Knowledge and Information Systems (2020) 62:2281–2300
https://doi.org/10.1007/s10115-019-01422-6

REGULAR PAPER

Incremental community discovery via latent network representation and probabilistic inference

Zhe Cui[1] · Noseong Park[2,4] · Tanmoy Chakraborty[3]

Received: 8 July 2018 / Revised: 31 October 2019 / Accepted: 2 November 2019 /
Published online: 15 November 2019

Abstract
Most of the community detection algorithms assume that the *complete* network structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is available in advance for analysis. However, in reality this may not be true due to several reasons, such as privacy constraints and restricted access, which result in a partial snapshot of the entire network. In addition, we may be interested in identifying the community information of only a selected subset of nodes (denoted by $\mathcal{V}_T \subseteq \mathcal{V}$), rather than obtaining the community structure of all the nodes in $\mathcal{G}$. To this end, we propose an incremental community detection method that repeats two stages—(i) network scan and (ii) community update. In the first stage, our method selects an appropriate node in such a way that the discovery of its local neighborhood structure leads to an accurate community detection in the second stage. We propose a novel criterion, called *Information Gain*, based on existing network embedding algorithms (Deepwalk and node2vec) to scan a node. The proposed community update stage consists of expectation–maximization and Markov Random Field-based denoising strategy. Experiments with 5 diverse networks with known ground-truth community structure show that our algorithm achieves 10.2% higher accuracy on average over state-of-the-art algorithms for both network scan and community update steps.

Keywords Community detection · Incremental community detection · Network embedding · Probabilistic inference

1 Introduction

In social network analysis, the task of community detection has been widely studied [1]. In many cases, instead of discovering the community structure of the *entire* network, we may wish to detect communities within a target set of nodes [2,3]. For instance, a telecommunica-

✉ Noseong Park
npark9@gmu.edu

1 Dept. of Electrical and Computer Engineering, University of Maryland, College Park, USA

2 Center for Secure Information Systems, George Mason University, Fairfax, USA

3 Dept. of Computer Science & Engineering, IIIT-Delhi, New Delhi, India

4 Department of Information Sciences and Technology, George Mason University, Fairfax, VA 22030, USA

tion company may want to find communities that its valuable customers are part of, in order to provide better facilities.

Most studies in this direction assume that the *overall* network structure is known in advance [4]. However, in real networks, complete information is difficult or even impossible to obtain [5]. In Facebook network, for instance, the complete linkage structure of a user is often unobtainable due to several privacy constraints. Moreover, in many online social network sites, such as Twitter, many new edges are created daily. In such cases, the entire network structure available to the users is incomplete.

This leads us to tackle a more realistic problem setting—*given an initial sub-network and a set of target nodes, our task is to progressively scan the network and explore the communities*[1] *where the target nodes reside*. Scanning a node means checking its social profile and retrieving its neighborhood. By adding the scanned neighbors of a node, we keep accumulating knowledge about the topology of the network. This problem setting is realistic for several reasons—first, network information obtained is usually incomplete and hard to acquire. Second, many new relationships are created everyday. Even though network information is complete at a particular time, as time goes by, the network structure may need to be scanned and updated.

However, in general the *cost to scan a network* is limited, which further creates two challenges. The first is that we have to scan the network carefully. With no constraint on scanning, we can explore the network in a brute force manner and at some point, enough nodes are scanned and correct community membership of target nodes can be revealed. However, when there is an upper limit on the cost (i.e., budget) to scan nodes, it is necessary to judiciously scan nodes in order to explore the best community information of target node set. The second challenge is to incrementally update communities of target nodes based on partial information. After a network scan, more nodes are discovered, and the algorithm needs to *quickly and correctly* update communities of target nodes.

To address the first challenge, we propose a metric, called *information gain* for selecting a new node to scan. The key idea behind this metric is to use *network embedding* to find latent representation of nodes, and compute which node (and its associated edge) have the largest information gain. This metric tracks how the latent vector representations of nodes change over a series of network updates in order to decide which node to scan that has closed edge links. If a node's latent representation drastically changes over successive updates, it is likely that its neighborhood information has also been changed, and thus the node can be a potential candidate to scan.

To address the second challenge, we propose a three-step incremental community update method—*Step 1*: an incremental local update based on expectation–maximization [6]; *Step 2*: an intermittent global update to correct the local update; and *Step 3*: the Markov Random Field (MRF)-based denoising [7] to further adjust both local and global updates.

These three steps that consist of (i) network scan, (ii) EM, and (iii) MRF are systematically combined into one framework. To maximize the community structure inference performance of the EM method, our information gain-based network scan actively searches community boundaries rather than community cores. Connections are weaker around boundaries than cores so our network scan method is designed to help the EM algorithm better infer about community structures. Because we reveal community structures of selected targets only, its number of hidden variables (community memberships) is not large when formulated in the EM method. This is one of the main reasons why we have chosen the EM method. Our

[1] In this paper, we consider disjoint communities around the target nodes.

MRF-based denoising supports the EM algorithm. Briefly, the contributions of our paper are as follows:

- We investigate a contemporary problem which is more realistic than the traditional settings of community detection and has rarely been explored in the past [8].
- We propose a novel method that interweaves network discovery and community detection by first mapping the discovered network into an embedding space, followed by an incremental community update that adjusts the current community structure by leveraging the new information acquired in network discovery.
- We compare our method with 4 different variations of the existing state-of-the-art method [8] along with 3 other commonly used baselines on 5 diverse datasets. We observe that our proposed algorithm significantly outperforms other baselines with a performance improvement of 8%∼17% with an average improvement of 10.2%.

2 Related work

In network science, the task of detecting dense modules from the network has been extensively studied for static networks [4,9] (see [1,10] for a comprehensive review). Several attempts have been made to detect local communities around a target node [2,3,11–15]. Some work has focused on detecting dynamic communities in evolving networks [16–19]. Recently, [20–22] attempted to discover communities from incomplete/noisy networks. All these methods assume that the entire network structure is available a priori, whereas we assume that only target nodes are given, and one needs to scan the nodes to explore the network structure. Every scan operation incurs a cost, and the network exploration can be possible till a certain budget is exhausted.

In the line of only network discovery (without community detection), there has been plenty of research focusing on general sampling techniques [23,24], sampling web documents [25], and sampling social network [26,27]. These techniques are not applicable in our settings because they initially consider the entire network structure for sampling. Moreover, none of them really focus on discovering the underlying community structure in the sampled network. There is also a lot of work on incremental community discovery of dynamic graph, such as [28–35]. These decentralized algorithms have similar scenarios as we choose here, but they do not consider the cost when the algorithm makes a query or explores one or more nodes in the graph. This implies our scenario is more difficult.

The most similar existing work with ours is `NetDiscover` [8] which also attempted to discover disjoint communities of a target node set. Although the problem definition is exactly the same, we differ from their method w.r.t. both candidate node selection and incremental community detection. In the former case, `NetDiscover` selects a query node based on one of the two community scoring metrics (modularity, normalized cut), whereas the proposed node selection algorithm is based on a novel metric *information gain*. A close inspection of `NetDiscover` reveals that there is an information leakage while selecting the query node in that it uses the entire network to compute the node scoring function. However, `NetDiscover` detects initial communities using spectral clustering [36] (where the number of communities needs to be given *a priori*) and adopts generative model (`GM`) [6] to update communities, whereas we combine both EM and MRF algorithms for community update. To compare with existing work, we consider both `NetDiscover` and commonly used random sampling and greedy algorithm as baselines (see Sect. 7.3). We also compare our method with **Quick Community Adaptation (QCA)** [37] and **Dynamic Permanence (DyPerm)**

[38] methods. QCA is an adoptive modularity-based approach for identifying and tracking community structure of dynamic network. DyPerm is another modularity maximization-based approach which incrementally updates the community structure of a network at t_i based on the community structure at t_{i-1} with detecting the community structure from the scratch. We adopt these algorithms to our setting (incomplete network) and make them comparable.

3 Problem statement

We denote a network as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set of edges. Assume that initially we do not know the entire network $\mathcal{G}$; only a partial subset of the network $\mathcal{G}_\mathrm{s} = (\mathcal{V}_\mathrm{s}, \mathcal{E}_\mathrm{s})$ (where $\mathcal{V}_\mathrm{s} \subseteq \mathcal{V}$ and $\mathcal{E}_\mathrm{s} \subseteq \mathcal{E}$) is known. Among all the nodes in $\mathcal{V}_\mathrm{s}$, we are particularly interested in detecting the community structure of a given set of target nodes $\mathcal{V}_\mathrm{T} \subseteq \mathcal{V}_\mathrm{s}$.

We iteratively scan nodes and explore the network with the neighborhood information of the scanned nodes. We use $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ to denote an intermediate network at the ith scan iteration (thus, initially $\mathcal{G}_0 = \mathcal{G}_\mathrm{s}$). The performance of community detection and the cost incurred by scanning nodes greatly depend upon the node selection strategy (i.e., which node to scan next). We assume a function $Q(v)$ denoting the cost to scan a node v. If v is a private node[2] and it does not allow a scan, then $Q(v) = \infty$. In the simple case, the cost to scan all non-private nodes may be same, i.e., $Q(v) = 1$. However, we also adopt a more general setting with heterogeneous cost per node. Another general setting is that the available budget B we invest for scanning is limited.

To this end, we consider two sub-problems: One is to decide candidate nodes[3] to scan next, and the other is to update the community structure incrementally.

Network scan Given a budget B, an intermediate network $\mathcal{G}_i$ with a cost function Q associated with it, and a target node set $\mathcal{V}_\mathrm{T}$, the aim of *candidate selection* is to decide the next candidates whose exploration of the local neighborhoods leads to the best community detection. After that, we actually scan the selected candidates, and $\mathcal{G}_{i+1}$ is generated from $\mathcal{G}_i$ and scan results of the candidates.

Community update Given $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$, the task of community update is to efficiently and effectively discover the community structure in $\mathcal{G}_{i+1}$ considering the community structure in $\mathcal{G}_i$ from the last iteration.

4 Overall algorithm

The purpose of integrating both network scan and community update steps is to effectively determine the community structure of a set of target nodes $\mathcal{V}_\mathrm{T}$ while incrementally obtaining network structure through scanning the nodes within a given budget B. Even in the case that a budget is not specified, the proposed method should be able to progressively enhance the community structure as network scan proceeds. Our method undergoes the following steps (see Fig. 1): (i) Select the k best candidates that are not scanned yet among all nodes in an intermediate network $\mathcal{G}_i$; (ii) scan the selected candidates to obtain their neighbors'

[2] A private node does not allow others to access its profile. Therefore, we cannot directly scan a private node.

[3] For efficiency, we allow to scan multiple nodes in an iteration.

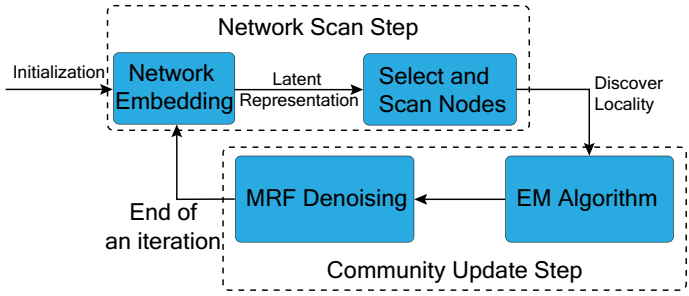


Fig. 1 Algorithmic flow of our proposed method

Algorithm 1 *CommunityDetection* (Seed network: $\mathcal{G}_s$, Max community numbers: K, Target nodes: $\mathcal{V}_T$, The number of nodes to scan k, Budget: B)

1: Scan all nodes in $\mathcal{V}_T$
2: $q = \sum_{v \in \mathcal{V}_T} Q(v)$ // Update total cost
3: $\mathcal{G}_0 = \mathcal{V}_T \cup \{u | u$ is a neighbor of $v \in \mathcal{V}_T\}$
4: **while** $q < B$ **do**
5: Choose a set V of k nodes to scan
6: $q = q + \sum_{v \in V} Q(v)$
7: Scan the selected nodes in V
8: Update $\mathcal{G}_{i+1}$ from $\mathcal{G}_i$ with the scan results
9: $\mathsf{C}_{i+1} = CommunityUpdate(\mathcal{G}_{i+1}, V, \mathsf{C}_i, K)$

information that has not been discovered so far; and (iii) simultaneously update the network $\mathcal{G}_i$ to generate $\mathcal{G}_{i+1}$ and its community structure. While iterating the above procedures, the currently available network structure $\mathcal{G}_i$ is used as the main information source to decide candidate nodes to scan. *It is critical to scan the nodes in the network in such an order that it facilitates the most efficient discovery of network communities while maintaining a low cost.*

Algorithm 1 alternately solves the two sub-problems we mentioned earlier. It starts by querying all the nodes in $\mathcal{V}_\mathrm{T}$ (line 1). This is to ensure that all the target nodes and their local network structures are fully scanned for an initial community assignment. We handle two sub-problems by two parts: the network scan part in lines 5–8, and the community update part in line 9. C_i is the community structure for $\mathcal{V}_\mathrm{T}$ at the ith iteration, and is incrementally modified using *CommunityUpdate*() function (Algorithm 2). K is the maximum number of communities the target set can have. The parameter K is internally used in the network scan and community update steps. We will discuss these two parts in detail in the following sections.

5 Network scan

The purpose of network scan is to discover unexplored parts of the network, where a node is scanned to know its full neighborhood structure. It mainly aims at exploring and acquiring more nodes and their connections, leading to the detection of better community structure. Since the effectiveness of the algorithm mainly depends on the scan sequence, the main focus of our network scan approach is to decide the best candidate nodes to be processed next given an intermediate network. The parameter k denotes the number of nodes chosen to be scanned.

Candidates for network scan Let S_i be a set of scanned nodes till iteration $(i-1)$. The neighbor set is defined as $\mathcal{N}_i = \{v | \exists u \in S_i, (v,u) \in \mathcal{E}\}$. The scan candidate set $\mathcal{C}_i = \mathcal{N}_i \setminus S_i$ contains only nodes that are not scanned among all nodes in $\mathcal{N}_i$.

There are many candidate selection algorithms for network scan [39,40]. Two of them are simple but widely used in various fields: random sampling and greedy sampling [41]. We will use these methods as baselines in Sect. 7. Soundarajan et al. [40] suggested to select the node that maximizes the total number of nodes scanned with the aim of exploring the complete network structure. Other methods were specifically designed for community detection [8]. These methods dynamically sample nodes from an intermediate network in such a way that a certain community quality measurement metric is expected to be improved. Liu et al. [8] selected nodes in such a way that the value of 'normalized cut' decreases or 'modularity' increases. Here, we briefly describe these two metrics and how we use them to design baseline methods.

Normalized cut Given K communities at iteration i, the normalized cut is defined as: $\sum_{k=1}^{K} \frac{\text{cut}(\mathsf{C}_k, \mathcal{G}_i - \mathsf{C}_k)}{\text{assoc}(\mathsf{C}_k, \mathcal{G}_i)}$, where $\text{assoc}(\mathsf{C}_k, \mathcal{G}_i)$ represents the total degree of nodes in C_k within $\mathcal{G}_i$ and $\text{cut}(\mathsf{C}_k, \mathcal{G}_i - \mathsf{C}_k)$ is the number of edge-cuts between C_k and all other remaining communities. The optimization of the above cost function is to minimize edge-cuts (connections) among different communities. The baseline algorithms used in the experiment follow Liu et al. [8]. We calculate the minimum normalized cut cost for each node in the candidate set $\mathcal{C}_i$—at this step, the correct community membership of candidates is not known, and thus all possible community assignments of candidates have to be tested after fixing the community structures of the scanned nodes in S_i. Among all the candidates in $\mathcal{C}_i$, k nodes leading to the minimum normalized cut are selected and added to the intermediate network. Here, we assume that the community structure does not change for all nodes but newly added ones in the intermediate network.

Modularity Modularity is used to evaluate the strength of partitioning a network into different communities: $\sum_{k=1}^{K} (e(\mathsf{C}^k, S) - a(\mathsf{C}^k, S)^2)$, where $e(\mathsf{C}^k, S)$ is the fraction of edges where both nodes are in the same community C^k, and $a(\mathsf{C}^k, S)$ is the fraction of edges that at least one node is in C^k. A high modularity value means dense connections between the nodes in a community but sparse connections between the nodes in different communities. We take the same approach as in normalized cut, i.e., testing with all possible community membership assignments of candidates, and choosing k candidate nodes that lead to the maximum modularity [8].

Here, we propose a new candidate selection method that does not incur any additional hidden cost. We utilize Deepwalk [42] and node2vec [43] to obtain latent feature vectors of nodes, i.e., network embedding into feature space. We first briefly introduce Deepwalk and node2vec and then discuss our proposed algorithm.

5.1 Network embedding algorithms

Deepwalk [42] and node2vec [43] are deep learning-based embedding methods to learn latent representations of nodes in a network. Deepwalk encodes social relations into a continuous vector space after modeling a series of random walks with a Natural Language Processing method. The key idea is that each visited node during a random walk can be considered as a word, and the random walk corresponds to a sentence. Node2vec learns a latent feature vector that maximizes the likelihood of maintaining the neighborhoods of the node.

The feature representation framework generally consists of two main parts: a random walk generator and a representation update. Both of the above two frameworks have the common

generator, and for representation updates, Deepwalk uses SkipGram [44] and node2vec uses a modified SkipGram with customizations. The generator takes a graph as input and randomly samples a path of a given length from the starting node which is uniformly chosen over all possible nodes in the network. Each node is a neighbor of the previous node in the path. Deepwalk and node2vec are both scalable, and their effectiveness for community detection is shown in [42,43,45,46].

Terminology 1 *(Latent representation)* A latent vector representation of a node v in $\mathcal{G}_i$ generated by a network embedding algorithm is an abstracted neighborhood information of v. Thus, two nodes with similar latent vectors are close neighbors.

5.2 Proposed candidate selection method—information gain

The proposed algorithm is based on the latent representation described above. In each iteration, we run a network embedding algorithm to update the representation of the nodes in the network. Let $LR_i(v)$ be a vector representing the latent information of a node $v \in \mathcal{V}_i$.

Information gain The information gain of a node v at iteration i is defined as $Gain_i(v) = ||LR_{i-1}(v) - LR_i(v)||_{1 \text{ or } 2}$, where $||\cdot||_1$ (resp. $||\cdot||_2$) means the L_1 (resp. L_2) vector norm. The higher the information gain, the greater the changes in network structures around v after the last scan. If $LR_{i-1}(v)$ and $LR_i(v)$ are same, it means there is no change around v.

If a candidate node c is discovered and added in the ith iteration (i.e., $LR_{i-1}(c)$ is not defined), then $Gain_i(v) = \alpha||LR_i(c)||$, where α accounts for the penalty for missing information in the last iteration. A node with low information gain has very stable and rigid neighborhood structure. It is unlikely that scanning such stable neighborhoods brings any drastic update in community structure. Thus, we choose top k nodes with the *highest information gain*. Throughout our experiments, the L_2 norm is considered for information gain, and α is set to 1.

5.3 Comparison of node selection metrics

In this section, we analyze the characteristics of three node selection metrics: normalized cut, modularity, and information gain. We first suggest two key factors to be considered while scanning a network as follows:

1. Scan as many communities as possible, and make the number of scanned nodes in each community as even as possible.
2. Scan actively around community boundaries (rather than core of the communities).

These two actions are crucial for the EM algorithm to better identify community structure. In general, community cores have very dense connections and are easier to detect [47]. However, connections are weak around community boundaries because their community memberships are ambiguous in many cases [9]. To help the EM algorithm in this hostile situation, we scan more around those community boundaries and evenly for each community. Our information gain is designed to meet these key factors.

In Table 1, we summarize key statistics of three metrics that can show how good a metric is w.r.t. the above two factors. We define the key statistics as follows:

1. Given a selected node v and its ground-truth community C_v, we calculate the ratio of the discovered nodes to the size of C_v, i.e., $\frac{\text{number of discovered nodes so far in } C_v}{|C_v|}$.
2. The number of ground-truth communities that v's neighbors belong to.

Table 1 Statistics of various metrics to scan a selected node in Coauthorship dataset [48]: (i) the average ratio of the number of discovered nodes to the number of all nodes in the same ground-truth community of top-5 candidates, and (ii) the average number of communities that the neighbors of top-5 candidates belong to

Candidate ranking	Algorithm	Discovered/total nodes	No. of neigh.' comm.
1st	Normalized Cut	0.06	5.5
	Modularity	0.08	5.6
	Information Gain	**0.05**	**6.3**
2nd	Normalized Cut	0.08	5.0
	Modularity	0.10	5.2
	Information Gain	**0.06**	**5.8**
3rd	Normalized Cut	0.09	4.9
	Modularity	0.12	4.7
	Information Gain	**0.07**	**5.5**
4th	Normalized Cut	0.11	4.5
	Modularity	0.15	4.3
	Information Gain	**0.10**	**5.0**
5th	Normalized Cut	**0.12**	4.0
	Modularity	0.17	3.9
	Information Gain	**0.12**	**4.7**

The best methods are shown in bold

In each iteration, the statistics of top-5 candidates are collected and the average across iterations is reported. The proposed information gain shows the lowest values for the first statistics and the highest for the second statistics, which indicates that it can (i) scan more communities than others (so that the average number of discovered nodes in a community is smaller than others given the same budget) and (ii) scan around community boundaries more actively.

Algorithm 2 *CommunityUpdate*(Intermediate network: $\mathcal{G}_i$, A set of scanned nodes: V, Community structure: C_i, Max community numbers: K)

for each $c \in V$ **do**
  // Let $\mathcal{N}_c$ be a set of neighbors of c. $\mathcal{N}_c$ was updated after scanning c.
  Initialize edge parameters $q_{cw}(h)$, where $w \in \mathcal{N}_c$
  while until parameters are converged **do**
    for each node v in $\mathcal{N}_c$ **do**
      Update θ_{vh}
      Update $q_{vw}(h)$, where $w \in \mathcal{N}_c$
    if the global update condition is met **then**
      // This part is a global update.
      for each community h in C_v **do**
        Update θ_{vh} for all nodes in $\mathcal{G}_i$
        Update $q_{vw}(h)$ for all edges in $\mathcal{G}_i$
  for each node w in $\mathcal{V}_i$ **do**
    // $Label_j$ means the community of w.
    Update $Label_j = \arg\max_k \theta_{wk}$
// Enhance the local and global update results using the Markov Random Field (MRF) denoising technique.
$MRFdenoising(Label, \mathcal{G}_i, \mathsf{C}_i, K)$

6 Community structure update

The task of this step is to update community membership of nodes in an intermediate network $\mathcal{G}_i$ based on (i) the community structure of $\mathcal{G}_{i-1}$, and (ii) new nodes discovered after scanning. Existing approaches involve both local and global updates. While local updates only consider new edges and nodes, global updates consider the whole intermediate network $\mathcal{G}_i$. The proposed update process consists of three steps—*Step 1*: an incremental local update based on the expectation–maximization [6]; *Step 2*: an intermittent global update to correct the local update; and *Step 3*: the MRF denoising [7] to further adjust both of the local and the global updates. The expectation–maximization (EM) algorithm is originally a part of the generative model suggested in [6]. The local update has less computational complexity. However, it may introduce errors due to the lack of information about the whole intermediate network.

The EM algorithm itself is very efficient but has a limitation. When the number of hidden variables (i.e., community structures in our case) to learn is large, it is known to be suboptimal. For our targeted community detection, however, we think that the EM method can still afford. In our method, however, the number of hidden variables is not as many as that of usual full community detection problems. Alternatives are other moment-based or spectral-based methods, e.g., hidden Markov model (HMM). However, the inference algorithm of HMM is not as cheap as that of the EM method [49]. We think that the EM algorithm is a good choice in our case considering the relatively small number of hidden variables to infer.

Thus, EM algorithm is applied in every iteration and a global update will be run periodically. After that, we further reduce errors introduced by the network updates using the Markov Random Field (MRF). We fully customize both of the EM and the MRF algorithms in the proposed community structure update method.

6.1 Expectation–maximization algorithm

The EM algorithm is an iterative method to find the maximum likelihood or maximum a posteriori estimate of parameters. It consists of parameter learning and estimation processes. For nodes in the network, the model is parameterized by θ_{vh} which represents the propensity that a node v has edges in a community h. θ_{vh} can be understood as a parameter that characterizes the number of edges. The product $\theta_{vh} \cdot \theta_{wh}$ is the expected number of edges in the community h that lie between nodes v and w. Let A be a matrix whose elements represent the number of edges between nodes. The number of edges, i.e., A_{vw}, is Poisson distributed around the expected value, according to the generative model in [6]. Thus, the probability of generating a graph $\mathcal{G}$ is:

$$P(\mathcal{G}|\theta) = \Pi_{v \leq w} \frac{(\Sigma_h \theta_{vh} \theta_{wh})^{A_{vw}}}{A_{vw}!} \exp(-\Sigma_h \theta_{vh} \theta_{wh})$$

We follow [6] for updates in the EM step of Algorithm 2. $q_{vw}(h)$ is a parameter in the update process:

$$q_{vw}(h) = \frac{\theta_{vh} \cdot \theta_{wh}}{\sum_h \theta_{vh} \cdot \theta_{wh}} \quad \text{and} \quad \theta_{vh} = \frac{\sum_w A_{vw} q_{vw}(h)}{\sqrt{\sum_{vw} A_{vw} q_{vw}(h)}}$$

The community label of a node v is h that maximizes the parameter θ_{vh}. The *CommunityUpdate*() function iterates the procedure for each scanned node $c \in V$. In Algorithm 2, for the local update, only edges that are linked to the nodes in $\mathcal{N}_c$ are initialized and updated

(lines 3–7). θ_{vh} and $q_{vw}(h)$ can be updated accordingly. In the maximization step, all nodes in $\mathcal{G}_i$ should be updated as parameter θ_{vh} changes, where $w \in \mathcal{V}_i$, and are also affected by nodes that are not in the locality of v. Local update usually produces small errors because $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$ are very similar in many cases.

However, it is also important to ensure that such errors do not become cumulative. Therefore, in Algorithm 2, a global update process (lines 10–12) is executed when the number of edges increases by at least 10% (i.e., the global update condition in line 8). Note that as network size increases, the global update happens less frequently.

We are mainly interested in detecting the community structure of $\mathcal{V}_\mathrm{T}$. This does not mean that only θ_{vh} (where $v \in \mathcal{V}_\mathrm{T}$) is needed to be calculated because θ_{vh} is strongly entangled with θ_{uh}, where $u \in \mathcal{V} \setminus \mathcal{V}_\mathrm{T}$ is a neighbor of v.

After the EM step, we assign an updated community label to each node in $\mathcal{G}_i$ (lines 13–15 of Algorithm 2). Lastly, we perform one more denoising process (line 17) after updating for scanned nodes.

6.2 Markov random field (MRF) denoising

The results of the community update operation from EM algorithm provide a good indication of actual community membership labels. However, errors naturally inhere in the process because it is an approximation of the true community assignment. We attempt to further eliminate the errors from the estimated community assignment. In this paper, we utilize conditional independence and clique factorization properties [7] of the Markov Random Field (MRF). Simply put, we can consider MRF as a generalization of the Markov Chain concept to graphs (Fig. 2). Thus a node's community is decided by its neighbors' community memberships. In fact, MRF is one of the most popular graphical inference methods such as Bayesian Network and Belief Propagation.

Note that the (observed) community results of EM algorithm are obtained from the (hidden) noise-free community structure. Our goal is to infer the noise-free hidden community structure from the results observed in the EM algorithm. Let $o_v \in \{1, 2, \ldots, K\}$ be the observed label of a node $v \in \mathcal{G}_i$, and h_v be the hidden actual community label. Given the observed noisy labels of all nodes, our goal is to recover the original noise-free community labels, considering the network connectivity of $\mathcal{G}_i$.

Since the noise level is likely to be small, there is a strong correlation among o_v, h_v and h_u, where v and u are neighbors of each other. This prior knowledge can be captured using MRF. This graphical model has two types of cliques, and each of them involves two nodes. The cliques of the form $\{h_v, o_v\}$ have an energy function that expresses the correlation between the two. Since there are multiple community labels, a closed form of the energy function between all observed and hidden label pairs is as follows:

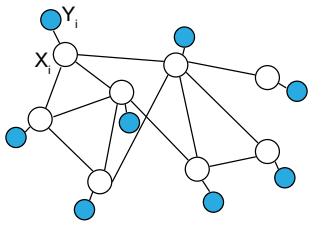


Fig. 2 MRF denoising model. X_i (white nodes) is the hidden variable denoting the actual community membership for each node, and Y_i (blue nodes) represents the corresponding observed variable obtained from the EM algorithm (colour figure online)

$$e(h, o) = -\eta \sum_{v} \min(1, |h_v - o_v|) \tag{1}$$

For each pair of labels, energy penalty is equal to 1 only if community membership is different, and 0 otherwise. This is desirable because it leads to a lower energy (i.e., high probability) when labels are the same, and a higher energy otherwise. η is a positive constant that needs a calibration.

The other cliques contain pairs of neighboring hidden node labels. Thus, $(v, u) \in \mathcal{E}_i$ for a pair h_v and h_u. Similarly, the energy is expected to be low when two neighbors have the same community label.

$$e(h, h) = -\beta \sum_{(v,u)\in\mathcal{E}_i} \min(1, |h_v - h_u|) \tag{2}$$

The complete energy function used to define a joint distribution is as follows: $p(h, o) = \frac{1}{Z}\exp\big(-e(h, o) - e(h, h)\big)$, where Z is a normalizing constant. To achieve better community updates, we wish to find h_j having a high probability $p(h, o)$. There are many algorithms to solve this optimization problem. Here, we use an iterative conditional modes technique (ICM) [50] which is a coordinate-wise gradient ascent method.

7 Experiments

In this section, we start by presenting the metrics used for evaluation, followed by detailed experimental results.

7.1 Evaluation metrics

The effectiveness should be measured in comparison with the true community labels of nodes. Note that the measure aims only for the target nodes since the goal of our community update is to obtain better community structure of the target set of nodes. There are in total at most K different communities in the network, and let $n_1, n_2, \ldots, n_m$ be the number of target nodes in the m different communities detected by the algorithms. The value of m may not be equal to the total number of communities K. Let f_{ij} be the fraction of target nodes in the estimated ith community that belong to the jth true community. Thus, we can find the true community that is most likely equivalent to the ith predicted community by $\arg\max_{j\in\{1,2,\ldots,K\}} f_{ij}$. In particular, $F_i = \max_{j\in\{1,2,\ldots,K\}} f_{ij}$. The reason we do this is that the estimated community may not have the same order of communities as the true labels; we have to find the mapping between estimated and true communities. Also, F_i is always in (0, 1]. In the ideal situation when all the nodes in the ith estimated community have the same true label, F_i is equal to 1. The Average Cluster Purity [8] is: ACP $= \frac{\sum_{i=1}^{m} n_i\ F_i}{\sum_{i=1}^{m} n_i}$. *A higher value of ACP indicates a better quality of the community structure*.

As the algorithm produces more estimated communities (i.e., m gets larger), ACP tends to be improved. Therefore, ACP may not always be a good metric; in particular when the number of communities is large, the ACP metric may be smaller compared with a small number of communities. We evaluate the performance using another measure called *Average Cluster Entropy* (ACE) that considers other estimated communities unlike F_i. Simply put, the entropy E_i for an estimated community i is $E_i = 1 - \sum_{j=1}^{K} f_{ij}^2$. The Average Cluster Entropy is defined as: ACE $= \frac{\sum_{i=1}^{m} n_i\ E_i}{\sum_{i=1}^{m} n_i}$. *A low value of ACE implies a high purity and a*

Table 2 Statistics of the datasets used in our experiments

Dataset	Nodes	Edges	Targets	Total Com.	Target Com.
DBLP	28,702	66,831	115	4	4
Coauthorship	90,302	352,184	1374	24	10
Synthetic	36,000	291,424	715	10	10
Amazon	334,863	925,872	602	75,149	20
YouTube	1,134,890	2,987,624	800	8385	40

We use five different networks with various sizes and characteristics

better community structure. An estimated community i that consists only of nodes from same true community will have the lowest entropy $E_i = 0$, and if the true labels of the estimated community i are evenly distributed over K different true communities, it will have entropy $(1 - 1/m)$.

7.2 Datasets

There are very few publicly available networks with *disjoint ground-truth communities*. We use the following networks in our experiments (see Table 2 for the statistics):

(i) DBLP network was collected by [8]. In this dataset, authors are considered as nodes and pairs of co-authors are connected with edges if they collaborated in a paper. Liu et al. [8] considered 115 authors from four real research groups led by Prof. Jiawei Han, Prof. Christos Faloutsos, Prof. Dan Roth, and Prof. Michael Jordan as target nodes.

(ii) Coauthorship network was released by [48]. It contains authors in Computer Science as nodes, and edges represent the co-author relationship. There are 24 disjoint ground-truth communities representing different research areas (Algorithm, AI, NLP, ML, etc.). It may be possible that an author has worked on multiple fields which causes the communities to overlap. To make it disjoint, we follow the approaches in [48]—assign each author to that research community in which he/she has published most papers. Total 1374 target nodes constituting 10 communities are randomly selected.

(iii) Synthetic network is a LFR network [51], consisting of 36,000 nodes. The average degree of a node is set to 8, and the number of nodes in a community is set in the range of [50, 100]. The target nodes are randomly sampled from 10 communities. There are at least 20 nodes in each target selected community.

Further in our experiments, we adopt 2 standard networks which contain known *overlapping community structure* [52], and pre-process the networks as follows—from each such network, we select those nodes as target nodes whose communities are completely disjoint. Then even though the underlying community structure of the entire network is overlapping, the ground-truth communities around the target nodes are disjoint. The networks are as follows:

(iv) Amazon network [52] is a co-purchase network, consisting of nodes as products, and two products are connected if they have been co-purchased by at least one customer. Products from the same category define a community. We randomly select 602 nodes constituting 20 communities that have no overlap.

(v) YouTube network [52] consists of users as nodes and friendships as edges. The ground-truth communities are user-defined groups. We randomly select 800 nodes constituting 40 communities that have no overlap.

7.3 Baseline algorithms

We consider several baseline methods for two different sub-problems (network scan and community update) separately. As the performance of community detection algorithm depends critically upon the order of nodes scanned, we test commonly used network scan algorithms while maintaining the same community update stated in Lines 3–15 of Algorithm 2 without the MRF denoising step. In particular, the following strategies mentioned in Sect. 5 were tested.

- **Random sampling** This algorithm randomly picks k nodes from Zipf (exponential) distribution in the intermediate network $\mathcal{G}_i$ that are not scanned, and searches their neighbors.
- **Greedy sampling** This algorithm selects the k nodes with the largest number of degree in the candidate node set.
- **Ratio of degree and entropy combination algorithm** This approach combines the greedy algorithm and community membership of the one-hop neighbors of the scanned node. The metric we compute for each node is $\frac{\text{entropy}}{\text{degree}}$, where *entropy* is computed with the community distribution of neighbors [8]. Specifically, when the neighbors are from one cluster, the entropy is small; it is large otherwise. We choose k candidates with the smallest metric value.
- **Normalized cut-based algorithm** This is a variation of [8] (see Sect. 5).
- **Modularity-based algorithm** This is another variation of [8] (see Sect. 5).

We once again emphasize that the last two strategies were mistakenly utilized by Liu et al. [8]; however, we use their original implementation without any modification. We will show that despite the information leakage in [8], our method still outperforms them across different datasets. We also compare our network scan algorithm with and without MRF denoising step to better illustrate the improvement due to the MRF denoising. In addition to this, we compare our method with two other incremental community detection methods: (i) **(QCA):** This framework uses a modularity-based approach for incremental community detection [37], and (ii) **(DyPerm):** It maximizes permanence, a local community-centric measure to detect communities [38]. DyPerm was shown to outperform most of the state-of-the-art incremental community detection methods (Fig. 3).

7.4 Sensitivity of parameters

In this section, we briefly describe our parameter selection strategy. The proposed method has three major parameters, the number of nodes to scan k and two MRF parameters β and

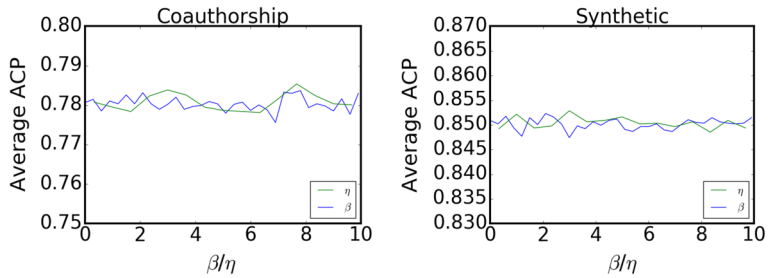


Fig. 3 ACP with values of β and η for Coauthorship, Synthetic and Amazon networks

η. We varied k from 1 to 10. Of course, $k = 1$ theoretically gives the best result. In many cases, however, we could not find any distinctive differences even for $k = 10$. Thus, we have chosen the median value $k = 5$. The MRF denoising performance varies up to 2% across different parameter settings (different values of β and η), which may not be significant. We can therefore conclude that the result of our method is less sensitive to parameter selection. In the rest of the section, we use the following parameter values as default: $k = 5$, $\beta = 8.8$ and $\eta = 1.9$.

7.5 Evaluation results

We run each experiment 20 times with random initialization of all parameters, and the average performance is shown. We conduct a threefold experimental setup—(i) the cost of scanning each node is equally set to 1 (Constant Cost); (ii) the cost varies across nodes (Varying Cost), and (iii) the impact of MRF denoising in our method.

7.5.1 Results without denoising

We discuss the experimental results of our method without MRF denoising as follows.

Constant cost In this experiment, the cost of scanning a node is set to 1. Figure 4 shows the performance of different network scan algorithms for all the networks. The performance for each algorithm is shown, with increasing values of the budget on the x-axis and the ACP/APE on the y-axis. Our proposed algorithm outperforms other baselines significantly in all cases. This proves the superiority of information gain over other scan metrics.

The greedy algorithm shows the worst performance even compared to random sampling (sometimes 20% less) as it adds a lot of noisy information to the network structure. Furthermore, it is the least stable one among all algorithms. On the other hand, modularity and normalized cut-based algorithms are expected to have better results as they use the information of candidates' neighbors. However, our algorithm (with both node2vec and Deepwalk) outperforms these baselines for all the datasets—Deepwalk shows slightly better performance than node2vec.

Varying cost To simulate real scenarios, we conduct experiments with various costs. The cost is generated according to the Zipf distribution as suggested in [8]—all nodes are randomly shuffled and $z(v) = 1/ind(v)^{\lambda}$, where $ind(v)$ is the index of node v after shuffling. The cost $Q(v)$ is the normalization of $z(v)$ over all nodes.

The ACP/ACE performance with varying costs shows the similar pattern —our method achieves better performance than the baseline methods. While at some points there are small fluctuations, the overall trend almost remains the same. This again confirms that our strategy is superior to its competitors in community detection.

7.5.2 Performance with denoising

Here, we show the results after including the MRF denoising mentioned in Sect. 6.2. Figure 5 shows the results before and after applying the MRF denoising step. For simplicity, we only show the ACP results (ACE results have similar trends). We observe that the MRF denoising can improve the ACP/ACE by 2%∼8%. The MRF denoising also improves the performance of baseline methods, e.g., the random sampling in Fig. 5. We argue that the MRF denoising has a positive effect on recovering actual hidden community memberships, and it can be generalized to many other strategies.

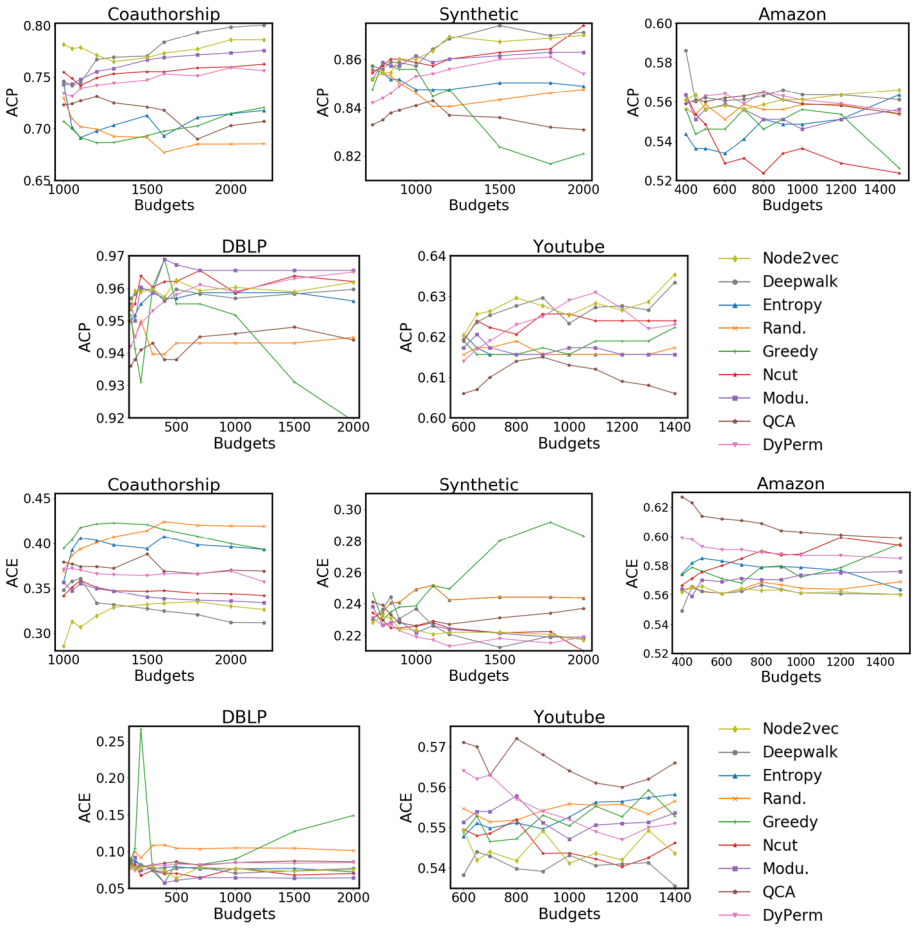


Fig. 4 Performance (ACP/ACE) of network scan and community update with *constant cost* and without the MRF denoising for five datasets. The higher (lower) the value of ACP (ACE), the better the performance

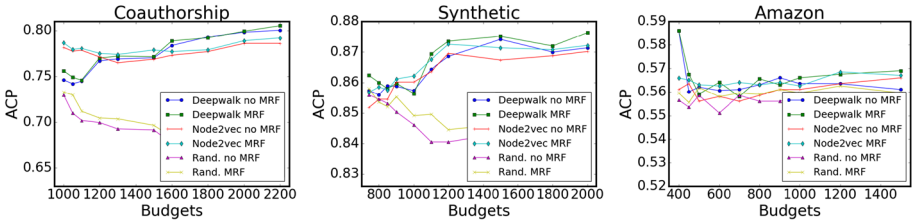


Fig. 5 MRF denoising performance (ACP) of our algorithm and random sampling methods

7.6 Summary of the experimental results

For all five networks, we report the performance (in terms of both ACP and ACE) of all the competing methods. We consider our complete method (node2vec/Deepwalk + EM algorithm + MRF denoising) and other existing baselines (without any modification). Table 3 shows

Table 3 Summary of the experimental results for all the datasets

Dataset	Coauthorship	Synthetic	Amazon	DBLP	YouTube	Average
[8]+Ncut						
ACP	0.76	0.87	0.52	**0.96**	0.62	0.74
ACE	0.34	0.21	0.59	0.07	0.55	0.35
[8]+Modu						
ACP	0.78	0.86	0.56	**0.96**	0.62	0.75
ACE	0.33	0.22	0.58	**0.06**	0.55	0.35
DyPerm [38]						
ACP	0.76	0.85	0.56	**0.96**	0.62	0.75
ACE	0.36	0.22	0.59	0.09	0.55	0.36
QCA [37]						
ACP	0.71	0.83	0.55	0.94	0.61	0.73
ACE	0.37	0.24	0.56	0.09	0.57	0.37
Random						
ACP	0.69	0.84	0.55	0.94	0.62	0.73
ACE	0.42	0.24	0.57	0.10	0.56	0.38
Greedy						
ACP	0.72	0.82	0.52	0.92	0.62	0.72
ACE	0.39	0.28	0.59	0.15	0.55	0.39
Entropy						
ACP	0.72	0.85	**0.56**	**0.96**	0.62	0.74
ACE	0.39	0.24	0.56	0.07	0.56	0.36
Our+Deepwalk						
ACP	**0.80**	**0.88**	0.56	**0.96**	0.64	**0.77**
ACE	**0.30**	0.22	**0.56**	0.08	**0.54**	**0.34**
Our+node2vec						
ACP	0.79	0.87	**0.57**	**0.96**	**0.64**	**0.77**
ACE	0.32	**0.21**	**0.56**	0.08	**0.54**	**0.34**

Bold values highlight the best performance. We consider the largest budget with constant cost and default parameter setting. The higher (lower) the value of ACP (ACE), the better the performance

that for all the networks, our method is as good as the best baseline [8] or even better than it. The ACP (ACE) values of our Deepwalk-based and node2vec-based methods averaged over all the datasets are the same, 0.77 (0.34), followed by [8]+Modu and [8]+NCut. In short, our method significantly beats the existing baselines over all the datasets.

One thing to note is that the proposed method usually takes 5–20 times longer than the existing `NetDiscover`[8] algorithm, while `NetDiscover` assumes every candidate node is scanned, and selects the best one but it only considers the cost of the selected node as the step for scanning step. The strict comparison of time between these methods may not be very useful.

8 Conclusion

In this paper, we studied a realistic setup for community detection—the entire network is not known a priori, and therefore one needs to progressively scan unknown nodes and

update the community structure around a target node set. The problem is divided into two sub-problems—network scan and community update. We proposed a novel method for each sub-problem. In the network scan step, a new metric *information gain* was designed to decide the best node to scan. A combination of the EM and MRF algorithm was proposed for the community update step to further recover the actual community labels of the nodes.

The major advances that the present work provides in the field of community detection are as follows:

- There are very few attempts made to process an incomplete network using an incremental way where one network construction reinforces the community detection that in turn helps discover better network structure. Most of the state-of-the-art algorithms assume that the static snapshot of the network is available beforehand, which may not be a realistic setting. Therefore, our problem definition is novel.
- The use of EM framework and Markovian denoising is the major technical novelty.

Our proposed method consistently achieved better performance (on average 10.2% higher than the best baseline) across five different datasets. We also make our experimental codes available in the spirit of reproducible research: https://github.com/ZheCui/MRFCommDetect.

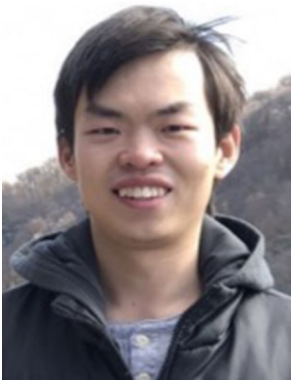
Acknowledgements T. Chakraborty would like to acknowledge the support of Ramanujan Fellowship, Early Career Research Award (ECR/2017/001691) (SERB, DST) and the centre for Design and New Media (supported by TCS), IIIT-Delhi. Noseong Park is the corresponding author.

References

1. Fortunato S (2010) Community detection in graphs. Phys Rep 486(3):75–174
2. Clauset A (2005) Finding local community structure in networks. Phys Rev E 72(2):026132
3. Luo F, Wang JZ, Promislow E (2008) Exploring local community structures in large networks. Web Intell Agent Syst Int J 6(4):387–400
4. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech Theory Exp 2008(10):P10008
5. Kim M, Leskovec J (2011) The network completion problem: Inferring missing nodes and edges in networks. In: Proceedings of the 2011 SIAM international conference on data mining, pp 47–58
6. Ball B, Karrer B, Newman MEJ (2011) Efficient and principled method for detecting communities in networks. Phys Rev E 84:036103
7. Bishop CM (2006) Pattern recognition and machine learning. Springer, Berlin
8. Liu J, Aggarwal C, Han J (2015) On integrating network and community discovery. In: WSDM, Shanghai, China, pp 117–126
9. Chakraborty T, Srinivasan S, Ganguly N, Mukherjee A, Bhowmick S (2016) Permanence and community structure in complex networks. ACM Trans Knowl Discov Data 11(2):14:1–14:34
10. Chakraborty T, Dalmia A, Mukherjee A, Ganguly N (2017) Metrics for community analysis: a survey. ACM Comput Surv 50(4):54:1–54:37
11. Nassar H, Kloster K, Gleich DF (2015) Strong localization in personalized PageRank vectors. In: International workshop on algorithms and models for the web-graph. Springer, pp 190–202
12. Yin H, Benson AR, Leskovec J, Gleich DF (2017) Local higher-order graph clustering. In: ACM Conference on knowledge discovery and data mining, pp 555–564
13. Liu C, Liu J, Jiang Z (2014) A multiobjective evolutionary algorithm based on similarity for community detection from signed social networks. IEEE Trans Cybern 44(12):2274–2287
14. Wang X, Liu J (2017) A layer reduction based community detection algorithm on multiplex networks. Physica A 471:244–252

15. Li Z, Liu J (2016) A multi-agent genetic algorithm for community detection in complex networks. Physica A 449:336–347
16. Chakrabarti D, Kumar R, Tomkins A (2006) Evolutionary clustering. In: ACM Conference on knowledge discovery and data mining, pp 554–560
17. Zhang J, Yu PS (2015) Community detection for emerging networks. In: Proceedings of the 2015 SIAM international conference on data mining, Vancouver, Canada, pp 127–135
18. Cheng J, Wu X, Zhou M, Gao S, Huang Z, Liu C (2018) A novel method for detecting new overlapping community in complex evolving networks. IEEE Trans Syst Man Cybern Syst 99(99):1–13
19. Wang Z, Zhang D, Zhou X, Yang D, Yu Z, Yu Z (2014) Discovering and profiling overlapping communities in location-based social networks. IEEE Trans Syst Man Cybern Syst 44(4):499–509
20. Lin W, Kong X, Yu PS, Wu Q, Jia Y, Li C (2012) Community detection in incomplete information networks. In: International conference on world wide web. Lyon, France, pp 341–350
21. Wang L, Wang J, Bi Y, Wu W, Xu W, Lian B (2014) Noise-tolerance community detection and evolution in dynamic social networks. J Comb Optim 28(3):600–612
22. Koujaku S, Kudo M, Takigawa I, Imai H (2015) Community change detection in dynamic networks in noisy environment. In: Proceedings of the international conference on World Wide Web, pp 793–798
23. Leskovec J, Faloutsos C (2006) Sampling from large graphs. In: International conference on knowledge discovery and data mining, pp 631–636
24. Ahmed NK, Neville J, Kompella R (2014) Network sampling: from static to streaming graphs. ACM Trans Knowl Discov Data 8(2):1–56
25. Baykan E, Henzinger M, Weber I (2013) A comprehensive study of techniques for url-based web page language classification. ACM Trans Web 7(1):1–37
26. Gabielkov M, Rao A, Legout A (2014) Sampling online social networks: an experimental study of twitter. ACM Comput Commun Rev 44(4):127–128
27. Lu J, Li D (2012) Sampling online social networks by random walk. In: International workshop on hot topics on interdisciplinary social networks research, pp 33–40
28. Yun S-Y, Proutiere A (2014) Community detection via random and adaptive sampling. In: Conference on learning theory, pp 138–175
29. Mahoney MW, Orecchia L, Vishnoi NK (2012) A local spectral method for graphs: with applications to improving graph partitions and exploring data graphs locally. J Mach Learn Res 13(8):2339–2365
30. Meng F, Zhang F, Zhu M, Xing Y, Wang Z, Shi J (2016) Incremental density-based link clustering algorithm for community detection in dynamic networks. Math Prob Eng 2016:1873504
31. Xie J, Chen M, Szymanski BK (2013) Labelrankt: incremental community detection in dynamic networks via label propagation. In: Proceedings of the workshop on dynamic networks management and mining, pp 25–32
32. Takaffoli M, Rabbany R, Zaïane OR (2013) Incremental local community identification in dynamic social networks. In: Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining, pp 90–94
33. Zakrzewska A, Bader DA (2015) Fast incremental community detection on dynamic graphs. In: International conference on parallel processing and applied mathematics, pp 207–217
34. Clementi A, Di Ianni M, Gambosi G, Natale E, Silvestri R (2015) Distributed community detection in dynamic graphs. Theor Comput Sci 584:19–41
35. Becchetti L, Clementi A, Natale E, Pasquale F, Trevisan L (2017) Find your place: simple distributed algorithms for community detection. In: Proceedings of the 28th annual ACM SIAM symposium on discrete algorithms, pp 940–959
36. Ng AY, Jordan MI, Weiss Y (2001) On spectral clustering: analysis and an algorithm. In: Advances in neural information processing systems. Vancouver, British Columbia, Canada, pp 849–856
37. Nguyen NP, Dinh TN, Xuan Y, Thai MT (2011) Adaptive algorithms for detecting community structure in dynamic social networks. In: IEEE International conference on computer communications, pp 2282–2290
38. Agarwal P, Verma R, Agarwal A, Chakraborty T (2018) Dyperm: maximizing permanence for dynamic community detection. In Pacific-asia conference on advances in knowledge discovery and data mining (PAKDD), pp 437–449
39. Li X, Wu B, Guo Q, Zeng X, Shi C (2015) Dynamic community detection algorithm based on incremental identification. In: 2015 IEEE International conference on data mining workshop, pp 900–907
40. Soundarajan S, Eliassi-Rad T, Gallagher B, Pinar A (2016) Maxreach: reducing network incompleteness through node probes. In: ASONAM, San Fransisco, CA, USA, pp 152–157
41. Vitter JS (1985) Random sampling with a reservoir. ACM Trans Math Softw (TOMS) 11(1):37–57
42. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: SIGKDD, New York, USA, pp 701–710

43. Grover A, Leskovec J (2016) Node2vec: scalable feature learning for networks. In: SIGKDD, San Francisco, CA, USA, pp 855–864
44. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space, *CoRR*, arXiv:1301.3781
45. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: large-scale information network embedding. In: WWW, Florence, Italy, pp 1067–1077
46. Chang S, Han W, Tang J, Qi G-J, Aggarwal CC, Huang TS (2015) Heterogeneous network embedding via deep architectures In: SIGKDD, Sydney, Australia, pp 119–128
47. Seifi M, Junier I, Rouquier J-B, Iskrov S, Guillaume J-L (2013) Stable community cores in complex networks. In: Menezes R, Evsukoff A, González MC (eds) Complex networks. Springer, Berlin, Heidelberg, pp 87–98
48. Chakraborty T, Srinivasan S, Ganguly N, Mukherjee A, Bhowmick S (2014) On the permanence of vertices in network communities. In: Proceedings of international conference on knowledge discovery and data mining, pp 1396–1405
49. Khreich W, Granger E, Miri A, Sabourin R (2010) On the memory complexity of the forward–backward algorithm. Pattern Recogn Lett 31(2):91–99
50. Besag J (1986) On the statistical analysis of dirty pictures. J R Stat Soc. Ser B (Methodol) 48(3):259–302
51. Lancichinetti A, Fortunato S (2009) Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. Phys Rev E 80:016118
52. Leskovec J, Krevl A (2014) SNAP Datasets: stanford large network dataset collection, http://snap.stanford.edu/data. Accessed May 2018



Zhe Cui received a B.E. degree from Zhejiang University, Hangzhou, China, in 2013 and an M.S. and Ph.D. degree from University of Maryland, College Park, USA, in 2018 and 2019, respectively. His research interests include recommendation systems, machine learning, and data mining.

Noseong Park is an Assistant Professor at the Dept. of IST, George Mason University, since August 2018. Prior to this, he was an assistant professor at the University of North Carolina at Charlotte. He completed his Ph.D. at the University of Maryland at College Park in 2016. His primary research interests include data mining and applied machine learning.

Tanmoy Chakraborty is an Assistant Professor and a Ramanujan Fellow at the Dept. of CSE, IIIT Delhi, India since May 2017. Prior to this, he was a postdoctoral researcher at University of Maryland, College Park, USA. He completed his Ph.D. as a Google India Ph.D, fellow at IIT Kharagpur, India in 2015. His primary research interests include social network analysis, graph mining, and natural language processing. He is a recipient of several awards, including two Faculty Awards from Google, Early Career Research Award, DAAD Faculty award.