{C}_v|}, \tag{33}$$

where π ranges over the permutations on $\widehat{q}$ elements. The normalized agreement for the snapshot at time t is denoted $\tilde{A}(t)$. Unlike the agreement metric defined earlier in (21), normalized agreement proves useful for quantifying performance in the presence of small clusters, since each community constitutes a fraction $\frac{1}{\widehat{q}(t)}$ of the normalized agreement, regardless of community size.

For summarizing the overall deviation in the actual and estimate communities for a snapshot sequence, we use the average-squared error

$$E_{\tilde{A}} = \frac{1}{T}\sum_{t=1}^{T}[1 - \tilde{A}(t)]^2, \tag{34}$$

where T is the total number of snapshots.

1. *Grow-shrink benchmark*

We use two concurrent instances of the grow-shrink benchmark with phase $\phi = 0$ for the first instance and $\phi = \tau/2$ for the second instance. Figure 8 shows planted partitions for an example with $f = 0.95$. The community detection results are shown for all algorithms in Fig. 9(a), where the value of $E_{\tilde{A}}$ is plotted as a function of f. The proposed algorithm has $E_{\tilde{A}} < 0.02$ for all values of f, whereas the other algorithms exhibit significantly larger values of $E_{\tilde{A}}$ especially for larger f.

To gain further insight into the behavior of the algorithms, we plot a heat map of $\tilde{A}(t)$ for each algorithm in Figs. 9(b)–9(f), with f varied along the vertical axis and time t along the horizontal. Increasing values of f result in smaller communities at times $t = \tau/4$ and $t = 3\tau/4$ when the graph is most imbalanced. It is exactly around these times that the algorithms tend to perform worst. (Independent) often loses track of the small clusters at $t = \tau/4$ and $t = 3\tau/4$, resulting in a merge of communities and a sharp drop in agreement.

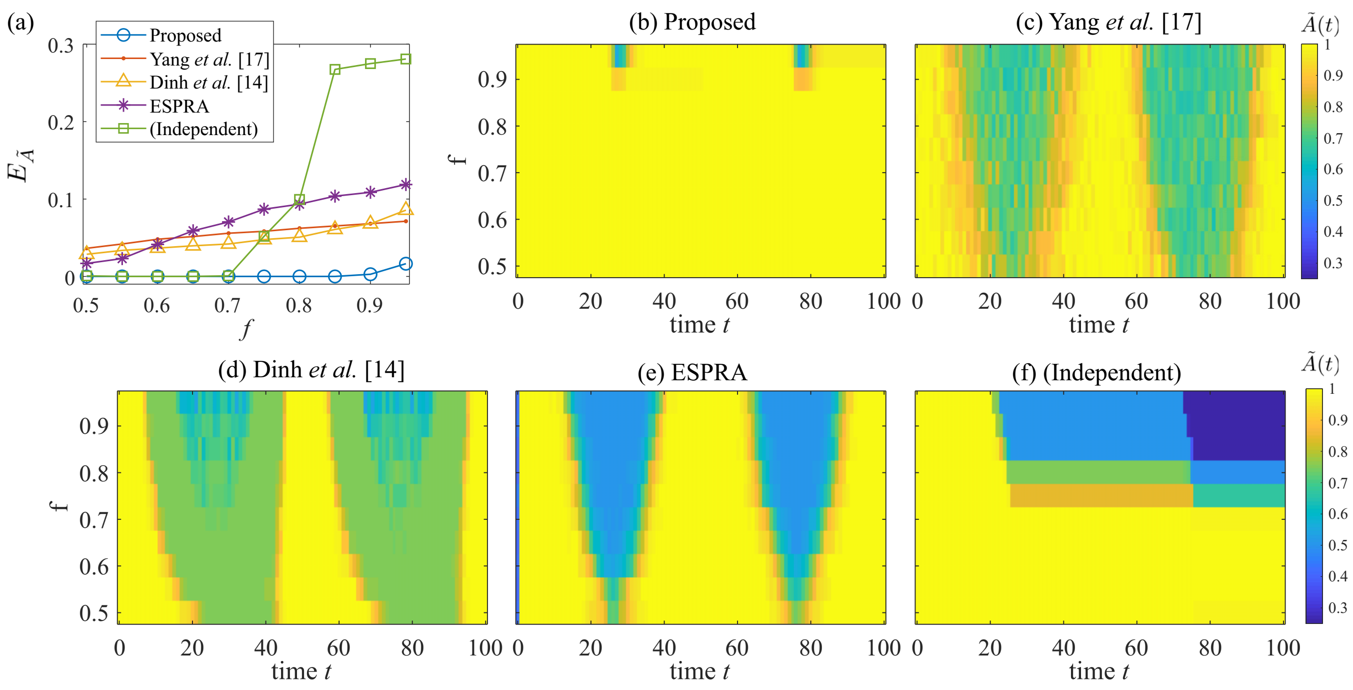

FIG. 9. Results for varying f in the grow-shrink example in Fig. 8. For the proposed algorithm we set $n' = 50$. Plot of $E_{\tilde{A}}$ is shown in (a). Panels (b) through (f) show ensemble averages of $\tilde{A}(t)$ as a function of time along the horizontal axis and f along the vertical axis for each algorithm.

The algorithm is not capable of detecting splits, and so does not recover.

2. Birth-death benchmark

We now present analogous examples for the birth-death benchmark. One means for producing small clusters is by using small values of γ, such that each community is small immediately after birth and before death. An example is shown in Fig. 10(a), with $\gamma = 0.1$. We execute the algorithms and show the estimated number of communities for each snapshot in Fig. 10(b). The algorithm of Ref. [14] tends to absorb small communities into the larger communities, as exhibited by the drop in estimated number of communities after birth and before death. Meanwhile, the proposed algorithm provides a near-perfect estimate. We expand on this example by plotting $E_{\tilde{A}}$ as a function of γ in Fig. 10(c). The proposed algorithm has $E_{\tilde{A}} < 0.003$ for all values of γ. We omit (Independent), Yang *et al.* [17], and ESPRA as they cannot handle graphs of changing size, nor new communities.

B. Scalability

To demonstrate the scalability of the proposed algorithm, let us consider the minimum community size over all snapshots

$$\overline{n}_{\min} = \min_{1\leqslant t\leqslant T} \min_{1\leqslant v\leqslant \widehat{q}(t)} \left|\widehat{C}_v(t)\right|. \tag{35}$$

We run the grow-shrink benchmark using the same parameters as in Sec. VIII A 1, except with $f = 1 - \frac{n}{\overline{n}_{\min}}$ such that the minimum cluster sizes are fixed at $\overline{n}_{\min} = 200$. Table I shows the value of $E_{\tilde{A}}$ as a function of n, averaged over five trials. There is a small increase in $E_{\tilde{A}}$ as n increases, due to a corresponding increase in the number of moving nodes (as described in Sec. VII C). Nonetheless, $E_{\tilde{A}}$ remains below 5.5×10^{-5} despite a dramatic increase in imbalance of the full graph and despite the fact that the sketch size remains fixed.

Likewise, we run the birth-death benchmark with the parameters of Sec. VIII A 2 but with $\gamma = 2\,\overline{n}_{\min}/n$ such that $\overline{n}_{\min} = 20$ regardless of graph size. The smallest community size is attained immediately before death and after birth. The results are shown in Table I. Unlike the results for the grow-shrink benchmark, there is no increase in $E_{\tilde{A}}$. This is consistent with the analysis in Sec. VII B, which showed no dependence on community size in the full graph.

For both benchmarks, Table I shows only a sublinear increase in runtime as community size n increases, owing to the fixed sketch size. To expand on this result, we run all of the algorithms on the grow-shrink benchmark, and show the results as a function of n in Fig. 11. As expected, the proposed algorithm finishes very fast, in under two seconds for all cases. On the other hand, algorithms [17], ESPRA, and (Independent) all cluster the full graph, and therefore scale superlinearly with network size. Although Ref. [14] clusters a graph of reduced size at each time step, nodes having changed edges are left as singleton nodes. In this example, the large number of edge changes forces a correspondingly large number of nodes to remain singletons, thus requiring the static clustering step to operate on large networks.

C. Merge-split detection

We next execute the algorithms on the merge-split benchmark, with two concurrent instances as shown in Fig. 12(a). The plot in Fig. 12(b) shows the estimated number of communities as a function of time for the proposed algorithm, as

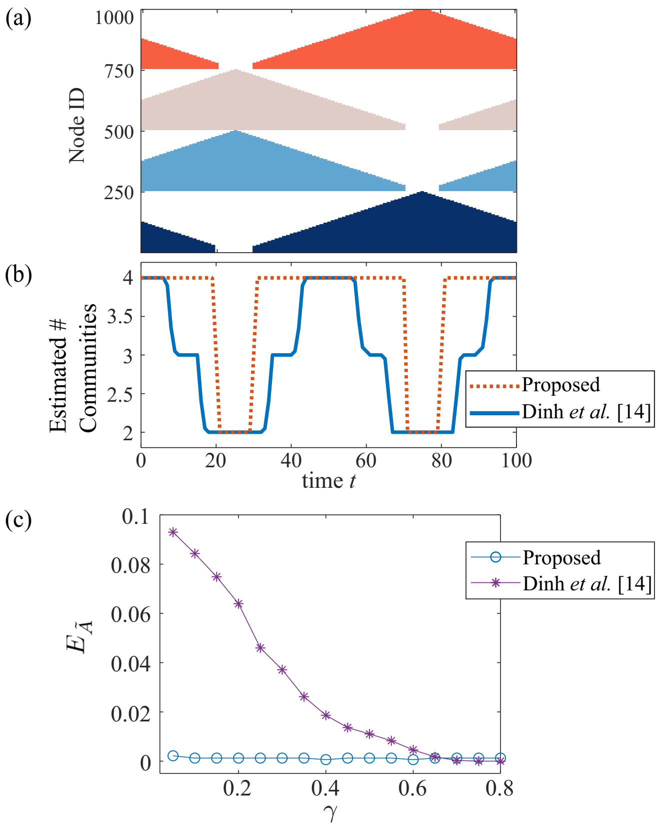

FIG. 10. (a) Planted partitions for a double-stacked version of the birth-death benchmark, with $\gamma = 0.1$. For the first instance, we use phase shift $\phi = 0$, whereas for the second instance we use $\phi = \tau/2$. Both instances have parameters $\widehat{q} = 4$, $p_{\text{in}} = 0.5$, and $p_{\text{out}} = 0.05$. We set $n' = 50$, which leads to a maximum sketch size of 200 nodes. (b) Ensemble average of number of communities estimated by algorithm plot as a function of time. (c) Squared error of normalized agreement $E_{\tilde{A}}$ is shown for varying γ.

well as for Ref. [14] and ESPRA. We omit (Independent) and Ref. [17] as they cannot handle the merge and split processes. Although the benchmark groundtruth undergoes an instantaneous transition between merged and split states, the network itself gradually interpolates between these states. This discrepancy in timescales, along with the fact that the benchmark sets the transition at the theoretical detectability limit, means we cannot expect the estimated partitions to exactly match the planted partitions. Indeed, all three algorithms overestimate the span of time during which the communities are merged.

The estimates of the proposed algorithm are shown in Fig. 12(c), where we can see that nodes start being misclassified at $t = 13$. This is expected due to the shrinking gap between p_{in} and the intercommunity edge densities, as described in Sec. VI. Nonetheless, the proposed algorithm detects the merge much closer to the benchmark's merge time than the other two algorithms. We note that using larger values of d in condition (18) will result in an earlier detection of the merge. In this way, d can act as a tuning parameter to adjust the sensitivity of the merge detection.

For studying the performance of the algorithms' split detection, we show the exact recovery limit for the sketch as a vertical white dashed line in Fig. 12(a). The proposed

TABLE I. Scalability of proposed algorithm.

Benchmark	n	$E_{\tilde{A}}$	Runtime (normalized)
Grow-shrink ($\overline{n}_{\min} = 200$)	250	3.0×10^{-7}	1
	1000	7.2×10^{-7}	1.2
	2000	4.6×10^{-6}	2.2
	3000	5.5×10^{-5}	3.7
Birth-death ($\overline{n}_{\min} = 20$)	250	1.6×10^{-5}	1
	1000	2.4×10^{-5}	1.2
	2000	2.6×10^{-7}	1.6
	3000	2.6×10^{-7}	1.7

algorithm detects the split close to this limit, although we point out that the detection could be shifted earlier by increasing the sketch size. Despite clustering the full network, for which estimation should be easier, ESPRA does not exceed the performance of the proposed algorithm, and Ref. [14] fares even worst.

D. Mixed benchmark

So far, our results have considered individual benchmarks in isolation. We now run the proposed algorithm on the mixed benchmark from Fig. 2(a), which has concurrent birth-death, grow-shrink and merge-split processes. The partition estimates are shown in Fig. 13(a). Most of the mismatch occurs in the merge-split communities, which is consistent with our earlier results.

The sketches produced by the proposed algorithm are shown in Fig. 13(b). The sketch nodes are sorted vertically according to their planted communities, with the color indicating the estimated community of the corresponding node. The deviation from the ideal sketch in Fig. 2(b) lies mostly

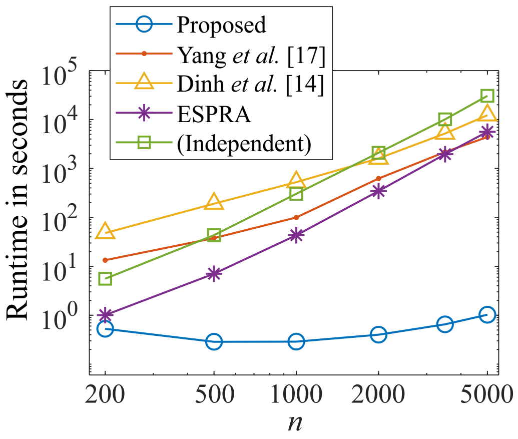

FIG. 11. Timing results for the grow-shrink benchmark with $\widehat{q} = 2$, $p_{\text{in}} = 0.5$, $p_{\text{out}} = 0.05$, $f = 0.5$. For the proposed algorithm we set $n' = 100$, leading to a total sketch size of 200 nodes. Due to the large runtimes in the four algorithms we compare against, only 10 iterations of the algorithms are performed, and time is averaged over five trials. Note that logarithmic scales are used for both axes. All algorithms had perfect community estimates for all network sizes, except for ESPRA which still had less than 1% misclassified nodes per snapshot.

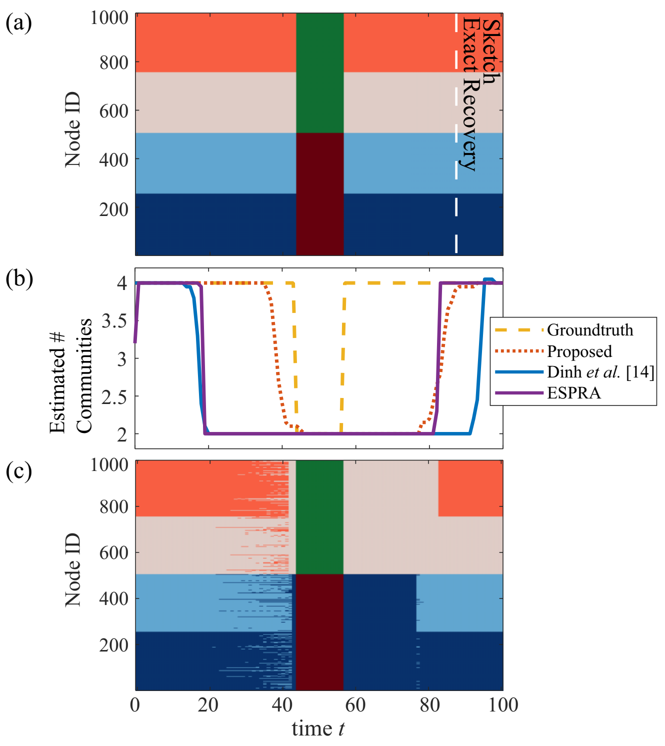

FIG. 12. (a) Planted partitions for a double-stacked version of the merge-split benchmark. The parameters of the model are $\widehat{q} = 4$, $n = 250$, $p_{\text{in}} = 0.5$, and $p_{\text{out}} = 0.05$. We set $n' = 50$ for the proposed algorithm. This results in a sketch size of 100 in the merged state, and 200 in the split state. The exact recovery limit for the sketch, based on (21), is shown as a dashed white vertical line. (b) The estimated number of communities at each time step for the proposed algorithms, as well for Ref. [14] and ESPRA. (c) The estimated partitions for the proposed algorithm.

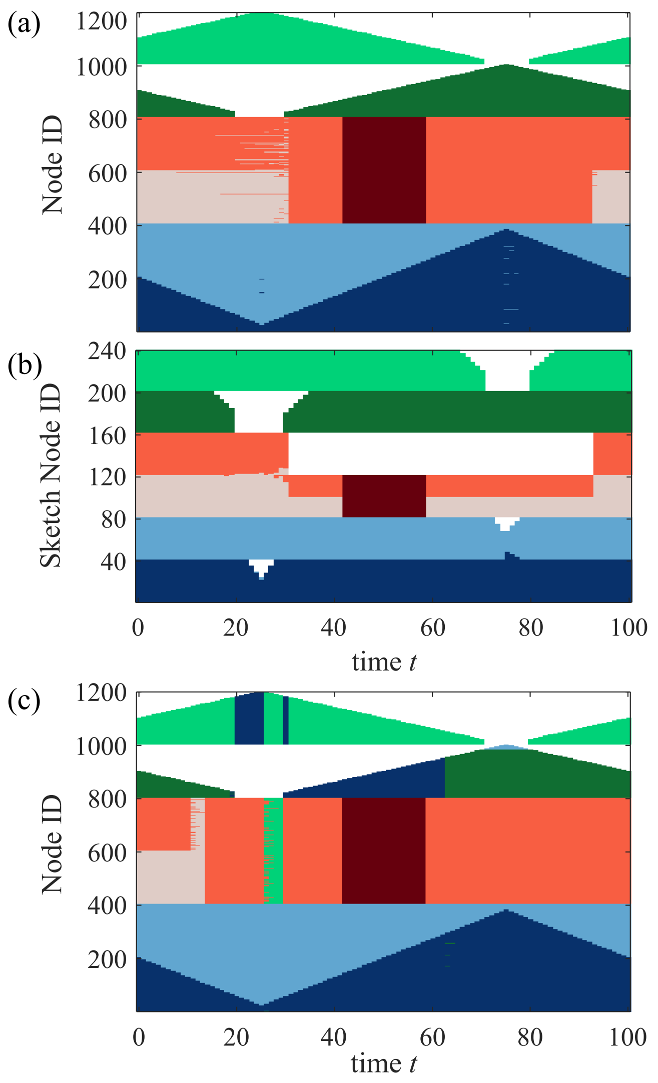

FIG. 13. Results for mixed benchmark with $p_{\text{in}} = 0.5$ and $p_{\text{out}} = 0.05$. Estimated partitions produced by the proposed algorithm are shown for (a) the full network and (b) the sketches produced by the proposed algorithm. The estimated partitions produced by Ref. [14] are shown in (c).

inside the merge-split communities, due to the errors present in the estimates of the full graph.

The estimated partitions for Ref. [14] are presented in Fig. 13(c). As with the earlier results, Ref. [14] encounters difficulties in correctly identifying the small clusters in the grow-shrink and birth-death communities.

IX. CONCLUSION

This paper concerned a sketch-based approach for community detection in time-evolving networks. We presented an SBM-based model along with possible evolutionary processes which may occur within this model. We then proposed sketch-based techniques for tracking these processes, as well as an algorithm incorporating these techniques to produce community estimates for concurrent processes. We provided an analysis to guide the choice of sketch size, and generated numerical results comparing the proposed algorithm to full-scale community detection algorithms.

We conclude by briefly noting possible extensions. First, an arbitrary community detection algorithm may be used in place of **StaticCluster**, provided that it can estimate the number of communities. Second, a straightforward extension to the network model and algorithm would allow the intracommunity edge density p_{in} to vary for each community. Third, our approach is extendable to other graph models as well, for example the Degree Corrected SBM (DCSBM) [29]. This can be accomplished by substituting a suitable sampling technique for constructing DCSBM sketches (e.g., the sampling method of Ref. [30]), a similarity definition between nodes in the full network and the sketch communities, and an appropriate technique for determining when clusters split or merge.

ACKNOWLEDGMENTS

This work was supported by NSF CAREER Award No. CCF-1552497 and NSF Award No. CCF-2106339. The University of Central Florida Advanced Research Computing Center provided computational resources that contributed to results reported herein.

APPENDIX A: STATIC CLUSTERING

We use the following algorithm to perform static clustering of graph G with N nodes.

```
StaticCluster($G$)
(1) Construct $\mathbf{B}'$ from $G$ using (12).
(2) Calculate eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_N$ and corresponding
    eigenvectors of $\mathbf{B}'$.
(3) Calculate $q$ as the maximum value of $i$ such that $\lambda_i > \sqrt{\lambda_1}$.
(4) for $i = 1 \dots q$ do
(5)     Build matrix $\mathbf{M}$ from the $i$ normalized eigenvectors of $\mathbf{B}'$
        corresponding to eigenvalues $\lambda_1, \dots, \lambda_i$. Apply k-means
        clustering to $\mathbf{M}$ to obtain community estimates
        $\mathcal{C}_i = \{C_v \mid v \in \{1, \dots, i\}\}$ We repeat 100 iterations with
        three random initializations and take the best result.
(6)     Calculate modularity $Q_i$ of $G$ with partition $\mathcal{C}_i$. Modularity
        is defined as in Sec. IV of Ref. [31].
(7) end for
(8) $j \leftarrow \arg\max_{1 \leqslant i \leqslant q} Q_i$
(9) Return estimate $\mathcal{C}_j$.
```

Steps (1)–(3) estimate the number of communities q, and are as described in Ref. [23]. We find that adding Steps (4)–(7) provides a more reliable estimate of the number of communities. These steps repeat k-means clustering, varying the number of clusters up to q, and then return the partition giving the highest modularity.

Figure 14 compares the proposed function **StaticCluster** (solid lines), against the standard approach (dashed lines). The standard approach only uses k-means to identify q communities, rather than executing Steps (4)–(7). The plot shows the fraction of runs in which the estimated number of communities is correct of 20 runs. As can be seen, the proposed algorithm identifies the correct number of communities for much smaller values of average degree c.

The additional steps do not increase the complexity of the algorithm. In particular, the time for Steps (4)–(7) is $O(N\widehat{q}^3)$, where $\widehat{q}$ is the actual number of communities. We assume that $\widehat{q} \ll n'$, such that the eigendecomposition is still the dominant cost, making the computational complexity of the algorithm $O(N^3)$.

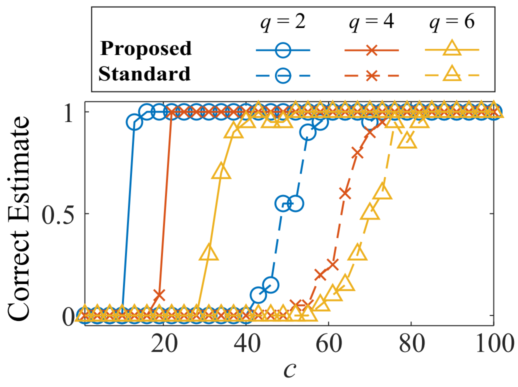

FIG. 14. Results showing improved performance of the proposed static clustering algorithm afforded by the modularity-based heuristic in Steps (4)–(7). We generate a graph G with 800 nodes divided into q equal-sized communities, and $p_{\text{in}} = 5p_{\text{out}}$.

APPENDIX B: MAIN ALGORITHM: DETAILS

Two helper functions are needed. The first, **Sample**(G, N'), returns a set of N' nodes sampled uniformly at random from G. The second infers community membership of nodes in the full graph G based on the sketch community estimates $\mathcal{C}'$, as described in Sec. V A. This function is defined as follows:

```
Infer($G, \mathcal{C}'$)
(1) $C_v \leftarrow \emptyset$ for $v \in \{1, \dots, q\}$
(2) for each node $i \in V$ do
(3)     $w \leftarrow \arg\max_{v \in \{1,\dots,q\}} \frac{|\{(i,j) \in E \mid j \in C'_v\}|}{|C'_v|}$
(4)     $C_w \leftarrow C_w \cup \{i\}$
(5) end for
(6) return partition $\mathcal{C} = \{C_v \mid v = 1, \dots, q\}$
```

We use the following definition: given a graph $G = (V, E)$ and node set $V' \subset V$, the subgraph of G induced by V' is denoted $G[V']$. The complete definition of **MainAlgorithm** follows.

```
(1) $G \leftarrow G(0)$
(2) $\mathcal{S}' \leftarrow$ Sample($G, N'$)
(3) $\mathcal{C}(0) \leftarrow$ Infer($G$, StaticCluster($G[\mathcal{S}']$))
(4) Build sketch set $\mathcal{S}(0)$ by sampling $n'$ nodes uniformly at
    random from each community $C \in \mathcal{C}(0)$. If $n' > |C|$, then
    include all nodes from $C$.
(5) $r \leftarrow |\mathcal{C}(0)|$.
(6) for $t = 1, 2, \dots$ do
(7)     $G \leftarrow G(t)$
(8)     $\{C_1, \dots, C_r\}$
            $\leftarrow$ Infer($G[V(t) \setminus \hat{V}_{\text{birth}}], \mathcal{C}'(t-1)$)
(9)     if $\hat{V}_{\text{birth}} \neq \emptyset$ then
(10)        $\overline{G} \leftarrow G[\hat{V}_{\text{birth}}]$
(11)        $\overline{\mathcal{S}} \leftarrow$ Sample($\overline{G}, n'$)
(12)        $\{C_{r+1}, \dots, C_{r+q}\}$
                $\leftarrow$ Infer ($\overline{G}$, StaticCluster ($\overline{G}[\overline{\mathcal{S}}]$))
(13)        $r \leftarrow r + q$
(14)    end if
(15)    for $u \in \{1, \dots, r\}$, where $|C_u| > a$ do
(16)        $C' \leftarrow C_u \cap \mathcal{S}(t-1)$
(17)        $G' \leftarrow G[C']$
(18)        Let $\mathbf{A}$ be the adjacency matrix of $G'$. Calculate the
            eigenvalues $\lambda_1, \lambda_2, \dots$ of $\mathbf{B}'$ [defined in (12)].
(19)        Calculate $q$ as the maximum value of $i$ such that
            $\lambda_i > \sqrt{\lambda_1}$.
(20)        if $q > 1$ then
(21)            $G_u \leftarrow G[C_u]$
(22)            $\mathcal{C}'' \leftarrow$ StaticCluster($G_u[C']$)
(23)            $\{C_u, C_{r+1}, \dots, C_{r+q-1}\}$
                    $\leftarrow$ Infer($G_u, \mathcal{C}''$)
(24)            $r \leftarrow r + q - 1$
(25)        end if
(26)    end for
```

```
(27)    for community pairs $u, u' \in \{1, \dots, r\}$ do
(28)      if (18) holds then
(29)        $C_u \leftarrow C_u \cup C_{u'}$
(30)        $C_{u'} \leftarrow \emptyset$
(31)      end if
(32)    end for
(33)    $\mathcal{C}(t) \leftarrow \{C_v \mid v = 1, \dots, r\}$
(34)    $V^- \leftarrow V(t-1) \setminus V(t)$
(35)    $\mathcal{S}(t) \leftarrow \mathcal{S}(t-1) \setminus V^-$
(36)    Re-proportion sketch $\mathcal{S}(t)$ such that it contains
        $\min\{n', |C|\}$ nodes from each community $C \in \mathcal{C}(t)$.
(37) end for
```

Steps (1)–(3) cluster the first graph snapshot. This estimate is used to construct a balanced sketch in step (4). The remainder of the algorithm processes subsequent snapshots. Step (8) reevaluates the community membership of existing nodes, as well as new nodes joining existing communities. Steps (9)–(14) partition the set of newborn communities. Meanwhile, Steps (15)–(26) handle splits within each community. Only communities with size greater than a are checked, as the spectral estimates become unreliable for small communities. We set $a = 20$. Steps (27)–(32) handle merges among pairs of communities. Finally, steps (34)–(36) generate the new sketch.

APPENDIX C: DERIVATIONS FOR ANALYSIS IN SEC. VII

In this section, we denote a binomial random variable having n trials with probability of success p by $\mathrm{Bin}(n, p)$. All of the binomial random variables found in this section indicate the number of edges between nodes and/or communities. Unless otherwise indicated, we assume that the number of edges is sufficient such that $np \geqslant 10$, and that the network is sparse enough such that $n(1-p) \geqslant 10$. This justifies the use of Assumption 1 [24]. We denote a normal random variable with mean μ and variance σ^2 by $\mathcal{N}(\mu, \sigma^2)$.

1. Initial sketch

We comment here on the validity of the exact recovery analysis. The SBM model used for analyzing exact recovery in Ref. [25] does not have fixed community sizes, but rather defines a probability distribution over communities. The membership of each node is then sampled from this distribution. In fact, the initial sketch $G'(0)$ adheres to this model, since the probability that a given node in the sketch belongs to community u is $\frac{|\widehat{C}_u(0)|}{N'}$.

2. Inferring community membership

Let $m'_{(u\to\cdot)} = \sum_{w\neq u} m'_{u\to w}$ be the total number of sketch nodes moving out of community u, and $m'_{(\cdot\to u)} = \sum_{w\neq u} m'_{w\to u}$ be the total number of sketch nodes moving into community u. Then,

$$s_{i,v}(t) = \frac{S^v_{\text{out}} + S^v_{\text{in}}}{n'}, \tag{C1}$$

where

$$S^v_{\text{in}} \sim \begin{cases}\mathrm{Bin}(n' - m'_{(u\to\cdot)}, p_{\text{in}}), & v = u\\ \mathrm{Bin}(m'_{v\to u}, p_{\text{in}}), & v \neq u\end{cases}, \tag{C2a}$$

$$S^v_{\text{out}} \sim \begin{cases}\mathrm{Bin}(m'_{(u\to\cdot)}, p_{\text{out}}), & v = u\\ \mathrm{Bin}(n' - m'_{v\to u}, p_{\text{out}}), & v \neq u\end{cases}. \tag{C2b}$$

Since the random variables in (C2) are mutually independent, it follows from Assumption 1 that $s_{i,u}(t) - s_{i,v}(t) \sim \mathcal{N}(\mu_{\Delta s}, \sigma^2_{\Delta s})$ for any $v \neq u$, where

$$\mu_{\Delta s} = \frac{(p_{\text{in}} - p_{\text{out}})(n' - m'_{(u\to\cdot)} - m'_{v\to u})}{n'},$$
$$\sigma^2_{\Delta s} = \frac{p_{\text{in}}(1-p_{\text{in}}) + p_{\text{out}}(1-p_{\text{out}})}{n'} - \frac{[p_{\text{in}}(1-p_{\text{in}}) - p_{\text{out}}(1-p_{\text{out}})](m'_{(u\to\cdot)} - m'_{v\to u})}{n'^2}. \tag{C3}$$

We note that if there are few moving nodes or few edges between communities, then S^v_{in} and S^v_{out} may not be well approximated by a normal random variable. Nonetheless, in these cases the expected values of S^v_{in} and S^v_{out} are small enough that they do not contribute significantly to the bound regardless.

Let $f \sim \mathcal{N}(\overline{\mu}, \frac{\overline{x}}{n'})$. If (24) holds, then $\overline{\mu} \geqslant \Phi^{-1}(1 - \frac{1-\alpha}{\widehat{q}(t)-1})\sqrt{\frac{\overline{x}}{n'}}$ and hence $\mathbb{P}(f < 0) \leqslant \frac{1-\alpha}{\widehat{q}(t)-1}$. Furthermore, from Assumption 2, we have $\mu_{\Delta s} \geqslant \overline{\mu}$ and $\sigma^2_{\Delta s} \leqslant \frac{\overline{x}}{n'}$, and thus $\mathbb{P}(s_{i,u}(t) < s_{i,v}(t)) \leqslant \mathbb{P}(f < 0)$. Then, by applying the union bound, the probability of successfully inferring the community membership of node i is

$$\mathbb{P}\left(\bigcap_{\substack{w=1\\ w\neq u}}^{\widehat{q}(t)} s_{i,u}(t) > s_{i,w}(t)\right) = 1 - \mathbb{P}\left(\bigcup_{\substack{w=1\\ w\neq u}}^{\widehat{q}(t)} s_{i,u}(t) < s_{i,w}(t)\right)$$
$$\geqslant 1 - \sum_{\substack{w=1\\ w\neq u}}^{\widehat{q}(t)} \mathbb{P}(s_{i,u}(t) < s_{i,w}(t))$$
$$= \alpha. \tag{C4}$$

3. Split detection

Following a similar line of reasoning as in Sec. VII A, exact recovery is efficiently solvable if

$$\frac{2n'}{\ln(2n')} > 2(\sqrt{p_{\text{in}}} - \sqrt{p_{u,u'}})^{-2}. \tag{C5}$$

On the other hand, from (13), the split is asymptotically detectable in the spectrum of $\mathbf{B}'$ if

$$n'(p_{\text{in}} - p_{u,u'})^2 > p_{\text{in}} + p_{u,u'}, \tag{C6}$$

which is equivalent to

$$p^2_{u,u'} - p_{u,u'}\left(2p_{\text{in}} + \frac{1}{n'}\right) + p_{\text{in}}\left(p_{\text{in}} - \frac{p_{\text{in}}}{n'}\right) \geqslant 0. \tag{C7}$$

The bounds in (29) follow directly by solving (C5) and (C7) in terms of $p_{u,u'}$.

4. Merge detection

Define $\widehat{m} = \widehat{q}(t)n'(n'-1)/2$ as the maximum possible number of intracommunity edges at time t. Then,

$$\widehat{p}_{\rm in} = \frac{\widehat{P}_{\rm in}}{\widehat{m}}, \tag{C8a}$$

$$\widehat{p}_{u,u'} = \frac{\widehat{P}_{u,u'}}{(n')^2} \tag{C8b}$$

where $\widehat{P}_{\rm in} \sim {\rm Bin}(\widehat{m}, p_{\rm in})$, and $\widehat{P}_{u,u'} \sim {\rm Bin}((n')^2, p_{u,u'}(t))$. Since $\widehat{P}_{\rm in}$ and $\widehat{P}_{u,u'}$ are independent, it follows from Assumption 1 that $\widehat{p}_{\rm in} - \widehat{p}_{u,u'} \sim \mathcal{N}(\mu^-, \sigma_m^2)$ and $\widehat{p}_{\rm in} + \widehat{p}_{u,u'} \sim \mathcal{N}(\mu^+, \sigma_m^2)$, where $\mu^\pm = p_{\rm in} \pm p_{u,u'}$.

Let $\beta' = 1 - \frac{1-\alpha}{2}$. If condition (30) holds, then

$$p_{u,u'}^2 - p_{u,u'}\left(2p_{\rm in} + 2\Phi^{-1}(\beta')\sigma_m + \frac{d^2}{n}\right) + \left[p_{\rm in} + \Phi^{-1}(\beta')\sigma_m\right]^2 - d^2\frac{p_{\rm in} - \Phi^{-1}(\beta')\sigma_m}{n} < 0, \tag{C9}$$

which in turn implies that

$$\mu^- + \Phi^{-1}(\beta')\sigma_m < d\sqrt{\frac{\mu^+ - \Phi^{-1}(\beta')\sigma_m}{n}}. \tag{C10}$$

Then, from (C10) and the union bound, the probability that (18) will indicate a split state is

$$\begin{aligned}
&\mathbb{P}\left(\widehat{p}_{\rm in} - \widehat{p}_{u,u'} < d\sqrt{\frac{\widehat{p}_{\rm in} + \widehat{p}_{u,u'}}{n}}\right)\\
&\geqslant \mathbb{P}\Bigg(\widehat{p}_{\rm in} - \widehat{p}_{u,u'} < \mu^- + \Phi^{-1}(\beta')\sigma_m\\
&\qquad \bigcap d\sqrt{\frac{\widehat{p}_{\rm in} + \widehat{p}_{u,u'}}{n}} > d\sqrt{\frac{\mu^- - \Phi^{-1}(\beta')\sigma_m}{n}}\Bigg)\\
&\geqslant 1 - \mathbb{P}(\widehat{p}_{\rm in} - \widehat{p}_{u,u'} < \mu^- + \Phi^{-1}(\beta')\sigma_m)\\
&\quad - \mathbb{P}(\widehat{p}_{\rm in} + \widehat{p}_{u,u'} > \mu^+ - \Phi^{-1}(\beta')\sigma_m)\\
&= \alpha
\end{aligned} \tag{C11}$$

We can justify (32) using a similar argument.

5. Detecting the birth event

From (C1) and (C8a) and under Assumption 1 with no moving nodes,

$$s_{i,v}(t) \sim \mathcal{N}\left(p_{\rm out}, \frac{p_{\rm out}(1-p_{\rm out})}{n'}\right), 1 \leqslant v \leqslant \widehat{q}(t) \tag{C12}$$

$$\widehat{p}_{\rm in} \sim \mathcal{N}\left(p_{\rm in}, \frac{p_{\rm in}(1-p_{\rm in})}{\widehat{m}}\right) \tag{C13}$$

Then $\mathbb{P}(p^- \leqslant \widehat{p}_{\rm in} \leqslant p^+) = 1 - \beta$, and therefore

$$\begin{aligned}
&\mathbb{P}\left(\widehat{p}_{\rm in} - 3\sqrt{\frac{\widehat{p}_{\rm in}(1-\widehat{p}_{\rm in})}{n'}} < p^- - 3\sqrt{\frac{p^+(1-p^-)}{n'}}\right)\\
&\leqslant 1 - \mathbb{P}(p^- \leqslant \widehat{p}_{\rm in} \leqslant p^+) = \beta.
\end{aligned} \tag{C14}$$

Furthermore, for any community v,

$$\mathbb{P}\left(s_{i,v}(t) \geqslant p_{\rm out} + \Phi^{-1}(1-\beta)\sqrt{\frac{p_{\rm out}(1-p_{\rm out})}{n'}}\right) = \beta. \tag{C15}$$

Condition (21) is equivalent to

$$p_{\rm out} + \Phi^{-1}(1-\beta)\sqrt{\frac{p_{\rm out}(1-p_{\rm out})}{n'}} \leqslant p^- - 3\sqrt{\frac{p^+(1-p^-)}{n'}}. \tag{C16}$$

Then, the probability that the condition inside (17) will fail for a particular v is

$$\begin{aligned}
&\mathbb{P}(s_{i,v}(t) \geqslant \widehat{p}_{\rm in} - 3\widehat{\sigma})\\
&= \mathbb{P}\left(s_{i,v}(t) \geqslant \widehat{p}_{\rm in} - 3\sqrt{\frac{\widehat{p}_{\rm in}(1-\widehat{p}_{\rm in})}{n'}}\right)\\
&\leqslant \mathbb{P}\Bigg(s_{i,v}(t) \geqslant p_{\rm out} + \Phi^{-1}(1-\beta)\sqrt{\frac{p_{\rm out}(1-p_{\rm out})}{n'}}\\
&\qquad \bigcup \widehat{p}_{\rm in} - 3\sqrt{\frac{\widehat{p}_{\rm in}(1-\widehat{p}_{\rm in})}{n'}} < p^- - 3\sqrt{\frac{p^+(1-p^-)}{n'}}\Bigg)\\
&\leqslant \mathbb{P}\left(s_{i,v}(t) \geqslant p_{\rm out} + \Phi^{-1}(1-\beta)\sqrt{\frac{p_{\rm out}(1-p_{\rm out})}{n'}}\right)\\
&\quad + \mathbb{P}\left(\widehat{p}_{\rm in} - 3\sqrt{\frac{\widehat{p}_{\rm in}(1-\widehat{p}_{\rm in})}{n'}} < p^- - 3\sqrt{\frac{p^+(1-p^-)}{n'}}\right)\\
&= 2\beta.
\end{aligned} \tag{C17}$$

Consequently, the probability that the condition will hold for all communities is

$$\begin{aligned}
&\mathbb{P}\left(\bigcap_{\substack{v=1\\v\neq u}}^{\widehat{q}(t)} s_{i,v}(t) + 3\widehat{\sigma} < \widehat{p}_{\rm in}\right)\\
&= 1 - \mathbb{P}\left(\bigcup_{\substack{v=1\\v\neq u}}^{\widehat{q}(t)} s_{i,v}(t) + 3\widehat{\sigma} \geqslant \widehat{p}_{\rm in}\right)\\
&\geqslant 1 - \sum_{\substack{v=1\\v\neq u}}^{\widehat{q}(t)} \mathbb{P}(s_{i,v}(t) + 3\widehat{\sigma} \geqslant \widehat{p}_{\rm in})\\
&= 1 - 2[\widehat{q}(t) - 1]\beta = \alpha.
\end{aligned} \tag{C18}$$

APPENDIX D: NUMERICAL RESULTS: DETAILS OF ALGORITHMS USED IN COMPARISON

For the algorithm of Yang *et al.* [17], we use the same tuning strategy as in the experimental results section of Ref. [17]. Specifically, we use the same temperature and iteration sequences, with $\alpha = 0.8$, $\beta = 0.5$, $\gamma = 1$, $\mu_{kk} = 10$. We run five instances of the algorithm with (1) $\alpha_{kk} = 1$, $\beta_{kl} = 1$; (2) $\alpha_{kk} = 5$, $\beta_{kl} = 1$; (3) $\alpha_{kk} = 10$, $\beta_{kl} = 1$; (4) $\alpha_{kk} = 10^2$, $\beta_{kl} = 10$; and (5) $\alpha_{kk} = 10^4$, $\beta_{kl} = 10$, and then take the community assignments among the five trials yielding the highest average modularity (modularity is defined as in Ref. [17]). For the algorithm of Dinh *et al.* [14], we use the CNM algorithm [7] for the static clustering step, as in Ref. [14]. When running the ESPRA algorithm, we use the same parameters as used in the experimental results of Ref. [26]: $\alpha = 0.8$, $\beta = 0.5$. The algorithm (Independent) applies **StaticCluster** to each snapshot to obtain community estimates $\{\overline{C}_1, \dots, \overline{C}_q\}$. To provide continuity in the community assignments of the nodes, community u in each snapshot at time $t > 0$ is matched to the community at time $t-1$ having the largest overlap according to the Jaccard coefficient. Specifically, for each community u, we set the new estimate as $C_u(t) = \overline{C}_{u'}$ where $u' = \arg\max_{1 \leqslant v \leqslant q} \frac{|\overline{C}_u \cap C_v(t-1)|}{|\overline{C}_u \cup C_v(t-1)|}$.

[1] C. Aggarwal and K. Subbian, Evolutionary network analysis: A survey, ACM Comput. Surv. **47**, 1 (2014).

[2] D. Greene, D. Doyle, and P. Cunningham, Tracking the evolution of communities in dynamic social networks, in *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining, Odense, Denmark* (IEEE, Piscataway, NJ, 2010), pp. 176–183.

[3] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine, Synopses for massive data: Samples, histograms, wavelets, sketches, Found. Trends Databases **4**, 1 (2011).

[4] M. Rahmani, A. Beckus, A. Karimian, and G. K. Atia, Scalable and robust community detection with randomized sketching, IEEE Trans. Signal Process. **68**, 962 (2020).

[5] A. Beckus and G. K. Atia, Scalable community detection in the heterogeneous stochastic block model, in *Proceedings of the IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP'19), Pittsburgh, PA, USA* (IEEE, Piscataway, NJ, 2019), pp. 1–6.

[6] J. Shang, L. Liu, X. Li, F. Xie, and C. Wu, Targeted revision: A learning-based approach for incremental community detection in dynamic networks, Physica A **443**, 70 (2016).

[7] A. Clauset, M. E. J. Newman, and C. Moore, Finding community structure in very large networks, Phys. Rev. E **70**, 066111 (2004).

[8] N. Dakiche, F. B.-S. Tayeb, Y. Slimani, and K. Benatchba, Tracking community evolution in social networks: A survey, Inf. Process. Manage. **56**, 1084 (2019).

[9] S. Zhang and H. Zhao, Community identification in networks with unbalanced structure, Phys. Rev. E **85**, 066114 (2012).

[10] C. Granell, R. K. Darst, A. Arenas, S. Fortunato, and S. Gómez, Benchmark model to assess community structure in evolving networks, Phys. Rev. E **92**, 012805 (2015).

[11] P. W. Holland, K. B. Laskey, and S. Leinhardt, Stochastic blockmodels: First steps, Soc. Networks **5**, 109 (1983).

[12] G. Rossetti and R. Cazabet, Community discovery in dynamic networks: A survey, ACM Comput. Surv. **51**, 1 (2018).

[13] J. Hopcroft, O. Khan, B. Kulis, and B. Selman, Tracking evolving communities in large linked networks, Proc. Natl. Acad. Sci. USA **101**, 5249 (2004).

[14] T. N. Dinh, Y. Xuan, and M. T. Thai, Towards social-aware routing in dynamic communication networks, in *Proceedings of the IEEE International Performance Computing and Communications Conference (IPCCC'09), Scottsdale, AZ, USA* (IEEE, Piscataway, NJ, 2009), pp. 161–168.

[15] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. (2008) P10008.

[16] J. He and D. Chen, A fast algorithm for community detection in temporal network, Physica A **429**, 87 (2015).

[17] T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin, Detecting communities and their evolutions in dynamic social networks—A Bayesian approach, Mach. Learn. **82**, 157 (2011).

[18] A. Ghasemian, P. Zhang, A. Clauset, C. Moore, and L. Peel, Detectability Thresholds and Optimal Algorithms for Community Structure in Dynamic Networks, Phys. Rev. X **6**, 031005 (2016).

[19] K. S. Xu and A. O. Hero, Dynamic stochastic blockmodels for time-evolving social networks, IEEE J. Sel. Top. Sign. Process. **8**, 552 (2014).

[20] M. Pensky and T. Zhang, Spectral clustering in the dynamic stochastic block model, Electron. J. Statist. **13**, 678 (2019).

[21] P. Jiao, T. Li, H. Wu, C.-D. Wang, D. He, and W. Wang, HB-DSBM: Modeling the dynamic complex networks from community level to node level, IEEE Trans. Neural Netw. Learn. Syst., 1 (2022).

[22] C. Matias and V. Miele, Statistical clustering of temporal networks through a dynamic stochastic block model, J. R. Stat. Soc. B **79**, 1119 (2017).

[23] F. Krzakala, C. Moore, E. Mossel, J. Neeman, A. Sly, L. Zdeborová, and P. Zhang, Spectral redemption in clustering sparse networks, Proc. Natl. Acad. Sci. USA **110**, 20935 (2013).

[24] J. Devore, *Modern Mathematical Statistics with Applications* (Springer, New York, 2012).

[25] E. Abbe, Community detection and stochastic block models: Recent developments, J. Mach. Learn. Res. **18**, 1 (2018).

[26] P. Wang, L. Gao, and X. Ma, Dynamic community detection based on network structural perturbation and topological similarity, J. Stat. Mech. **2017**, 013401 (2017).

[27] T. Aynaud, E. Fleury, J.-L. Guillaume, and Q. Wang, *Communities in Evolving Networks: Definitions, Detection, and Analysis Techniques* (Springer, Berlin, 2013), pp. 159–200.

[28] P. Jaccard, The distribution of the flora in the alpine zone. 1, New Phytol. **11**, 37 (1912).

[29] B. Karrer and M. E. J. Newman, Stochastic blockmodels and community structure in networks, Phys. Rev. E **83**, 016107 (2011).

[30] Y. He, A. Beckus, and G. K. Atia, Scalable community detection in the degree-corrected stochastic block model, in *Proceedings of the IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP'21), Gold Coast, Australia* (IEEE, Piscataway, NJ, 2021), pp. 1–6.

[31] M. E. J. Newman and M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E **69**, 026113 (2004).