2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)
Allerton Park and Retreat Center
Monticello, IL, USA, September 24-27, 2019

A Sequential and Scalable Approach to Community Detection in Dynamic Graphs

Andre Beckus and George K. Atia

Abstract— **We study a sequential sketch-based approach for the clustering of time-evolving graphs. We present a dynamic extension to the static Stochastic Block Model, which accommodates growing and shrinking graphs, as well as the movement of nodes between clusters. We then propose an online algorithm which constructs and maintains a small sketch consisting of nodes sampled from the full graph. The sketch is clustered and a retrieval algorithm is used to infer cluster membership of nodes in each successive graph snapshot. We demonstrate that the use of a small sketch not only improves computational complexity, but also improves the success rate when sketches are properly proportioned. We present a sampling method which chooses nodes according to node degree, whereby very small clusters can be successfully tracked.**

Index Terms— **Clustering, Community Detection, Sequential Methods, Dynamic Graphs**

I. Introduction

Static community detection has received a great deal of attention, with large strides being made in developing efficient algorithms. However, it is often the case that graphs change with time, thus requiring new types of algorithms for tracking evolving community structure (see [1] and references therein). The extra dimension inherent to this dynamic setting only makes the search for efficient algorithms more critical. In this paper we present sequential dynamic graph clustering algorithms which are not only scalable, but also exhibit improved accuracy.

Means of representing dynamic networks generally fall under two main categories [2], [3]. Temporal networks treat events as a continuous stream, and track the exact appearance and disappearance times of individual edges. On the other hand, snapshot networks treat the network as a series of static graphs occurring at regular time intervals. Here, we propose a framework around the latter formalism. We note that it is often possible to convert a temporal network into a snapshot network.

The evolution of a graph can be described by a set of six standard community events [3]. The birth and death events signal when communities pass in and out of existence, respectively. The merge event marks when a set of two or more clusters join into one cluster, and the split event marks when a single cluster separates into multiple clusters. In this paper, we will focus on the grow and shrink events, in which clusters gain or lose nodes, respectively. Note that these two events do not cause the number of clusters to change.

The authors are with the Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816 USA. Emails: `abeckus@knights.ucf.edu, george.atia@ucf.edu`

This work was supported by NSF CAREER Award CCF-1552497. The authors acknowledge the University of Central Florida Advanced Research Computing Center for providing computational resources that contributed to results reported herein.

II. Related Work

A. Data models and benchmarks

Our data model is based on the Stochastic Block Model (SBM) [4], [5], which has found frequent use in the static community detection literature due its ease of analysis. This probabilistic generative model plants a set of disjoint clusters in the graph. Pairs of nodes within a cluster are connected with probability p, whereas pairs with nodes in different clusters are connected with probability q.

The well-known set of dynamic SBM-based benchmarks [6] comprise three specific sequences of SBM graphs. One sequence consists of growing and shrinking communities, while another exhibits merging and splitting communities. The last sequence combines all four event types. While these benchmarks are used as a point of comparison by many papers, they do not take the form of a general model.

Several dynamic models based on the SBM have been proposed. These are typically referred to as Dynamic Stochastic Block Models (DSBMs). The generative DSBMs of [7], [8], which are very similar, specify that nodes change between a fixed set of clusters according to a stationary transition probability matrix. A very restricted planted bisection DSBM is proposed in [9], containing two clusters with equal numbers of nodes. In contrast to the previous DSBMs, the versions presented in [10], [11] allow the cluster edge probabilities to vary with time, albeit the cluster memberships of the nodes are fixed in [10].

With the exception of [11], all of the aforementioned DSBMs fix the number of nodes across all time steps. Our model allows nodes to be added and deleted from the graph, and also allows movement of nodes between clusters in an unrestricted manner.

B. Algorithms

A variety of algorithms have been proposed for dynamic community detection (see surveys [2], [3] for a comprehensive list). We will consider two broad categories of algorithms.

In the first category, “independent community detection and matching”, each snapshot is clustered independently and then reconciled with the previous snapshot to provide traceability in the cluster identities. Starting with [12], a number of variations have appeared in this category, incorporating assorted static clustering and matching techniques. The matching strategy typically includes heuristics for identifying high level cluster events such as birth and death of communities. A simple implementation of this approach, which we will use for comparison purposes in the experimental results, is given as Algorithm 1 (notations will be defined precisely in Section III-A). Here, we use the Jaccard similarity index for the matching step, with the simplifying assumption that the only relevant events are movement, addition, and deletion of nodes.

Algorithm 1 Standard Independent Clustering

```
Input: Stream of graph snapshots $G^t = (V^t, E^t)$
Output: Cluster estimates $\widehat{\mathcal{C}}^t = \left\{\widehat{C}_1^t, ..., \widehat{C}_{\hat{r}}^t\right\}$, where $\hat{r}$ is the estimated number of clusters.
  ▷ 1. Initialization
  Cluster initial graph $G^1$ using any community detection algorithm to get estimated clusters $\left\{\widehat{C}_1, ..., \widehat{C}_{\hat{r}}\right\}$.
  $\widehat{\mathcal{C}}^1 \leftarrow \left\{\widehat{C}_1, ..., \widehat{C}_{\hat{r}}\right\}$
  ▷ 2. Sequential clustering
  for each new snapshot $G^t$ do
      Cluster graph $G^t$ using any community detection algorithm to get cluster estimates $\left\{\widehat{C}_1, ..., \widehat{C}_{\hat{r}}\right\}$.
      for $i = 1$ to $\hat{r}$ do
          $j \leftarrow \arg\max_k \frac{|\widehat{C}_i \cap \widehat{C}_k^{t-1}|}{|\widehat{C}_i \cup \widehat{C}_k^{t-1}|}$
          $\widehat{C}_i^* \leftarrow \widehat{C}_j$
      end for
      $\widehat{\mathcal{C}}^t \leftarrow \left\{\widehat{C}_1^*, ..., \widehat{C}_{\hat{r}}^*\right\}$
  end for
```

The second algorithm category, "dependent community detection", uses the previous snapshot to aid in subsequent clustering steps. These algorithms have the potential to outperform independent community detection algorithms, since they incorporate previous knowledge *directly* in the clustering step.

Bayesian inference algorithms, such as those found in [7], are a natural choice when the data fits the DSBM. Under this framework, the previous clustering can be used as a prior for the next step. These algorithms heavily depend on the SBM structure of the data.

Another common approach found in this category is to represent the graph in a compact form. In [13], referred to here as (Dinh, 2009), a small weighted graph is built at the end of each time step with each cluster expressed as a single "supernode". The weights of edges between supernodes indicate the number of edges between the corresponding clusters, and self loops indicate the number of edges within the cluster. At the next time step, nodes likely to have changed cluster membership are extracted from the supernodes as singleton nodes, and the compact graph is reclustered. In this manner, the full graph must only be clustered at the first step, and subsequent clusterings only occur on the smaller compact graph. A similar idea can be seen in dynamic methods built around the Louvain algorithm, e.g. [14]. Because the static Louvain algorithm already depends on the concept of supernodes, the extension to dynamic graphs is a natural one.

In our approach, we also make use of small graphs, however, using an altogether different idea of sketching, as described next.

C. Sketch-based clustering

At the core of our work, we use sketches, i.e., small subgraphs consisting of nodes sampled from the full graph. To build the sketch, we use two sampling techniques presented in [15], both of which are performed without replacement.

The first sampling technique is Uniform Random Sampling (URS), in which each node of the full graph has the same probability of being sampled. However, with this approach, the expected number of samples from each cluster is proportional to the cluster sizes of the full graph. If small clusters are present in the full graph, these will tend to be small in the sketch as well, making them extremely difficult to track.

To avoid this problem, Sparsity-based Sampling (SbS) samples inversely proportional to the degree of a node. Specifically, the probability of sampling node j is set as B/d_j, where d_j is the degree of node j and $B = \left(\sum_{k=1}^{N} \frac{1}{d_k}\right)^{-1}$ is a normalization constant. As explained in [15], with this technique, the sizes of the clusters in the sketch will tend to be more balanced even in the presence of strong size imbalance in the full graph. This in turn will tend to improve the chance of clustering success in the sketch.

Algorithm 2 Static Sketch-based Clustering [15]

```
Input: Graph $G = (V, E)$
Output: Cluster estimates $\left\{\widehat{C}_1, ..., \widehat{C}_{\hat{r}}\right\}$, where $\hat{r}$ is the estimated number of clusters.
  ▷ 1. Random node sampling
  Form node set $\overline{V} \subset V$ consisting of $N'$ nodes sampled using either URS or SbS from the full set of nodes. The sampling is without replacement. Let $\overline{G}$ be the sub-graph of $G$ induced by nodes $\overline{V}$.
  ▷ 2. Sketch graph clustering
  Cluster $\overline{G}$ using any community detection algorithm to get cluster estimates $\{\overline{C}_1, ..., \overline{C}_{\hat{r}}\}$.
  ▷ 3. Full data clustering
  for $i = 1$ to $\hat{r}$ do
      $\widehat{C}_i \leftarrow \overline{C}_i$
  end for
  for each node $v \in V \setminus \overline{V}$ do
      $i \leftarrow \arg\max_j \frac{|\{(v,w) \in E \mid w \in \overline{C}_j\}|}{|\overline{C}_j|}$
      $\widehat{C}_i \leftarrow \widehat{C}_i \cup \{v\}$
  end for
```

In addition to the sampling techniques, we will also use the static sketch-based clustering algorithm from [15], which is shown as Algorithm 2. This algorithm first applies one of the randomized sampling techniques described earlier. Next, the sketch is clustered, and then the clusters of the remaining nodes in the full graph are inferred from the sketch clusters. Reducing the size of the graph input into the clustering algorithm can provide order-wise complexity savings in the expensive clustering step (see [15] for a detailed analysis).

III. Proposed Approach

A. Proposed data model

We propose the following generative DSBM model.

Data Model 1: The dynamic network consists of a series of snapshots. The snapshot graph at time step t is denoted $G^t = (V^t, E^t)$. As with SBM, if a node pair has both nodes in the same community, then an edge exists between them with probability p. On the other hand, if the two nodes in the pair belong to different communities, then an edge exists between them with probability q.

The clusters at time step t are contained in the set $\mathcal{C}^t = \{C_1^t, ..., C_r^t\}$, where r is the number of clusters, and $C_i^t \subseteq V^t$ is the set of nodes in cluster i at time t. Set $\mathcal{C}^t$ forms a partition over V^t.

At each time step $t > 1$, the following events are permitted:

1) **Addition of new nodes** A set of new nodes V_+^t is added to the graph. Each new node joins one of the existing clusters.
2) **Deletion of nodes** A set of nodes $V_-^t \subseteq V^{t-1}$ is removed from the graph.
3) **Movement of nodes between clusters** A set of nodes $\Delta V^t \subseteq V^{t-1}$ are reassigned to new clusters. The edges attached to these nodes are regenerated according to the edge probabilities corresponding to the new cluster.
4) **Regeneration of edges** Select uniformly at random m node pairs from $(V^{t-1})^2$ (the set of all possible node pairs at time t–1). Regenerate edges between these pairs, according to the new cluster memberships at time t.

The cluster membership of new and reassigned nodes are indicated by the sets $\Delta\mathcal{C}^t = \{\Delta C_1^t, ..., \Delta C_r^t\}$.

B. Proposed algorithms

Because Algorithm 1 performs community detection on the full graph at each time step, it can be slow with large networks. We improve on Algorithm 1 by replacing the clustering algorithm (which is often the bottleneck in terms of computational complexity) with the sketch-based Algorithm 2. The resulting algorithm is Algorithm 3.

Algorithm 3 Sketch-based Independent Clustering

```
Input: Stream of graph snapshots $G^t = (V^t, E^t)$
Output: Cluster estimates $\widehat{\mathcal{C}}^t = \{\widehat{C}_1^t, ..., \widehat{C}_{\hat{r}}^t\}$
  ▷ 1. Initialization
  Cluster initial graph $G^1$ using Algorithm 2 to get estimated clusters $\{\widehat{C}_1, ..., \widehat{C}_{\hat{r}}\}$.
  $\widehat{\mathcal{C}}^1 \leftarrow \{\widehat{C}_1, ..., \widehat{C}_{\hat{r}}\}$
  ▷ 2. Sequential clustering
  for each new snapshot $G^t$ do
      Cluster graph $G^t$ using Algorithm 2 to get cluster estimates $\{\widehat{C}_1, ..., \widehat{C}_{\hat{r}}\}$.
      for $i = 1$ to $\hat{r}$ do
          $j \leftarrow \arg\max_k \frac{|\widehat{C}_i \cap \widehat{C}_k^{t-1}|}{|\widehat{C}_i \cup \widehat{C}_k^{t-1}|}$
          $\widehat{C}_i^* \leftarrow \widehat{C}_j$
      end for
      $\widehat{\mathcal{C}}^t \leftarrow \{\widehat{C}_1^*, ..., \widehat{C}_{\hat{r}}^*\}$
  end for
```

Algorithm 3 falls under the category of independent graph clustering and matching. However, the sketching approach can also be used to build a dependent community detection algorithm, as shown in Algorithm 4. This algorithm differs from Algorithm 3 in the following way. In Algorithm 4, the static sketch-based clustering algorithm is only called for the first snapshot. For subsequent snapshots, this same sketch is used, and only Step 3 "Full data clustering" of the static clustering algorithm is repeated.

Algorithm 4 Efficient Single-Sketch Clustering

```
Input: Stream of graph snapshots $G^t = (V^t, E^t)$
Output: Cluster estimates $\widehat{\mathcal{C}}^t = \{\widehat{C}_1^t, ..., \widehat{C}_{\hat{r}}^t\}$
  ▷ 1. Initialization
  Cluster initial graph $G^1$ using any community detection algorithm to get estimated clusters $\{\widehat{C}_1, ..., \widehat{C}_{\hat{r}}\}$ and sketch nodes $\overline{V}$.
  $\widehat{\mathcal{C}}^1 \leftarrow \{\widehat{C}_1, ..., \widehat{C}_{\hat{r}}\}$
  $\overline{V}^1 \leftarrow \overline{V}$
  ▷ 2. Sequential clustering
  for each new snapshot $G^t$ do
      $\overline{V}^t \leftarrow \overline{V}^{t-1} \setminus V_-^t$
      for $i = 1$ to $\hat{r}$ do
          $\widehat{C}_i \leftarrow \emptyset$
          $\overline{C}_i \leftarrow \widehat{C}_i^{t-1} \cap \overline{V}^t$
      end for
      for each node $v \in \overline{V}^t$ do
          $E_j' \leftarrow \{ (v, w) \in E^t \mid w \in \overline{C}_j \}$
          $i = \arg\max_j |E_j'|/|\overline{C}_j|$
          $\widehat{C}_i \leftarrow \widehat{C}_i \cup \{v\}$
      end for
      $\widehat{\mathcal{C}}^t \leftarrow \{\widehat{C}_1, ..., \widehat{C}_{\hat{r}}\}$
  end for
```

In choosing whether to use Algorithm 3 or Algorithm 4, there is a potential trade-off. Algorithm 4 will always be faster, since it completely circumvents the sampling and clustering steps. However, if the quality of the graph data improves with time (for example the clusters become more balanced), then rebuilding and reclustering the sketch could be advantageous. Additionally, if nodes are continuously deleted from a cluster, the cluster may completely disappear from the sketch when using Algorithm 4.

IV. Numerical Experiments

We now present a numerical study of the proposed framework. For comparison purposes, we have used Spectral Clustering [16] for the clustering step of all algorithms.

A. Simultaneous events

We first show the dynamic output of our algorithms when run with a single snapshot sequence. The following settings are used: $r = 3, p = 0.15, q = p/10, |C_1^1| = 200, |C_2^1| = 400, |C_3^1| = 600, N' = 600$. At each time step, 20 nodes are chosen from each of clusters 2 and 3, and two nodes are chosen from cluster 1. These nodes are moved to cluster 1 with probability $\frac{2}{42}$, to cluster 2 with probability $\frac{20}{42}$, and to cluster 3 with probability $\frac{20}{42}$. At each time step, one node is deleted from cluster 1, two nodes are added to cluster 3, and $m = 10$ edges are regenerated.

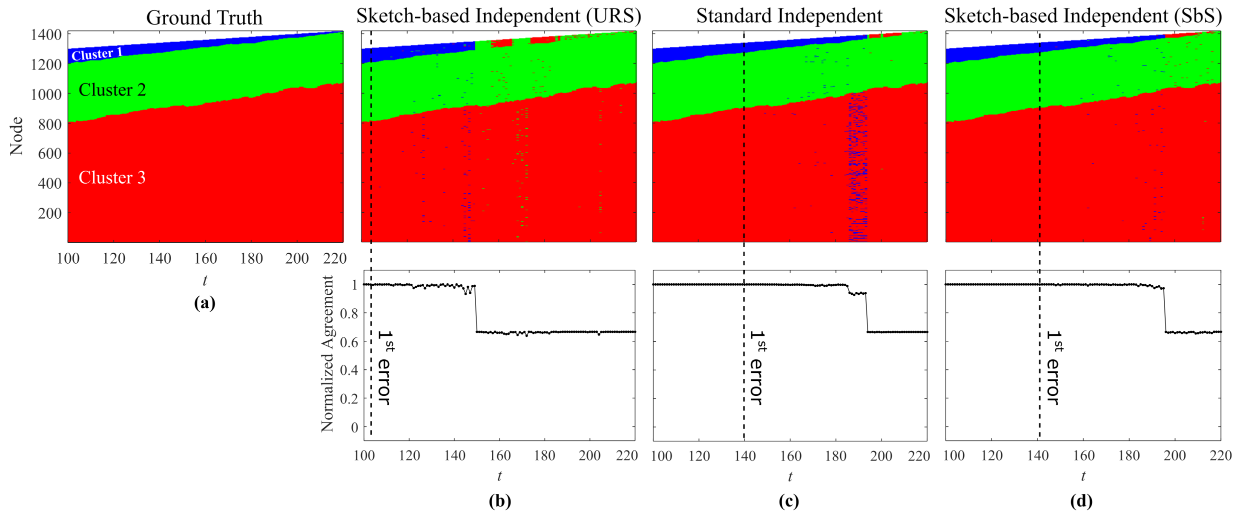

Fig. 1. Comparison of Algorithm 1 and Algorithm 3 when data model is simultaneously generating all event types. To emphasize regions of interest, the plots start at $t = 100$. Normalized agreement at time step t is defined as $\frac{1}{r} \max_{\pi \in S_r} \sum_{i=1}^{r} |C_i^t \cap \widehat{C}_i^t|/|C_i^t|$ [17].

Fig. 1(a) shows the true clusters generated by the data model. Along each vertical line, the nodes are sorted by cluster, so that the relative size of each cluster is clearly indicated for each time step. The top row of (b)-(d) show the labels returned by Algorithm 1 and Algorithm 3 (the latter being run with both URS and SbS). The bottom row shows the corresponding normalized agreement [17] for each time step. This metric normalizes the agreement for each cluster, such that clusters are weighted equally regardless of their sizes. The time of the first error is also shown, i.e., the first time when a node is misclassified. We will use this metric again in the last experiment.

Because of the small size of cluster 3, Algorithm 3 (URS) starts encountering errors early, completely losing track of cluster 1 at $t = 150$. On the other hand, SbS performs much better due to improved balance in the sketch. Although Algorithm 1 and Algorithm 3 (SbS) have similar performance in terms of normalized agreement, as shown in the next experiment, the latter algorithm is much faster.

B. Timings

We next show the improved run time of the proposed algorithms in Fig. 2. We generate data using Data Model 1, with the following parameters: $r=2, |V^t|=N, |C_1^1|=|C_2^1|=N/2, N'=500$. For the dense graph, $p=0.8, q=0.1$ and for the sparse graph, $p=0.1, q=10^{-4}$. At each time step, 20 nodes are chosen uniformly at random from each cluster, and moved to a new cluster (chosen uniformly at random). We use SbS for the proposed algorithms. These settings were chosen such that all algorithms achieve perfect estimates.

Due to the fixed sketch size, the run times of the proposed algorithms barely increase even for large graph size. Meanwhile, Algorithm 1 grows substantially slower as the graph size increases. Unlike the proposed algorithms, (Dinh, 2009) is heavily dependent on the density of the graph. When an edge changes, (Dinh, 2009) marks both end-nodes as

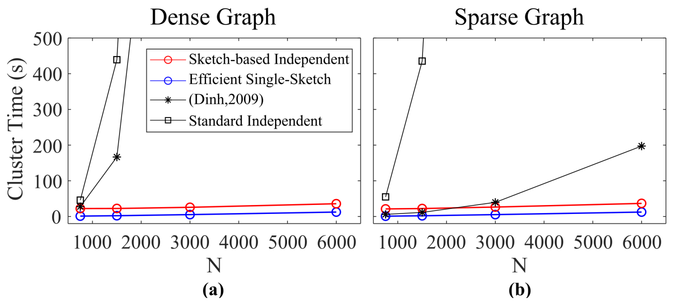

Fig. 2. Timing comparison between algorithms. For a clear comparison, we report only the time required for the clustering steps (comprising the dominant cost). For Algorithm 4, we time the inner loop of Step 2. Times are averaged over five runs, except for Standard Independent Clustering, which is run only once due to the excessive run times involved.

potentially changed, therefore triggering more nodes to be reclustered at each step when density increases.

C. Performance with small clusters

In the final experiment, we show a simple scenario where imbalance smoothly increases with time, and show when each algorithm encounters its first error.

In this experiment, we have $q = 0.02, r = 3, |C_1^1| = 4000, |C_2^1| = |C_3^1| = 500$. At each time step, one node each is chosen from each of clusters 2 and 3, and both nodes moved to cluster 1.

The actual clusters are shown in Fig. 3(a), while Fig. 3(b) shows the first error over a range of intra-cluster densities p, with fixed $N' = 600$. While the proposed algorithms have superior performance in dense graphs, for sparse graphs they underperform Algorithm 1. This is because SbS does not balance as well in these sparse regimes. For $p \geq 0.2$, Algorithm 3 performs best among all algorithms by rebuilding the sketch at each time step, thus maintaing a more balanced sketch. In Algorithm 4, however, the sketch becomes more imbalanced as nodes change cluster. For (Dinh, 2009), the supernodes

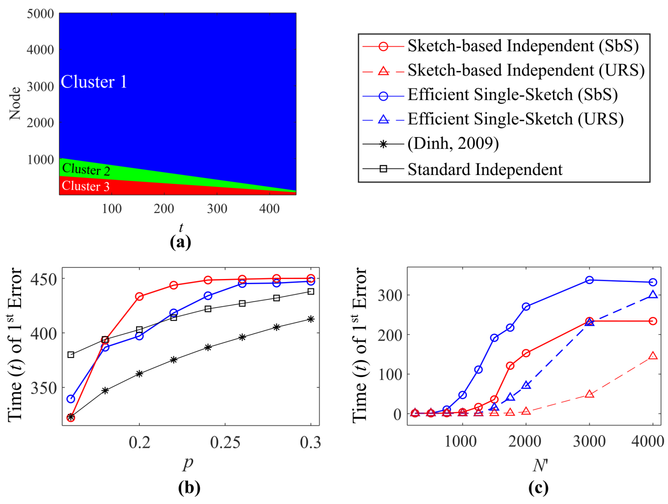

Fig. 3. Plots showing when the algorithms encounter their first errors. This scenario specifically tests the algorithms' abilities to track ever-shrinking clusters. Times are averaged over 20 runs, except for the Standard Independent Clustering, which is only run five times.

corresponding to the small clusters are very small, therefore resulting in the worst performance of all algorithms.

Fig. 3(c) shows the performance of the proposed algorithms (with URS and SbS) over a range of sketch sizes N', with fixed $p = 0.1$. In this sparser regime, the sampling step of Algorithm 3 produces small cluster sizes with high variance, thus resulting in less reliable clusterings. Algorithm 4 on the other hand builds the sketch only at the first step, when the small clusters are still large, and the chance of constructing a balanced sketch is higher. As expected, URS sampling always underperforms the corresponding SbS versions.

References

[1] C. Aggarwal and K. Subbian, "Evolutionary network analysis: A survey," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 10:1–10:36, May 2014.

[2] G. Rossetti and R. Cazabet, "Community discovery in dynamic networks: A survey," *ACM Comput. Surv.*, vol. 51, no. 2, pp. 35:1–35:37, Feb. 2018.

[3] N. Dakiche, F. B.-S. Tayeb, Y. Slimani, and K. Benatchba, "Tracking community evolution in social networks: A survey," *Inform. Process. Manag.*, vol. 56, no. 3, pp. 1084 – 1102, 2019.

[4] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Soc. Netw.*, vol. 5, no. 2, pp. 109 – 137, 1983.

[5] A. Condon and R. M. Karp, "Algorithms for graph partitioning on the planted partition model," *Random Struct. Algor.*, vol. 18, no. 2, pp. 116–140, 2001.

[6] C. Granell, R. K. Darst, A. Arenas, S. Fortunato, and S. Gómez, "Benchmark model to assess community structure in evolving networks," *Phys. Rev. E*, vol. 92, p. 012805, Jul 2015, Art. no. 012805.

[7] 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.*, vol. 82, no. 2, pp. 157–189, Feb. 2011.

[8] 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*, vol. 6, Jul 2016, Art. no. 031005.

[9] A. Clementi, M. D. Ianni, G. Gambosi, E. Natale, and R. Silvestri, "Distributed community detection in dynamic graphs," *Theor. Comput. Sci.*, vol. 584, pp. 19 – 41, 2015.

[10] Q. Han, K. S. Xu, and E. M. Airoldi, "Consistent estimation of dynamic and multi-layer block models," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 1511–1520.

[11] K. S. Xu and A. O. Hero, "Dynamic stochastic blockmodels for time-evolving social networks," *IEEE J. Sel. Topics Signal Process*, vol. 8, no. 4, pp. 552–562, Aug. 2014.

[12] J. Hopcroft, O. Khan, B. Kulis, and B. Selman, "Tracking evolving communities in large linked networks," *P. Natl. Acad. Sci.*, vol. 101, no. 1, pp. 5249–5253, 2004.

[13] T. N. Dinh, Ying Xuan, and M. T. Thai, "Towards social-aware routing in dynamic communication networks," in *IEEE IPCCC*, Dec. 2009, pp. 161–168.

[14] J. He and D. Chen, "A fast algorithm for community detection in temporal network," *Physica A*, vol. 429, pp. 87–94, 2015.

[15] M. Rahmani, A. Beckus, A. Karimian, and G. Atia, "Scalable and Robust Community Detection with Randomized Sketching," *arXiv e-print*, May 2018.

[16] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[17] E. Abbe, "Community detection and stochastic block models," *Found. Trends. Commun. Inform. Theor.*, vol. 14, no. 1-2, pp. 1–162, 2018.