Information Sciences 425 (2018) 92–106

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Hidden community detection in social networks

Kun He[a,b,*], Yingru Li[a,*], Sucheta Soundarajan[c], John E. Hopcroft[b]

[a] Huazhong University of Science and Technology, Wuhan, China
[b] Cornell University, Ithaca, New York
[c] Syracuse University, Syracuse, New York

ARTICLE INFO

Article history:
Received 19 September 2017
Revised 4 October 2017
Accepted 7 October 2017
Available online 9 October 2017

Keywords:
Community detection
Hidden community
Structure mining
Social networks

ABSTRACT

This paper introduces a new graph-theoretical concept of hidden community for analysing complex networks, which contain both stronger or dominant communities and weak communities. The weak communities are termed as being with the *hidden community structure* if most of its members also belong to the stronger communities. We propose a meta-approach, namely *HICODE* (HIdden COmmunity DEtection), for identifying the hidden community structure as well as enhancing the detection of the dominant community structure. Extensive experiments on real-world networks are carried out and the obtained results demonstrate that *HICODE* outperforms several state-of-the-art community detection methods in terms of uncovering both the dominant and the hidden structure. Due to the difficulty of labeling all ground truth communities in real-world datasets, *HICODE* provides a promising technique to pinpoint the existing latent communities and uncover communities for which there is no ground truth. Our finding in this work is significant to detect hidden communities in complex social networks.

1. Introduction

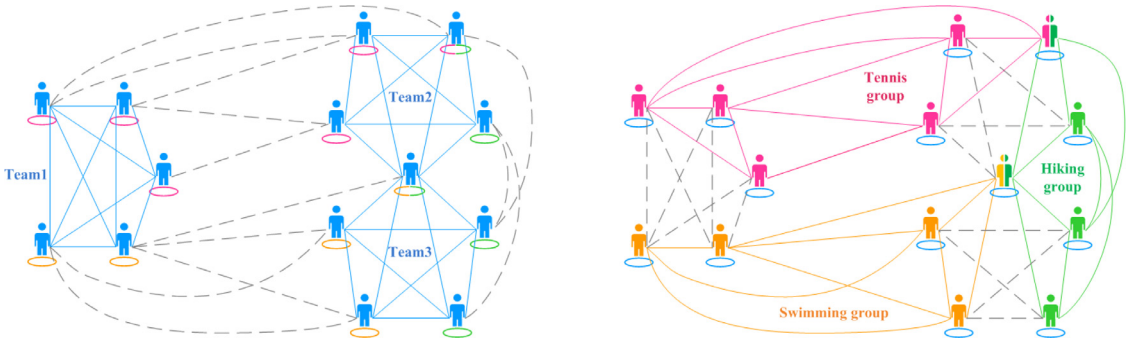
Over the past decades, community detection has emerged as an essential task in the realm of network analysis, and provides insight into the underlying structure and potential functions of the networks [1,2]. Early work focused primarily on identifying disjoint communities that partition the set of nodes within a network [3,4]. More recently, researchers have observed the multiplicity of interwoven memberships of communities and have developed algorithms for finding overlapping communities [5,6]. Some partitioning techniques are also extended to tackle the overlapping case [7,8]. Within these two categories, one can further build a hierarchical dendrogram based on the granularity of the detected communities.

Although much progress has been made, there is a type of new community structure, which we call the **hidden community structure**, that has attracted little attention in the literature. Real-world networks contain sparse community structure, such as secret organizations or temporary groups, which is considerably weaker than the dense community structure like families, colleagues or close friends, as evaluated by popular community scoring metrics. If most of the members in the less modular community also belong to other denser communities, the community is usually overlooked.

For instance, in a social network, individuals may belong to multiple strong social communities, corresponding to groups such as families, colleagues and friends. Though overlapping, the connections inside these communities are strong and nu-

* Corresponding authors.
E-mail addresses: brooklet60@hust.edu.cn, kh555@cornell.edu (K. He), szrlee@hust.edu.cn (Y. Li), susounda@syr.edu (S. Soundarajan), jeh@cs.cornell.edu (J.E. Hopcroft).

https://doi.org/10.1016/j.ins.2017.10.019
0020-0255/



(a) Dominant Communities (b) Hidden Communities

Fig. 1. The illustration of the dominant communities and the hidden communities in a social network. (a) The three cliques correspond to communities of students working closely as teams in projects. (b) The three groups of different colors correspond to sports communities with sparser connections. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

merous enough that existing overlapping community detection algorithms can perceive and uncover these latent but dominant modular structures. However, in addition to these strong communities, individuals may also belong to some weaker communities, such as a group of medical patients that see each other at the doctor's office and communicate infrequently, or high school alumni whom have infrequent contact. As illustrated in Fig. 1, the hidden community structure is sparser and harder for detection.

For applications in a large variety of scientific disciplines, the hidden structure is of great interest and deserves to be explored. For example, in Protein-Protein Interaction (PPI) networks [9], biologists wish to identify gene groups serving similar functions. However, the current annotation is far from complete [10], and the dominant, clearest groupings are more likely to be annotated. In such a scenario, a way to help the biologists identify the hidden, less obvious groups is of great value. Another example could be a criminal organization in a social network that is much weaker than communities corresponding to family or location. Identifying this hidden structure in the presence of the stronger community structure is even more important but faces a major challenge.

This paper aims to provide insights into the hidden structure. We define the **hiddenness value** of a community as the portion of nodes in stronger communities, and present a meta-approach called Hidden Community Detection (*HICODE*) to identify the dominant structure as well as the hidden structure in networks. *HICODE* begins by first applying an existing algorithm as a base algorithm to a network, and then weakening the structure of the detected communities in the network. In this way, the weaker, hidden community structure becomes visible. This step is repeated iteratively until no further significant structure is detected. Next, *HICODE* weakens the structure of the hidden communities, and thus obtains a more accurate version of the dominant community structure.

Hidden community structure can be regarded as a special type of overlapping communities. However, existing overlapping community detection methods mainly focus on communities in which a considerably portion of the members are not "hidden", that is, they could also belong to other weaker communities but this community is clearly the strongest for these members. Our experiments also show that they overlook the "hidden communities" while *HICODE* uncovers the hidden structure much preciser.

We believe the insights we obtained on hidden community structure will provide valuable guidance for future investigations. The main contributions of this paper include:

- **Conception on Hidden Community.** We introduce the concept of hidden community structure that exists widely in social networks, and we formally define the hiddenness value of a community.
- **Methods on Hidden Community Detection.** We present *HICODE* for identifying both the dominant and the hidden structure. We implement *HICODE* with several community detection algorithms as the base algorithm, and provide several structure weakening methods: RemoveEdge, ReduceEdge and ReduceWeight.
- **Validation on real world datasets.** Through experiments on a variety of real-world networks, we demonstrate that the higher the hiddenness value a community is, the harder for an algorithm to locate such community; *HICODE* outperforms several state-of-the-art community detection methods on uncovering the hidden communities.

2. Related work

In the past decades, a plethora of community detection algorithms have been presented for uncovering communities, the latent modular structure, based on different metrics and techniques. We give a brief summary for related works in the area of clustering and community detection. For comprehensive reviews to various community detection algorithms and techniques, please refer to survey papers [11,12].

2.1. Conventional community detection

Disjoint community detection. Several popular partitioning algorithms are based on modularity optimization, including the Clauset-Newman-Moore algorithm [2] and the Louvain method [4]. Other algorithms use random walks, with the intuition that a good community will 'trap' a random walk (*e.g., Walktrap* [13] *and Infomap* [3].) Model-based algorithms fit the observed network to a model of network structure. Such models include the stochastic block model, in which the nodes are partitioned into disjoint groups and the probability of a connection between two nodes depends only on the memberships of the nodes [14,15].

Overlapping community detection. The first algorithm to examine overlapping communities was the Clique Percolation algorithm [16], where a community is defined as a maximal $k-$clique percolation chain. More recently, other methods based on cliques or seeds expansion are proposed. For example, OSLOM uses a fitness function and joins together small clusters into statistically significant larger clusters [6], and Whang identifies overlapping communities by expanding 'seeds' into full communities [17]. Link communities (LC) is a landmark algorithm that finds communities by performing hierarchical clustering on the links, which results in overlapping communities of nodes [5]. Another type of overlapping community detection algorithms identifies communities by expanding 'seeds' into full communities [17], while some algorithms, for example DEMON [18], use a label propagation approach.

2.2. Works closely related to hidden structure

To the best of knowledge, this is the first work to formally propose and address the hidden community detection problem. In this subsection, we introduce closely related works in the area of data clustering and community detection.

Multi-view data clustering. Outside the realm of community detection, researchers have studied the problem of clustering data into multiple alternative groupings. By adopting orthogonal clustering and clustering in orthogonal subspaces, Cui et al. [19] cluster the data points in different views, where data points of one cluster can belong to different clusters in other views. By augmenting a spectral clustering objective function to incorporate dimensionality reduction and multiple views, and to penalize for redundancy between the views, Niu et al. proposed approaches to learn non-redundant subspaces that provide multiple views simultaneously [20] and iteratively [21].

Hidden community detection. In the realm of community detection, similarly to multi-view clustering, there are several works on multi-valued attributed graph that cluster graphs into densely connected components with homogeneous attribute values [22,23], but they still focus on strong communities. For hidden community detection, there are several pioneer and embryonic works that are proposed independently and almost simultaneously [24–26].

Chen et al. [24] remove nodes or edges based on the local Fiedler vector centrality (LFVC) which is associated with the sensitivity of algebraic connectivity to node or edge removals. Most importantly, they define a concept of deep community as a connected component that can only be seen after removal of all nodes or edges from the rest of the network. They prove that their method works on small synthetic networks under stochastic block model framework and do experiments on small networks of size hundreds.

Young's work [25] is most similar in spirit to our work, and they also reference our first version of the work [26]. They observe that smaller or sparser communities can be 'overshadowed' by the larger or denser communities, and communities may appear at different resolutions. Their Cascade approach uses two existing algorithms as base algorithms (LC [5] and CFinder [16]), find the first set of communities, remove all internal edges for the current detected communities, find a second set and repeated the process until no significant communities could be found. Note that communities are fixed once extracted. Cascade differs from *HICODE* in two ways. First, instead of strategically reducing the structure of a detected community, they simply remove all edges within the community. This corresponds to our RemoveEdge community reduction strategy (*described in detail in* Section 4.2), which performs poorly, and which was included only for comparison. Second, while *HICODE* contains both an Identification stage, in which rough copies of community layers are obtained, and a Refinement stage, in which the community quality is improved, Cascade only performs the Identification stage. One consequence of this is that the effects of hidden communities on stronger communities are never accounted for.

In our first version of the work [26], we present Hidden Community Detection (*HICODE*), an algorithm template that identifies both the strong, dominant community structure as well as the weak, hidden community structure in networks. In [26], we implement *HICODE* using three disjoint community detection methods as the base algorithm, provide detailed experiments on two synthetic networks (synL2 and synL3) and two small social networks (Grad and UGrad [27]), and show that Cascade behaves very similar to LC and they did not really uncover the weak, hidden communities that actually are incoherent with the dominant communities. In a survey for data and network analysis [28], and Teng introduces our "hidden community" as a new graph-theoretical concept.

This paper is an significant extension of our arXiv version [26]. We provide a formal definition of "hidden community", implement *HICODE* with both disjoint community detection algorithms and overlapping community detection algorithms as the base algorithm, improve the algorithm description and provide a mass of experiments on 11 larger and popular real-world networks as well as synthetic networks to extensively demonstrate the *HICODE* method.

3. Preliminaries

Let graph $G = (V, E)$ represent a network with n nodes and e edges. Let $\mathbf{A}$ be the adjacency matrix of G, the ij-entry $A_{ij} \in \{0, 1\}$ indicates whether there is an edge connecting nodes i,j.[1] Let $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$ be all the overlapping communities, where $C_k = (V_k, E_k)$ is a comparatively dense subgraph of G. The cardinality of a community is defined as the number of nodes in this community, i.e. $|C_k| = |V_k|$. We first introduce some necessary metrics, then provide formal definitions on the hidden structure.

3.1. Metrics

Modularity. To measure the strength of a set of communities that partition the network, we adopt the popular *modularity* metric. Modularity is defined by Newman as the ratio of the number of intra-community edges to the expected number of edges in the same set of communities if the edges had been distributed randomly while preserving degree distribution [2].

The modularity score $\mathcal{Q}$ of a partition is calculated by:

$$\mathcal{Q} = \sum_{k=1}^{K} \mathcal{Q}_k = \sum_{k=1}^{K} \left[\frac{e_{kk}}{e} - \left(\frac{d_k}{2e} \right)^2 \right], \tag{1}$$

where K is the number of communities in the partitioning, e is the number of edges in the graph, e_{kk} is the number of edges within community C_k, and d_k is sum of the node degrees in community C_k. $\mathcal{Q}$ lies in the range $[-0.5, 1)$, with larger values indicating a stronger community set.

Definition 1 Modularity of a Community. We call $\mathcal{Q}_k$ the sum modularity contribution of C_k, and the modularity of a single community is defined as the sum modularity contribution of that community divided by the number of nodes in the community, i.e. $\mathcal{Q}_k/|C_k|$.

Zhang et al. extend Newman's definition to a set of overlapping communities by considering the belonging coefficient w_{ik} for node i to community C_k [7]. Briefly, $w_{ik} = 1/m$ if community C_k is one of the m communities containing node i. Then in Eq. (1), e_{kk} is weighted by $e_{kk} = \frac{1}{2} \sum_{i,j \in C_k} \frac{w_{ik} + w_{jk}}{2} A_{ij}$, where A_{ij} is the ij-entry of the adjacency matrix. We know $d_k = 2e_{kk} + e_{k\_out}$, then e_{k_out} is weighted by

$$e_{k\_out} = \frac{1}{2} \sum_{i \in C_k, j \notin C_k} \frac{w_{ik} + (1 - w_{jk})}{2} A_{ij}.$$

The extended definition degenerates exactly to Eq. (1) for disjoint communities.

Normalized Mutual Information (NMI). We use *normalized mutual information (NMI)* to capture the similarity of two partitions X and Y [29].

$$NMI(X, Y) = \frac{2I(X, Y)}{h(X) + h(Y)}, \tag{2}$$

where $h(X)$ is the Shannon entropy of partition X, and $I(X, Y)$ is the *mutual information* that captures the similarity between two partitions X and Y. For a set of communities that partition a network, $x \in X$, $y \in Y p(x) = |x|$ denotes the number of nodes in community x, , and $p(x, y) = |x \cap y|$ denotes the number of common nodes in the two communities. The NMI score lies in the range [0,1], where 1 represents a perfect matching and 0 indicates total independence.

$$I(X, Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \frac{p(x, y)}{p(x)\, p(y)} \tag{3}$$

$$h(X) = -\sum_{x \in X} p(x) \log p(x) \tag{4}$$

For overlapping communities, the extended NMI [30] is chosen as the metric.

3.2. Definitions

Assume we have some metric function $\mathcal{F}$:$(G, C_k) \mapsto \mathbb{R}$ (*e.g., modularity* [2] *or conductance* [31]) that assigns a quality score to a community. For simplicity, let $\mathcal{F}_k$ denote the quality of C_k in G. Here, we assume that higher scores indicate stronger communities, but the definition below can be trivially modified for the case when lower scores indicate higher community quality.

[1] For simplicity, we discuss unweighted graph, but all discussions in this paper are easily extended to weighted graph by letting $A_{ij} \in [0, 1]$ to indicate the weight of each edge. In the weighted graph, the number of edges then corresponds to the sum of the edge weights.

Definition 2 Hiddenness Value of a Community. The hiddenness value $\mathcal{H}(C_k)$ of community C_k is the fraction of nodes of C_k belonging to various communities with a higher $\mathcal{F}$ score.

Let $\mathbb{S}_k$ be the set of all stronger communities for community C_k, a.k.a. all communities with higher quality score.

$$\mathbb{S}_k = \{C_i | \mathcal{F}_i > \mathcal{F}_k, C_i \in \mathcal{C}\} \tag{5}$$

The hiddenness value of C_k can be calculated as:

$$\mathcal{H}(C_k) = \frac{1}{|C_k|} \cdot \left| \bigcup_{C_i \in \mathbb{S}_k} C_i \cap C_k \right|, \tag{6}$$

a.k.a. the fraction of nodes of C_k in stronger communities.

$\mathcal{H}(C_k) \in [0, 1]$. Intuitively, the higher a community's hiddenness value is, the more difficult for the community to be uncovered.

The goal of the hidden community detection problem is to locate overlapping communities in the network such that communities having high hiddenness values can be found. Note that there is no single threshold for a 'high' hiddenness value, these values depend on the network under study, the metric being used, and the set of communities. For two communities $C_i, C_j \in \mathcal{C}$, if $\mathcal{H}(C_i) > \mathcal{H}(C_j)$, then C_i is comparatively hidden as compared with C_j, and C_j is comparatively dominant as compared with C_i.

For convenience, we separate the communities into layers.

Definition 3 Layer. A layer $\mathcal{L}$ is a set of communities that partitions or covers the nodes of the network, indicating that

$$\forall v \in V, \exists C \in \mathcal{L} \rightarrow v \in C$$

Here we allow trivial communities of size less than 3.

Definition 4 Hiddenness Value of a Layer. The hiddenness value $\mathcal{H}(\mathcal{L}_i)$ of a layer $\mathcal{L}_i$ is the weighted average hiddenness values of the communities in this layer.

$$\mathcal{H}(\mathcal{L}_i) = \frac{\sum_{C_k \in \mathcal{L}_i} |C_k| \cdot \mathcal{H}(C_k)}{\sum_{C_k \in \mathcal{L}_i} |C_k|}. \tag{7}$$

$\mathcal{H}(\mathcal{L}_i) \in [0, 1]$. The dominant layer is the layer with lowest hiddenness value. It is usually the set of communities found by a standard community detection algorithm that optimizes metric $\mathcal{F}$. A layer is called hidden if it has a comparatively high hiddenness value. Fig. 1 illustrates a small social network with two layers of communities, the dominant layer corresponds to project teams and the hidden layer corresponds to sports groups.

4. Hidden community detection

4.1. Algorithm overview

In this section, we propose a meta-approach called *HICODE* to find hidden community structure in a network. *HICODE* uses an existing community detection algorithm as the 'base' method, and iteratively weakens the structure of detected layers to reveal hidden structure. *HICODE* then applies a refinement procedure to increase the quality of the layers. Algorithm 1 shows the pseudo code.

HICODE contains two stages: **Identification** and **Refinement**.

Stage 1. Identification:

The Identification stage determines the initial layers of communities as follows:

1. Identify a layer of communities via the base method;
2. Weaken the structure of the detected layer;
3. Repeat until the appropriate number of layers are found.

In this stage, we iteratively identify a set of initial layers by weakening the structure of the previous, stronger layers.

Section 4.2 presents methods for weakening the detected community structure in order to reveal the hidden structure beneath.

A crucial aspect of the Identification stage is to automatically determine the number of layers n_L in a network. This is accomplished by increasing n_L until a stopping condition is met, described in detail in Section 4.3.

Stage 2. Refinement:

It is reasonable that stronger community structure can obscure weaker community structure, but critically, we observe that weaker structure can also hinder the accurate or complete detection of the stronger structures. After the Identification stage, one has only a rough approximation of the various community layers, and the purpose of the Refinement stage is to further improve the quality of these detected layers.

Refinement is an iterative process. In each iteration, we consider each layer, and improve the current layer as follows:

Algorithm 1 HICODE.

Require: graph $G = (V, E)$,
base algorithm $\mathcal{A}_{base}$,
reduce method $\mathcal{M}_{Reduce}$,
iterations $T = 100$ as the default,
number of layers n_L.
Ensure: n_L layers of communities$\{L_1, \cdots, L_{n_L}\}$
$G_R \leftarrow G$
for $k = 1 : n_L - 1$ **do** ▷ Identification Stage
  Identify a layer$L_k \leftarrow \mathcal{A}_{base}(G_R)$
  Using $\mathcal{M}_{Reduce}$ to weaken the structure of the detected layer L_k and get Reduced Graph $G_R \leftarrow \mathcal{M}_{Reduce}(G, L_k)$
for refinement iteration $t = 1 : T$ **do** ▷ Refinment Stage
  for $i = 1 : n_L$ **do**
    Using $\mathcal{M}_{Reduce}$ to weaken the structure of **All Other** detected layers $L_{k\neq i}$ from original graph G and get reduced graph G_R
    Identify the current layer$L_i \leftarrow \mathcal{A}_{base}(G_R)$
return the final n_L layers $\{L_1, \cdots, L_{n_L}\}$

1. Weaken the structures of *all other* layers from the original network to obtain a *reduced network*;
2. Apply the base algorithm to the resulting network.

In contrast to the Identification stage, where only the layers found so far (i.e., the stronger layers) are reduced, during the Refinement stage, we weaken the effects of both the stronger and weaker layers. This is necessary because the weaker layers can impair detection of the layer currently under consideration, even though they have a smaller impact on the network structure than the stronger layers. Through this process, a more accurate version of the current layer is produced.

We use NMI [29,30] to capture the similarity of two partitions or two sets of overlapping communities. For the synthetic data that we consider, the Refinement stage quickly converges within 30 iterations, meaning that the NMI of the corresponding layers at adjacent iterations is almost 1. For real-world datasets, 100 iterations is generally enough.

Although the trend over these iterations is to see higher quality community layers (i.e., higher modularity partitions or coverings), there are fluctuations in this trend. In each Refinement iteration, we calculate the modularity of the detected layers, and the final output corresponds to the iteration with the highest average modularity score.

4.2. Reducing methods

We present the following methods to reduce a single layer of community structure: RemoveEdge, ReduceEdge, and ReduceWeight.

4.2.1. Removeedge

RemoveEdge weakens a detected community by removing all intra-community edges. This method is inspired by algorithms that find communities by removing edges, such as the classic Girvan–Newman algorithm, which, removes edges with high betweenness [1]. RemoveEdge works reasonably well when there are few layers and communities in different layers have comparatively small overlaps.

4.2.2. Reduceedge

ReduceEdge approximates each layer as a single stochastic blockmodel, where other edges are regarded as background noise. This method randomly removes some edges within each community block so that the edge probability in the block matches the background edge probability of this block.

For each community C_k in a single layer that we want to reduce, we calculate the observed edge probability p_k in C_k, and the background block probability q_k, as illustrated in the reordered adjacency matrix in Fig. 2. Let n_k and e_{kk} be the number of nodes and edges inside C_k, and d_k the sum of degrees of nodes in C_k. We regard p_k as the actual number of edges in C_k divided by the maximum possible number of edges:

$$p_k = \frac{e_{kk}}{0.5n_k(n_k - 1)}, \tag{8}$$

and regard q_k as the average outgoing edge density of block C_k:

$$q_k = \frac{d_k - 2e_{kk}}{n_k(n - n_k)}. \tag{9}$$

If we treat the observed edge probability p_k in community C_k as the superposition of the underlying edge probability p'_k of C_k and the background block probability q_k, then $p_k = 1 - (1 - p'_k)(1 - q_k)$. The conditional probability that it is generated

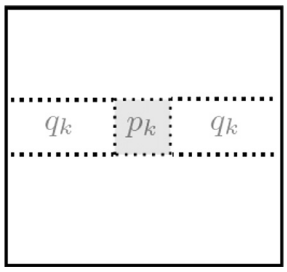


Fig. 2. p_k and q_k of a community block C_k.

by the background noise given an edge in C_k is:

$$q'_k = \frac{q_k}{p_k}. \tag{10}$$

ReduceEdge removes each edge within community C_k with probability $1 - q'_k$ (i.e., it keeps each internal edge with probability q'_k). In this way, edges are randomly removed from C_k such that the edge probability within C_k matches q_k.[2]

4.2.3. Reduceweight

This method reduces the weight of each edge within community C_k by a factor of q'_k, defined in Eq. (10). Like with ReduceEdge, we wish to set the weighted probability within C_k equal to the average weighted background block probability q_k.

Note that to use ReduceWeight, one's base algorithm must support weighted networks. However, the original network itself need not be weighted, as one can simply set the original weight of every edge to 1. Unlike ReduceEdge, ReduceWeight is deterministic. ReduceWeight is actually a derandomization of ReduceEdge.

For all cases, if one has a layer of overlapping communities, then in order to avoid duplicate weakening on the overlapping portions, we sort the communities according to their sizes, weaken larger communities first and do not weaken the overlapping portion again for subsequent communities in the same layer.

As we will see in experiments, ReduceWeight is the best-performing method of the three, and ReduceEdge generally performs better than RemoveEdge. However, one major advantage of ReduceEdge is that it does not require a base algorithm that supports weighted networks.

4.3. Selecting the number of layers

A major challenge for *HICODE* is determining n_L, the appropriate number of community layers.

4.3.1. Observations

To give intuition on how to automatically select the number of community layers n_L in a network, and the importance of selecting the appropriate number, we first analyze the synthetic networks presented earlier, in which the true communities and the correct number of layers are known.

We consider a variety of values for n_L, and make the following observation. If n_L is equal to the correct number of layers, then the similarity (NMI) between each detected layer L_i and the corresponding planted layer PL_i increases dramatically over the course of the Refinement stage. That is, as we refine the results, the detected communities become more and more like the ground truth communities. In contrast, if n_L is either too small or too large, then this trend is much reduced.

Additionally, if the n_L is chosen correctly, then during the Refinement stage, the average modularity of the detected layers increases. If n_L is either over- or under-estimated, then this trend declines.

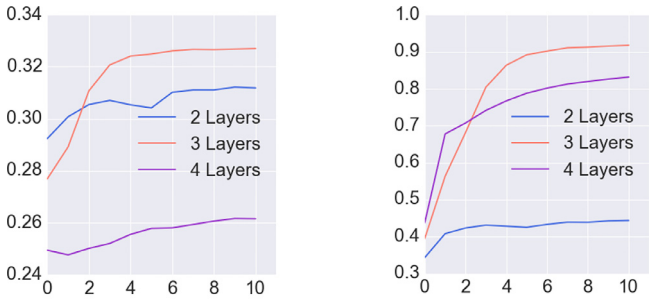
Fig. 3 illustrates this observation on the SynL3 network(*described in detail in* Section 5.2), which has 3 planted layers. It shows the average modularity scores after the Identification stage and during the first 10 iterations of the Refinement stage when we find 2, 3, and 4 layers on this network. The increase in modularity is sharpest at $n_L = 3$.

4.3.2. Stopping condition

Our rule for determining the number of layers n_L is motivated by this observation: if one selects the appropriate number of layers, the output will generally be of a higher quality.

We begin by setting the number of layers $n_L = 2$, and increase the number of layers until a stopping condition is met. For each candidate number of layers n_L, *HICODE* first calculates the modularity of the weakest layer obtained at the Identification stage. If the value is very low, then there are no more significant layers, so we set $n_L = n_L - 1$ and return; otherwise, let Q^t be the average modularity of all the detected layers at step t:

[2] Note that ReduceEdge is also suitable for weighted graph where e_{kk} corresponds to the sum of edge weights inside C_k, and d_k corresponds to the sum of degrees of nodes in C_k, weighted by the edge weight.



(a) Original Network (b) Reduced Network

Fig. 3. Change in average modularity when detecting 2, 3, or 4 layers on SynL3.

1. Calculate Q^0 for $t = 0$, i.e. after identification, before any refinement is conducted;
2. Perform $T = 10$ tentative iterations of refinement, and calculate Q^t for each $t \in \{1, \cdots, T\}$;
3. Calculate the average improvement ratio of modularity per iteration.

$$R_T = \frac{\sum_{t=1}^{T} Q_t}{T \cdot Q_0}$$

R_T represents how much refinement improves the detected layers. Instead of only considering how much improvement we get at step T, we use the average modularity of the T steps to balance the fluctuation on real-world networks. If n_L is too high, then when we remove the structure of the extra layers, we will be removing structure that actually belongs to some earlier layer. This will result in a lower quality partition, such that the refinement stage actually *lowers* the quality of the detected layers. Thus, we choose n_L corresponding to the peak R_T. The details are as shown in Algorithm 2.

Algorithm 2 Determine the number of layers.

Require: graph $G = (V, E)$,
base algorithm $\mathcal{A}_{base}$,
reduce method $\mathcal{M}_{Reduce}$,
tentative iterations $T = 10$ as the default, maximum number of layers $L_{max} = 8$ as the default.
Ensure: number of layers n_L

for $n_L = 2 : L_{max}$ **do**
  Run the identification stage of *HICODE* using $\mathcal{A}_{base}$ and $\mathcal{M}_{Reduce}$
  Calculate modularity Q^0 for all initial layers;
  Total modularity $Q^{sum} = Q^0$;
  for refinement iteration $t = 1 : T$ **do**
    Run the t-th refinement of *HICODE* using $\mathcal{A}_{base}$ and $\mathcal{M}_{Reduce}$;
    Calculate Q^t for all layers at the current iteration;
    $Q^{sum} = Q^{sum} + Q^t$;
  Average improvement ratio on modularity: $R_L = \frac{Q^{sum}}{T \cdot Q^0}$;
return n_L with the largest R_L;

5. Experimental setup

5.1. Evaluation metric

Besides *NMI*, we also define a Jaccard score-based metric to evaluate how well the detected layers resemble the ground truth communities. Given a set of detected communities $\mathcal{D}$ and a set of ground truth communities $\mathcal{G}$, the Jaccard similarity-based precision, recall, and F_1 score are defined as follows:

Each detected community D_i has its individual Jaccard Precision:

$$P(D_i) = \max_{G_j \in \mathcal{G}} \frac{|G_j \cap D_i|}{|G_j \cup D_i|}.$$

The *Jaccard Precision* P of the set $\mathcal{D}$ is defined as the weighted average of $P(D_i)$ over all detected communities, weighted by the size of the communities.

Table 1
Statistics for the synthetic datasets containing multiple layers of stochastic blockmodels. $|C|_{avg}$ denotes the average community size in different layers.

Dataset	SynL2_200	SynL2	SynL3
$\|V\|$	200	3000	3,000
$\|E\|$	960	14,446	21,510
#Layers	2	2	3
#Communities	5, 4	100, 50	100, 50, 30
$\|C\|_{avg}$	40, 50	30, 60	30, 60, 100
block probability	0.12, 0.10	0.16, 0.08	0.16, 0.08, 0.05

Table 2
The real-world network datasets.

Source	Domain	Dataset	**|V|**	**|E|**
Facebook	**Social**	Caltech	769	16,656
		Smith	2970	97,133
		Rice	4087	184,828
		Vassar	3068	119,161
		Wellesley	2970	94,899
		Bucknell	3826	158,864
		Carnegie	6637	249,967
		Ulllinois	30,809	1,264,428
SNAP	**Social**	YouTube	31,150	202,130
	Products	Amazon	13,288	41,730
	Collaboration	DBLP	49,097	170,284

Each ground truth community G_j has its individual Jaccard Recall:

$$R(G_j) = \max_{D_i \in \mathcal{D}} \frac{|G_j \cap D_i|}{|G_j \cup D_i|}.$$

The *Jaccard Recall* R of the set $\mathcal{G}$ is defined as the weighted average of $R(G_j)$ over all ground truth communities, weighted by the size of these communities.

We use weighted P and R to give a higher priority to bigger communities. The *Jaccard* F_1 *Score* is defined as the harmonic mean of P and R, i.e. $2PR/(P+R)$.

5.2. Synthetic stochastic blockmodel

We define a synthetic stochastic blockmodel containing multiple layers of planted communities. Each layer in a network corresponds to a single stochastic blockmodel that partitions the nodes into roughly equally-sized sets. In each layer, we first create the appropriate number of community IDs and randomly assign each node to a community, then we produce a $G(n, p)$ Erdos-Renyi random graph over each block. We select suitable p-values for the different layers so that they are of different densities, but roughly equal strengths as measured by modularity. By randomly assigning nodes to communities, we ensure that the communities in different layers are independent (i.e., a node's membership in different layers is unrelated).

Table 1 contains the parameters we have used for generating these synthetic datasets, and describes important structural characteristics. For instance, for the small network SynL2_200 that will be used to illustrate various details of our algorithm, there are two layers of communities, containing 5 communities of roughly size 40, and 4 communities of roughly size 50. The p values that govern the number of intra-community links for these two layers are 0.12 and 0.10. As shown in Table 3 (left column), their modularity scores are 0.40 and 0.39, and we observe a very low NMI of 0.05 between the two layers. We similarly generate larger synthetic networks SynL2 and SynL3, containing 2 and 3 layers, respectively. As with SynL2_200, the multiple layers have similar modularity scores, and are dissimilar from one another.

5.3. Real-world datasets

We do thorough testing on two groups of real-world networks, as summarized in Table 2.

(1) **Facebook networks**: These networks are portions of the Facebook social network for different universities in the United States [32]. For each university network, the data have categorical attributes encompassing the gender, major, year of matriculation, high school, dormitory, status (faculty, student, staff, etc.) and residence (house, dormitory, fraternity, etc.) of the users. We choose eight representative university networks which express comparatively high modularity scores when grouping the nodes by at least one of the attributes: Caltech, Smith, Rice, Vassar, Wellesley, Bucknell, Carnegie and Ulllinois.

Table 3
Statistics of the detected layers found by HC:Mod as compared with the annotated communities.

Datasets	Annotated communities				Community layers found by HC:Mod				
	Annotations (modularity, hiddenness value)	NMI_{max}	F_{max}	F_{avg}	#Layers	Modularity	NMI_{max}	F_{max}	F_{avg}
Synthetic									
SynL2_200	PL_1(0.40, 0.45) PL_2(0.39, 0.56)	0.05	0.04	0.04	2	0.40, 0.40	0.00	0.02	0.02
SynL2	PL_1(0.49, 0.50) PL_2(0.49, 0.50)	0.20	0.04	0.04	2	0.49, 0.49	0.00	0.04	0.04
SynL3	PL_1(0.33, 0.59) PL_2(0.32, 0.70) PL_3(0.32, 0.71)	0.20	0.04	0.03	3	0.33, 0.32, 0.32	0.00	0.04	0.03
Facebook									
Caltech	Dorm(0.30, 0.08) Year(0.19, 0.84)								
	Status(0.08, 0.85)	0.13	0.32	0.18	2	0.40, 0.25	0.00	0.18	0.18
Smith	Dorm(0.23, 0.42) Year(0.23, 0.49)	0.00	0.04	0.04	2	0.51, 0.34	0.03	0.11	0.11
Rice	Dorm(0.37, 0.02) Year(0.23, 0.94)								
	Status(0.13, 0.91)	0.11	0.26	0.14	3	0.50, 0.36, 0.34	0.02	0.20	0.13
Vassar	Year(0.34, 0.06) Dorm(0.15, 0.98)								
	Status(0.13, 0.89)	0.18	0.30	0.18	3	0.45, 0.32, 0.31	0.04	0.21	0.15
Bucknell	Year(0.40, 0.05) Status(0.12, 0.91)								
	Dorm(0.11, 0.98)	0.15	0.31	0.17	3	0.51, 0.37, 0.31	0.05	0.30	0.22
Carnegie	Year(0.28, 0.29) Major(0.12, 0.84)								
	Status(0.11, 0.78) Dorm(0.08, 0.92)	0.07	0.24	0.09	4	0.40, 0.42, 0.34, 0.32	0.06	0.23	0.17
Wellesley	Year(0.30, 0.10) Status(0.15, 0.79)	0.15	0.28	0.28	2	0.37, 0.26	0.00	0.15	0.15
UIllinois	Year(0.27, 0.24) High_school(0.15, 0.82)								
	Dorm(0.14, 0.76)	0.00	0.07	0.03	2	0.45, 0.34	0.04	0.16	0.14
SNAP									
YouTube	Co-liked $CommSet_A$(0.23, 0.24)								
	Co-liked $CommSet_B$(0.10, 0.73)	0.03	0.15	0.15	3	0.59, 0.48, 0.33	0.04	0.33	0.23
Amazon	Co-purchased $CommSet_A$(0.99, 0.29)								
	Co-purchased $CommSet_B$(0.47, 0.92)	0.69	0.72	0.72	2	0.99, 0.14	0.55	0.71	0.70
DBLP	Co-authorship $CommSet_A$(0.79, 0.13)								
	Co-authorship $CommSet_B$(0.03, 0.59)	0.53	0.10	0.10	3	0.90, 0.72, 0.56	0.01	0.10	0.09

(2) **SNAP networks**: We choose three networks with ground-truth communities collected by SNAP[3] [33]: Youtube, Amazon and DBLP. Youtube indicates friendships among users and the ground truth communities are user-defined groups. Amazon is a product co-purchasing network where co-purchased products are connected by edges, and the ground truth corresponds to product categories. DBLP is a co-authorship network and the ground truth corresponds to authors in the same conferences. For each network, we extract a subnetwork (*For convenience, we still use the same name*) containing all nodes in the top 5000 ground-truth. For further analysis, we partition the ground-truth communities into several community sets (*CommSet*) based on their hiddenness values.

More statistics for their ground truth communities are shown in Table 3 (left column): modularity and average hiddenness value for each ground truth community layer, as well as the maximum *NMI* (NMI_{max}), maximum and average F_1 scores (F_{max} and F_{avg}) for pairwise layers.

For Facebook networks, each layer gives rise to a set of communities grouped by a common attribute (i.e., nodes with a common annotation are in the same community), and we call each such set of communities an annotated layer (e.g., the 'Dorm' annotation gives rise to one annotated layer). These annotated layers cover all nodes (*coverage* = 100%) in the corresponding networks.

For SNAP networks, we manually divide the ground-truth communities into two sets, namely $CommSet_A$ or $CommSet_B$, depending on whether the hiddenness value is less than 0.5 or not. The coverage ratio (*ratio of nodes covered by the communities*) of $CommSet_A$ in Amazon, DBLP and Youtube are 100%%, 100%% and 80%%. So $CommSet_A$ forms a layer for Amazon and DBLP respectively, but it only covers 80%% of the nodes in Youtube. The coverage ratio of $CommSet_B$ on the three networks are 46%%, 4%% and 46%% respectively. $CommSet_A$ and $CommSet_B$ contain very different communities for DBLP and Youtube. But on Amazon, $CommSet_A$ and $CommSet_B$ contain very similar communities since their pairwise *NMI* and F_1 are very high, indicating that if an algorithm accurately detects most communities in $CommSet_A$, then this algorithm also has a high detection accuracy on $CommSet_B$.

5.4. Base algorithms and baselines

We compare *HICODE* to four popular algorithms in two categories:

(1) **Overlapping detection methods**: OSLOM (OS) [6] and Link Communities (LC) [5]. OS uses a fitness function and joins together small clusters into statistically significant larger clusters. LC is a landmark algorithm that finds communities by performing hierarchical clustering on the links, which results in overlapping communities of nodes.

[3] http://snap.stanford.edu.

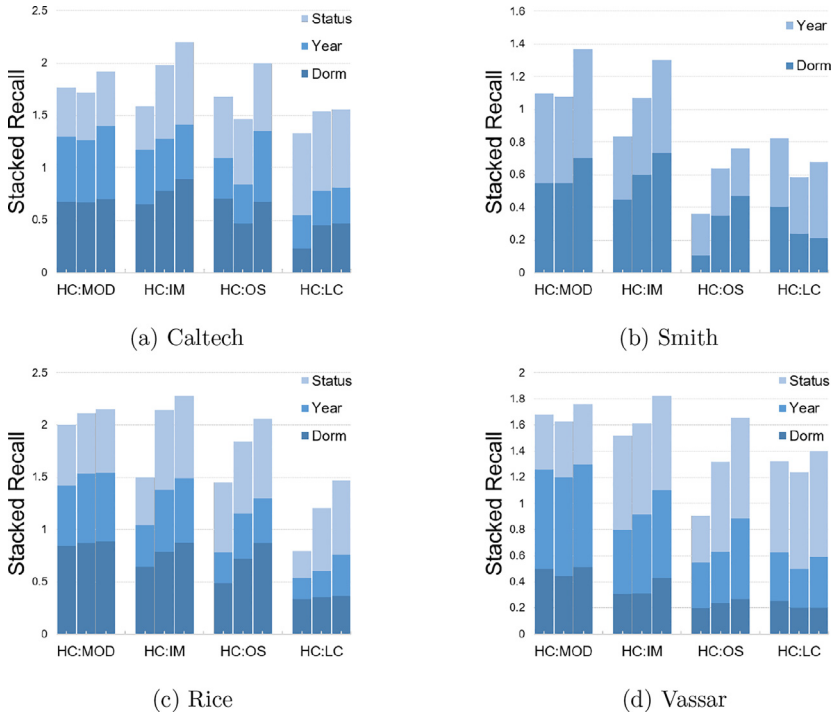


Fig. 4. Stacked Recall of all Layers for the reducing methods (left to right: RemoveEdge, ReduceEdge, Reduce Weight).

(2) **Disjoint detection methods**: Infomap (IM) [3] and Louvain method (Mod) [4]. IM is based on the random walk technique and minimizes the expected length of a description of information. Mod is a fast method popular for greedy modularity optimization.

We also implement *HICODE* using the four algorithms as the base, denoted as HC:Mod, HC:IM, HC:OS, and HC:LC.[4]

6. Experimental results

We first evaluate different variants of *HICODE* on different reducing methods and different base algorithms, and show that ReduceWeight performs the best in general, HC:Mod and HC:IM show similar performance and they outperform HC:OS, HC:LC shows lowest accuracy among the four versions of *HICODE*. We then illustrate the necessity of the Refinement stage that the accuracy improves among the iterations. In Sections 6.2 and 6.3, we show the statistics of the multiple layers detected by *HICODE* and compare the detection accuracy with state-of-the-art baselines on real-world networks. Experiments show that *HICODE* finds multiple significant, non-redundant community layers. These community layers are of high quality when evaluated mathematically by the popular modularity metric. Many of the layers are strongly associated with annotated communities grouped by living residence, year of registration or career position. The weaker layers are masked by the stronger layers and they are rarely uncovered by existing community detection algorithms.

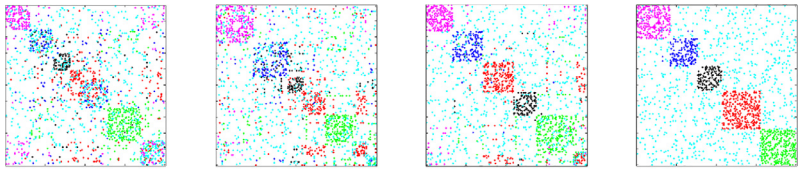
6.1. Analysis on HICODE

6.1.1. Comparison of the reduction methods

In this section, we first evaluate different reducing methods on each of the versions of *HICODE* (corresponding to different base algorithms). Fig. 4 shows the Recall for all layers of communities detected by HC:Mod, HC:IM, HC:OS and HC:LC on several examples of the Facebook networks: Caltech, Smith, Rice and Vassar.

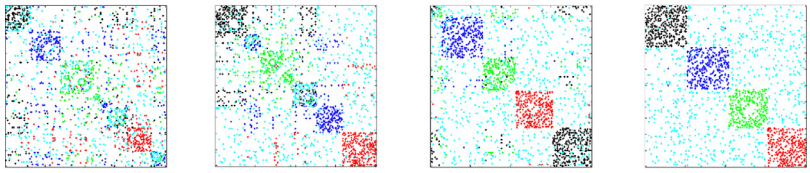
In general, ReduceWeight reaches the highest detection accuracy, followed by ReduceEdge and RemoveEdge. ReduceWeight and ReduceEdge provide a more accurate estimation on the background density, while RemoveEdge removes all intra-community edges and impairs more structure of other layers, especially when communities in different layers overlap considerably. ReduceWeight deterministically reduces the weight of intra-community edges and outperforms ReduceEdge, which randomly removes intra-community edges. To save space, we will present results using ReduceWeight.

[4] HICODE code and synthetic data: https://github.com/KunHe2015/HiCode/.



(a) $t = 0$ (b) $t = 5$ (c) $t = 10$ (d) $t = 20$

Fig. 5. Refinement of layer 1 on SynL2_200.



(a) $t = 0$ (b) $t = 5$ (c) $t = 10$ (d) $t = 20$

Fig. 6. Refinement of layer 2 on SynL2_200.

For the different versions of *HICODE*, HC:Mod and HC:IM perform consistently better than HC:OS and HC:LC. We will use HC:Mod to show the property of *HICODE*, and then compare our results on the four *HICODE* implementations with other algorithms.

6.1.2. Necessity of the refinement stage

To show the necessity of the Refinement stage, which gradually separates the community structures mixed together and strengthens the structure of each layer, we run HC:Mod on the small network SynL2_200 and illustrate its results.[5]

There are two planted layers on SynL2_200, with hiddenness values of 0.45 and 0.56. Figs. 5 and 6 illustrate several snapshots of the detected two layers during the execution (shown by the adjacency matrix but the node IDs are reordered for the two layers respectively). In the Identification stage ($t = 0$), *HICODE* only roughly identifies the stronger layer 1, and only roughly identifies the less strong layer 2 after the initial detected layer 1 is weakened. Then during the Refinement stage, by iteratively weakening the other community layers, we get a more accurate current layer, which forms a positive feedback cycle. The refinement process converges within 20 steps and the two detected layers remain stable in further iterations.

Fig. 7 illustrates the initial two layers of the communities that we detected immediately after the Identification stage as well as the final output community structure after refinement, where colors represent the planted community each node belongs to, and positions represent the detected communities. At each layer, each community is initially mixed with nodes from other communities, and the communities are purified during the Refinement stage.

In addition, we also observe that the average modularity scores of the detected layers on the original network are improved by the Refinement stage. As an example, Fig. 8 shows the increasing trend during the iteration when we detect 2, 3, 4 or 5 community layers on four real-world networks.

6.2. Evaluation on synthetic networks

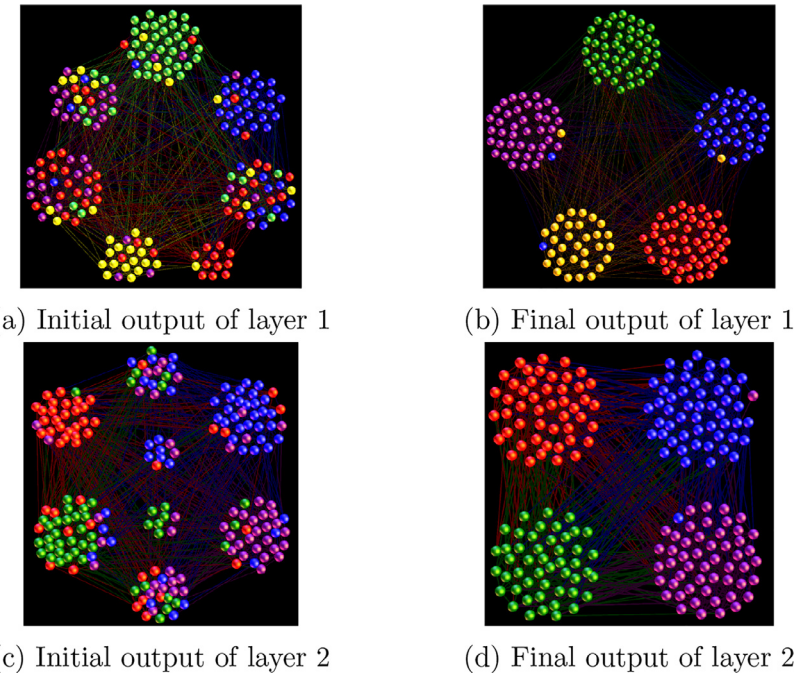
To demonstrate the ability of *HICODE* in finding multiple layers of community structures, we first examine the two synthetic networks SynL2 and SynL3, and find that *HICODE* substantially outperform the baseline algorithms, especially for networks with no less than 3 layers. For instance, Table 4 summarizes the results of evaluating how well each algorithm uncovers the planted layers on SynL3. *HICODE* finds almost all communities in the three layers, while the baseline algorithms mainly find some communities of the first layer and the second layer.

These experiments demonstrate that existing algorithms have difficulty locating communities outside the dominant layer, even if the hidden layers of communities have approximately the same modularity score compared with the dominant layer; however, by removing the effects of dominant community layers, *HICODE* is able to achieve much higher scores than competing methods.

6.3. Evaluation on real-world networks

For real-world networks from various domains, *HICODE* uncovers multiple layers of high modularity community structure. Table 3 (right column) shows the statistics of the layers found by HC:Mod in each network. When comparing these layers to

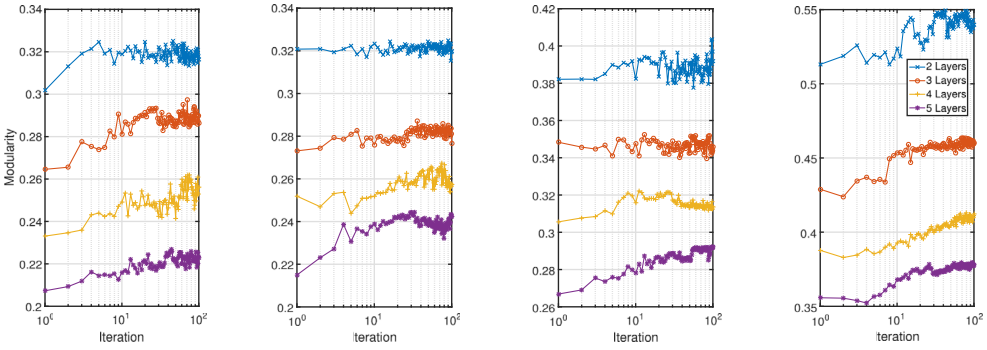
[5] The other base algorithms produce similar results.



(a) Initial output of layer 1 (b) Final output of layer 1

(c) Initial output of layer 2 (d) Final output of layer 2

Fig. 7. Initial and final output on SynL2_200.



(a) Caltech (b) Vassar (c) UIllinois (d) Youtube

Fig. 8. Improvement on Modularity of the detected layers.

one another, their maximum pairwise NMI, maximum and average pairwise F_1 scores are low, indicating that these layers of communities are distinct. One exception is Amazon, which seems only to contain one layer of very strong community structure.

When comparing the ground truth layers on the Facebook networks, we see the layers found by *HICODE* are strongly associated with the community categories. Tables 5 and 6 show the F_1 and *NMI* scores of *HICODE* and the baseline algorithms evaluated against different annotation categories on several examples: Caltech, Smith, Rice and Vassar. We see that:

(1) All baseline algorithms primarily locate the dominant structure, and rarely detect the hidden, weaker structures. Interestingly, different to other algorithms, LC regards Status and Year as the dominant on Vassar and Smith respectively. This occurs as LC may have a different notion of community structure.

(2) All *HICODE* implementations return community layers that are strongly associated with each ground truth category. *HICODE* not only uncovers the hidden layers that the baseline algorithms rarely detect, but also improves the detection accuracy on the dominant layer.

For each *CommSet* of the SNAP networks, we calculate the Recall of *AllLayers* (the union of all layers found by *HICODE*) and the Recall of the communities detected by each of the baselines. In Table 7, we see that the comparatively hidden $CommSet_B$ is consistently harder to be located than the dominant $CommSet_A$. When compared with the baselines, *HICODE* not only has higher detection accuracy on the comparatively hidden $CommSet_B$, but also boost the detection accuracy on the

Table 4
F_1 and *NMI* scores of all algorithms on SynL3 community categories. PL_1, PL_2, and PL_3 are the three layers of planted communities, with PL_1 containing small, dense communities and PL_3 containing larger, sparse communities.

		HC:MOD			HC:IM			HC:OS			HC:LC			Partitioning		Overlapping	
SynL3		L_1	L_2	L_3	L_1	L_2	L_3	L_1	L_2	L_3	L_1	L_2	L_3	Mod	IM	OS	LC
PL_1	F_1	**0.93**	0.04	0.03	**0.99**	0.03	0.03	0.05	0.04	**0.89**	**0.67**	0.06	0.07	0.44	0.88	0.58	0.56
HV = 0.59	*NMI*	**0.93**	0.00	0.00	**0.94**	0.00	0.00	0.01	0.00	**0.82**	**0.43**	0.00	0.00	0.14	0.72	0.50	0.19
PL_2	F_1	0.04	**0.99**	0.04	0.03	**0.98**	0.03	**0.94**	0.03	0.04	0.05	**0.41**	0.10	0.11	0.09	0.45	0.20
HV = 0.70	*NMI*	0.00	**0.96**	0.00	0.00	**0.93**	0.00	**0.87**	0.00	0.00	0.00	**0.11**	0.00	0.00	0.00	0.29	0.00
PL_3	F_1	0.03	0.04	**0.98**	0.03	0.03	**0.98**	0.04	**0.95**	0.04	0.05	0.07	**0.28**	0.12	0.08	0.25	0.12
HV = 0.71	*NMI*	0.00	0.00	**0.95**	0.00	0.00	**0.85**	0.00	**0.87**	0.00	0.00	0.00	**0.11**	0.00	0.00	0.06	0.00

Table 5
F_1 and *NMI* scores of all algorithms on Caltech, Smith community categories.

		HC:MOD		HC:IM			HC:OS		HC:LC		Partitioning		Overlapping	
Caltech		L_1	L_2	L_1	L_2	L_3	L_1	L_2	L_1	L_2	Mod	IM	OS	LC
Dorm	F_1	**0.58**	0.11	**0.65**	0.11	0.11	**0.48**	0.11	**0.21**	0.10	0.51	0.51	0.49	0.18
HV = 0.08	*NMI*	**0.39**	0.00	**0.48**	0.01	0.01	**0.32**	0.01	**0.16**	0.02	0.36	0.42	0.28	0.14
Year	F_1	0.11	**0.60**	0.12	**0.40**	0.20	0.14	**0.45**	0.07	**0.15**	0.13	0.14	0.12	0.07
HV = 0.84	*NMI*	0.00	**0.38**	0.03	**0.19**	0.09	0.00	**0.29**	0.02	**0.10**	0.05	0.13	0.00	0.03
Status	F_1	0.17	**0.37**	0.12	0.38	**0.64**	0.02	**0.36**	0.23	**0.51**	0.16	0.16	0.12	0.03
HV = 0.85	*NMI*	**0.16**	**0.11**	0.15	0.14	**0.32**	0.00	**0.22**	0.12	**0.25**	0.06	0.30	0.19	0.13
Smith		L_1	L_2	L_1	L_2	–	L_1	L_2	L_1	L_2	Mod	IM	OS	LC
Dorm	F_1	**0.45**	0.04	**0.50**	0.04	–	**0.40**	0.07	0.05	0.01	0.25	0.43	0.38	0.04
HV = 0.42	*NMI*	**0.26**	0.00	**0.36**	0.00	–	**0.25**	0.01	0.01	0.00	0.14	0.31	0.23	0.00
Year	F_1	0.12	**0.56**	0.10	**0.35**	–	0.15	**0.24**	0.01	**0.20**	0.21	0.18	0.16	0.18
HV = 0.49	*NMI*	0.00	**0.37**	0.03	**0.16**	–	0.05	**0.11**	0.00	**0.03**	0.06	0.06	0.06	0.00

Table 6
F_1 and *NMI* scores of all algorithms on Vassar, Rice community categories.

		HC:MOD			HC:IM			HC:OS			HC:LC			Partitioning		Overlapping	
Vassar		L_1	L_2	L_3	L_1	L_2	L_3	L_1	L_2	L_3	L_1	L_2	–	Mod	IM	OS	LC
Year	F_1	**0.67**	0.16	0.10	**0.54**	0.17	0.15	0.14	**0.44**	0.22	**0.17**	0.08	–	0.68	0.47	0.37	0.16
HV = 0.06	*NMI*	**0.47**	0.01	0.00	**0.31**	0.07	0.02	0.06	**0.19**	0.06	**0.04**	0.02	–	0.38	0.27	0.14	0.04
Dorm	F_1	0.13	**0.41**	0.08	0.11	0.09	**0.25**	0.11	**0.15**	0.11	0.07	**0.08**	–	0.12	0.12	0.16	0.08
HV = 0.98	*NMI*	0.03	**0.24**	0.02	0.02	0.01	**0.08**	0.01	**0.07**	0.02	0.02	**0.03**	–	0.00	0.02	0.00	0.00
Status	F_1	**0.34**	0.23	0.12	0.34	**0.57**	0.19	0.52	0.23	**0.61**	**0.63**	0.27	–	0.33	0.33	0.23	0.61
HV = 0.89	*NMI*	**0.15**	0.06	0.01	0.10	**0.21**	0.06	0.20	0.09	**0.27**	**0.04**	**0.08**	–	0.15	0.10	0.07	0.04
Rice		L_1	L_2	L_3	L_1	L_2	L_3	L_1	L_2	L_3	L_1	L_2	L_3	Mod	IM	OS	LC
Dorm	F_1	**0.79**	0.11	0.07	**0.74**	0.08	0.08	**0.61**	0.22	0.22	**0.20**	0.10	0.07	0.70	0.55	0.54	0.10
HV = 0.02	*NMI*	**0.71**	0.00	0.00	**0.45**	0.00	0.00	**0.50**	0.10	0.11	**0.10**	0.01	0.00	0.59	0.32	0.29	0.04
Year	F_1	0.08	0.22	**0.55**	0.08	0.20	**0.34**	0.09	**0.22**	0.20	0.07	0.14	**0.15**	0.07	0.08	0.14	0.05
HV = 0.94	*NMI*	0.00	0.07	**0.29**	0.00	0.06	**0.16**	0.00	**0.13**	0.10	0.01	0.01	**0.07**	0.05	0.05	0.00	0.00
Status	F_1	0.11	**0.42**	0.23	0.10	**0.61**	0.23	0.13	0.32	**0.58**	0.05	**0.61**	0.12	0.10	0.08	0.14	0.03
HV = 0.91	*NMI*	0.00	**0.20**	0.11	0.01	**0.25**	0.09	0.01	0.16	**0.42**	0.00	**0.05**	0.02	0.04	0.01	0.00	0.01

Table 7
Jaccard Recall *R* of all algorithms on SNAP data.

Youtube	**HC:MOD**	**HC:IM**	**HC:OS**	**HC:LC**	Mod	IM	OS	LC
$CommSet_A$	**0.16**	**0.27**	**0.30**	**0.28**	0.13	0.04	0.27	0.28
$CommSet_B$	**0.10**	**0.12**	**0.13**	**0.13**	0.05	0.03	0.08	0.12
Amazon	**HC:MOD**	**HC:IM**	**HC:OS**	**HC:LC**	Mod	IM	OS	LC
$CommSet_A$	**0.93**	**0.90**	**0.90**	**0.91**	0.89	0.88	0.78	0.83
$CommSet_B$	**0.90**	**0.86**	**0.89**	**0.88**	0.86	0.83	0.82	0.79
DBLP	**HC:MOD**	**HC:IM**	**HC:OS**	**HC:LC**	Mod	IM	OS	LC
$CommSet_A$	**0.23**	**0.21**	**0.29**	**0.22**	0.19	0.19	0.21	0.15
$CommSet_B$	**0.11**	**0.15**	**0.20**	**0.24**	0.09	0.09	0.14	0.18

comparatively dominant communities in $CommSet_A$. Again, on Amazon, all algorithms detect both $CommSet_A$ and $CommSet_B$ fairly well due to the highly similarity of the two sets of ground truth communities.

7. Conclusions

The hidden community structure concept presented in this paper is useful for understanding and uncovering relationships among communities in complex social networks. This work contributes to the working field and provides a systematic analysis method to uncover the hidden communities. Experimental results over various domains indicate that some hidden communities exist in real-world networks, and our proposed HICODE can detect them and performs favourably compared to existing methods, including overlapping community detection algorithms. As a meta-approach, HICODE is scalable by applying any conventional community detection algorithm as the base algorithm.

The framework proposed in this work sheds light on the organization of complex networks and provides new research methodology for community detection. In the future, we plan to explore the hidden communities on directed or weighted networks, apply other community detection algorithm as the base, as well as design new reduction method for weakening the detected communities. It is also possible to apply our ideas for other network mining tasks such as link-prediction or maximum influence propagation.

Acknowledgments

The authors are very grateful to the editors and reviewers for their valuable comments and suggestions. The work is supported by National Natural Science Foundation of China (61772219, 61472147, 61602196), US Army Research Office (W911NF-14-1-0477), Microsoft Research Asia Collaborative Research (No. 97354136).

References

[1] M. Girvan, M.E. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. 99 (12) (2002) 7821–7826.
[2] M.E. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2) (2004) 026113.
[3] M. Rosvall, C.T. Bergstrom, Maps of random walks on complex networks reveal community structure, Proc. Natl. Acad. Sci. 105 (4) (2008) 1118–1123.
[4] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech: Theory Exp. 2008 (10) (2008) P10008.
[5] Y.-Y. Ahn, J.P. Bagrow, S. Lehmann, Link communities reveal multiscale complexity in networks, arXiv:0903.3178 (2009).
[6] A. Lancichinetti, F. Radicchi, J.J. Ramasco, S. Fortunato, Finding statistically significant communities in networks, PLoS ONE 6 (4) (2011) e18961.
[7] S. Zhang, R.-S. Wang, X.-S. Zhang, Identification of overlapping community structure in complex networks using fuzzy c-means clustering, Physica A 374 (1) (2007) 483–490.
[8] K. He, Y. Sun, D. Bindel, J. Hopcroft, Y. Li, Detecting overlapping communities from local spectral subspaces, in: Data Mining (ICDM), 2015 IEEE International Conference on, IEEE, 2015, pp. 769–774.
[9] D. Park, R. Singh, M. Baym, C.-S. Liao, B. Berger, Isobase: a database of functionally related proteins across ppi networks, Nucleic Acids Res. 39 (suppl 1) (2010) D295–D300.
[10] J. Das, H. Yu, Hint: high-quality protein interactomes and their applications in understanding human disease, BMC Syst. Biol. 6 (1) (2012) 92.
[11] M. Coscia, F. Giannotti, D. Pedreschi, A classification for community discovery methods in complex networks, Statis. Anal. Data Min. 4 (5) (2011) 512–546.
[12] J. Xie, S. Kelley, B.K. Szymanski, Overlapping community detection in networks: the state-of-the-art and comparative study, ACM Comput. Surv. (csur) 45 (4) (2013) 43.
[13] P. Pons, M. Latapy, Computing communities in large networks using random walks, J. Graph Algorithms Appl. 10 (2) (2006) 191–218.
[14] K. Nowicki, T.A.B. Snijders, Estimation and prediction for stochastic blockstructures, J. Am. Stat. Assoc. 96 (455) (2001) 1077–1087.
[15] B. Karrer, M.E. Newman, Stochastic blockmodels and community structure in networks, Phys. Rev. E 83 (1) (2011) 016107.
[16] G. Palla, I. Derenyi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature 435 (7043) (2005) 814–818.
[17] J.J. Whang, D.F. Gleich, I.S. Dhillon, Overlapping community detection using seed set expansion, in: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, ACM, 2013, pp. 2099–2108.
[18] M. Coscia, G. Rossetti, F. Giannotti, D. Pedreschi, Demon: a local-first discovery method for overlapping communities, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2012, pp. 615–623.
[19] Y. Cui, X.Z. Fern, J.G. Dy, Non-redundant multi-view clustering via orthogonalization, in: Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on, IEEE, 2007, pp. 133–142.
[20] D. Niu, J.G. Dy, M.I. Jordan, Multiple non-redundant spectral clustering views, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 831–838.
[21] D. Niu, J.G. Dy, M.I. Jordan, Iterative discovery of multiple alternativeclustering views, IEEE Trans. Pattern Anal. Mach. Intell. 36 (7) (2014) 1340–1353.
[22] J. Yang, J. McAuley, J. Leskovec, Community detection in networks with node attributes, in: Data Mining (ICDM), 2013 IEEE 13th International Conference on, IEEE, 2013, pp. 1151–1156.
[23] X. Huang, H. Cheng, J.X. Yu, Dense community detection in multi-valued attributed networks, Inf. Sci. (Ny) 314 (2015) 77–99.
[24] P.-Y. Chen, A.O. Hero, Deep community detection, IEEE Trans. Signal Process. 63 (21) (2015) 5706–5719.
[25] J.-G. Young, A. Allard, L. Hébert-Dufresne, L.J. Dubé, A shadowing problem in the detection of overlapping communities: lifting the resolution limit through a cascading procedure, PLoS ONE 10 (10) (2015) e0140133.
[26] K. He, S. Soundarajan, X. Cao, J. Hopcroft, M. Huang, Revealing multiple layers of hidden community structure in networks, arXiv:1501.05700 (2015).
[27] A. Mislove, B. Viswanath, K.P. Gummadi, P. Druschel, You are who you know: inferring user profiles in online social networks, in: Proceedings of the Third ACM International Conference on Web Search and Data Mining, ACM, 2010, pp. 251–260.
[28] S.-H. Teng, et al., Scalable algorithms for data and network analysis, Found. Trends® Theor. Comput. Sci. 12 (1–2) (2016) 1–274.
[29] L. Danon, A. Diaz-Guilera, J. Duch, A. Arenas, Comparing community structure identification, J. Stat. Mech: Theory Exp. 2005 (09) (2005) P09008.
[30] A.F. McDaid, D. Greene, N. Hurley, Normalized mutual information to evaluate overlapping community finding algorithms, arXiv:1110.2515 (2011).
[31] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[32] A.L. Traud, P.J. Mucha, M.A. Porter, Social structure of facebook networks, Physica A 391 (16) (2012) 4165–4180.
[33] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, Knowl. Inf. Syst. 42 (1) (2015) 181–213.