FACH: Fast Algorithm for Detecting Cohesive Hierarchies of Communities in Large Networks

Mojtaba Rezvani
Australian National University
Canberra, ACT 2601, Australia
mojtaba.rezvani@anu.edu.au

Qing Wang
Australian National University
Canberra, ACT 2601, Australia
qing.wang@anu.edu.au

Weifa Liang
Australian National University
Canberra, ACT 2601, Australia
wliang@cs.anu.edu.au

ABSTRACT

Vertices in a real-world social network can be grouped into densely connected communities that are sparsely connected to other groups. Moreover, these communities can be partitioned into successively more cohesive communities. Despite an ever-growing pile of research on hierarchical community detection, existing methods suffer from either inefficiency or inappropriate modeling. Yet, some cut-based approaches have shown to be effective in finding communities without hierarchies. In this paper, we study the hierarchical community detection problem in large networks and show that it is NP-hard. We then propose an efficient algorithm based on edge-cuts to identify the hierarchy of communities. Since communities at lower levels of the hierarchy are denser than the higher levels, we leverage a fast network sparsification technique to enhance the running time of the algorithm. We further propose a randomized approximation algorithm for information centrality of networks. We finally evaluate the performance of the proposed algorithms by conducting extensive experiments using real datasets. Our experimental results show that the proposed algorithms are promising and outperform the state-of-the-art algorithms by several orders of magnitude.

KEYWORDS

Hierarchical community detection; large-scale networks

ACM Reference Format:
Mojtaba Rezvani, Qing Wang, and Weifa Liang. 2018. FACH: Fast Algorithm for Detecting Cohesive Hierarchies of Communities in Large Networks. In *WSDM 2018: WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining , February 5–9, 2018, Marina Del Rey, CA, USA,* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.1145/3159652.3159704

1 INTRODUCTION

It is well-known that communities in a network often exhibit a hierarchical structure [9, 23, 27, 28]. For instance, metabolic networks of organisms can be decomposed into highly connected communities, where communities form a hierarchy in which communities at lower levels of the hierarchy are more cohesive, and vertices within those communities are closer to each other [23]. Researchers in a collaboration network can be grouped into communities based on their research areas, from general areas such as computer science to more specific ones such as database and data mining, where information circulates more quickly among them. Therefore, small and cohesive communities are nested into larger and less cohesive communities in a hierarchical manner. Despite the importance of the hierarchical community detection, existing approaches have their limitations. For example, they fail to reveal a cohesive hierarchical structure among communities, or they suffer from inefficiency in large-scale networks. In order to capture cohesiveness, existing approaches for flat community detection, such as k-truss [5] and k-core [3], can be adjusted to detect a hierarchical structure among communities. For instance, one can identify a hierarchy of communities by starting with $k = 1$ and increasing the value of k to obtain more cohesive communities at the lower levels of the hierarchy. Fig. 1 shows a hierarchical structure of communities detected in a real-world network Amazon, where k-core and k-truss both fail to detect the hierarchical structure of communities. It can be seen that for $k < 4$, the whole network is in a single k-core community, while k-truss is unable to detect any community for $k > 4$ or $k < 3$.

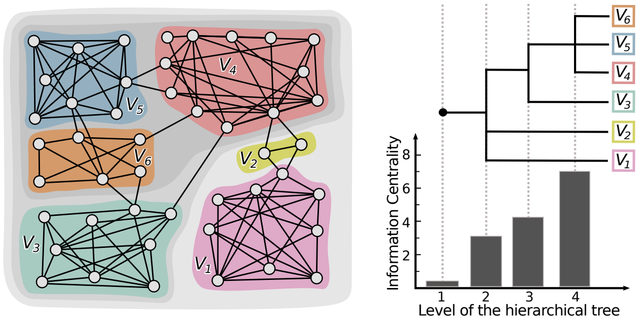


Figure 1: A hierarchical structure of communities in Amazon, where from the root to its leaves connections within communities become denser and the values of their information centrality increase.

Traditionally, hierarchical community detection methods find hierarchies by removing/merging edges and vertices of a network one by one, which can be very time consuming. For example, Girvan and Newman [11, 20] suggested starting with a given network as a community and partitioning the network by removing edges in decreasing order of their betweenness centrality. Similarly, Fortunato et al. [10] advocated to remove edges in order of impact of their removal on the information centrality (mean distance) of a network. Newman [19] proposed an approach, where communities are merged if the modularity of the resulting community can be

WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA

ACM ISBN 978-1-4503-5581-0/18/02…$15.00
https://doi.org/10.1145/3159652.3159704

increased, starting from vertices. However, repeatedly calculating modularity, difference in modularity, betweenness and information centralities in a large network is computationally infeasible.

To address the aforementioned issues, we study the problem of hierarchical community detection based on the intuitive cohesiveness among communities in a hierarchy, where communities at lower levels of a cohesive hierarchy are more densely connected with each other. We define the notion of *cohesive hierarchy* that is a rooted tree of communities, where communities at the k-th level of the hierarchy are connected to each other through weak cuts with no larger than k edges. For example in Fig. 1, communities at the second level (V_1, V_2 and $V_3 \cup \ldots \cup V_6$) are connected to each other by two edges, and communities at the third level (V_3 and $V_4 \cup V_5 \cup V_6$) are connected to each other by 3 edges. As we move towards the leaves of this hierarchy, the connectivity becomes stronger and communities become more densely connected.

We propose an efficient, yet scalable cut-based algorithm for detecting a cohesive hierarchy in a large-scale network. However, finding cuts in a network with hundreds of millions of edges is a painstaking task. Particularly at high levels of a hierarchy, when the network has not been broken into smaller subgraphs, finding cuts in the network is a challenge. Based upon this intuition, we optimize the cut detection algorithm by sparsifying the network in early iterations. Specifically, when finding communities at the k-th level of a cohesive hierarchy, we remove unnecessary edges that are not part of any edge-cut with size k. Therefore, when detecting communities at top levels of the hierarchy, the number of edges is significantly reduced, i.e. at most $\min\{m, (k+1)(n-1)\}$ at level k, where n and m are the number of edges and vertices.

One of the key ingredients of the proposed cohesive hierarchical community structure is the information centrality of communities, which is increased from communities at one level to communities at a lower level of the hierarchy. However, the information centrality can be calculated in time $O(nm)$, which is unrealistic in real-world networks. Moreover, the diameter in real-world networks is usually small, as they exhibit a small-world characteristic. Therefore, we propose a randomized approximation algorithm for calculating the information centrality whose error is no larger than a fraction ϵ of the network diameter, with high probability (at least $1 - \frac{1}{n}$), while it runs in time $\Theta(m \log n/\epsilon^2)$, where n and m are the number of vertices and the number of edges, respectively.

Our contributions in this paper are as follows,

- We first formalize the problem of cohesive hierarchical community detection, where the granularity of a community is measured using information centrality, and prove the NP-hardness of this problem.
- We then develop an efficient algorithm for the hierarchical community detection problem and enhance the algorithm by incorporating a fast sparsification for efficiently finding less granular levels of the hierarchy.
- We devise a randomized algorithm for approximating the information centrality of a network with a guaranteed approximation ratio.
- We finally evaluate the effectiveness and efficiency of our proposed algorithms by quantitatively analysing their performance using five real-world datasets.

The rest of this paper is organized as follows. Section 2 introduces preliminaries and Section 3 introduces the problem definition. Section 4 presents two novel algorithms for the hierarchical community detection problem and Section 5 presents an approximation algorithm for information centrality. Section 6 discusses the experimental results, which compare the performance of our proposed algorithms against the benchmark algorithms. Section 7 provides a literature review, and Section 8 concludes the paper.

2 PRELIMINARIES

A network can be modeled as an undirected connected graph $G = (V, E)$, where V is the set of vertices representing individuals and E is the set of edges representing relationships between individuals. Let $n = |V|$ and $m = |E|$. The *degree* of a vertex v is the number of edges incident to it, denoted by $deg(v)$. Two paths in G are called *edge-disjoint* if they do not share any edges. The number of edge-disjoint paths between two vertices u and v is the *edge-connectivity* between them, denoted by $\lambda(u, v)$. Two vertices u and v are said to be *k-edge-connected* if $\lambda(u, v) \geq k$. An *edge-cut* between two subsets V_1 and V_2 in G, denoted as $E[V_1, V_2]$, is the set of edges in G such that their removal will disconnect V_1 and V_2. For brevity, we simply write $E[V_1]$, whenever $V_1 = V_2$. Given a subset V' of V, $G[V'] = (V', E[V'])$ is the induced subgraph of G by V'.

Traditionally, communities are perceived as subsets of vertices of a graph G that the number of edges among them (density of connections) is large. Following this perception, it is possible to find hierarchical communities by recursively increasing the density threshold, and consequently finding denser communities. In this paper, we take the relationships among communities into account for finding hierarchical communities of a network, while previous studies only focused on the relationships among vertices. Specifically, we here represent the hierarchy of communities as a rooted tree of subsets of vertices in G. Given two partitions $P = \{V_1, \cdots, V_{|P|}\}$ and $P' = \{V'_1, \cdots, V_{|P'|}\}$ of V , we say that P has a higher *hierarchical order* than P', denoted by $P \succ P'$, if for every set $V'_i \in P'$ there is a strict superset $V_j \in P$ that includes V'_i, i.e. $V'_i \subset V_j$. However, the density of connections among communities is not uniform across different levels of a cohesive hierarchy. In fact, density of connections among communities becomes larger at lower levels of a cohesive hierarchy. Therefore, we say that P has a higher *hierarchical order at degree k* compared to P', denoted by $P \succ_k P'$, if for every set $V'_i \in P'$ there is a set $V_j \in P$ such that $V'_i \subset V_j$ and for every set $V'_j \in P'$, $E[V'_i, V'_j] \leq k$. A sequence of partitions of V, e.g. $\mathcal{P} = \langle P_1, \cdots, P_{|P|}\rangle$ is said to be a *hierarchy*, if $P_{i-1} \succ P_i$ $(1 < i \leq |\mathcal{P}|)$. We refer to $\mathcal{P}' = \langle P'_1, \cdots, P'_{|\mathcal{P}'|}\rangle$ as a *cohesive hierarchy* if $P'_{i-1} \succ_i P'_i$ $(1 < i \leq |\mathcal{P}'|)$. We note that every cohesive hierarchy is a hierarchy, but a hierarchy is not necessarily cohesive. Given a hierarchy $\mathcal{P}$, we refer to $P_{|\mathcal{P}|}$ the lowest level of hierarchy and we refer to P_1 as the root of the hierarchy. For every partition $P_i \in \mathcal{P}$, we say that $P_j \in \mathcal{P}$ is at a lower level if $j > i$.

Example 2.1. Let us consider the network illustrated in Fig. 1, where V is the set of vertices of the network and V_i is the set of vertices in a subgraph G_i $(1 \leq i \leq 6)$. We show that $\mathcal{P} = \langle P_1, P_2, P_3, P_4\rangle$ is a cohesive hierarchy, where $P_1 = \{V\}$, $P_2 = \{V_1, V_2, V_3 \cup V_4 \cup V_5 \cup V_6\}$, $P_3 = \{V_1, V_2, V_3, V_4 \cup V_5 \cup V_6\}$, and $P_4 = \{V_1, V_2, V_3, V_4, V_5, V_6\}$. It can be seen that $P_1 \succ_2 P_2$, since vertices in V_1, V_2 and $V_3 \cup V_4 \cup V_5 \cup V_6$

are connected to each other by at most 2 edges. Furthermore, $P_2 \succ_3 P_3$, since V_3 is connected to $V_4 \cup V_5 \cup V_6$ by at most 3 edges. Also $P_3 \succ_4 P_4$, as V_4, V_5 and V_6 are connected to their siblings by at most 4 edges, which means that $P_4 = \{V_1, V_2, V_3, V_4, V_5, V_6\}$. As a result, $\mathcal{P} = \langle P_1, P_2, P_3, P_4 \rangle$ is a cohesive hierarchy. It is also noted that a cohesive hierarchy such as $\mathcal{P}$ results in a decomposition tree that is illustrated in Fig. 2, where siblings at level k of the decomposition tree are connected to each other by at most k edges.

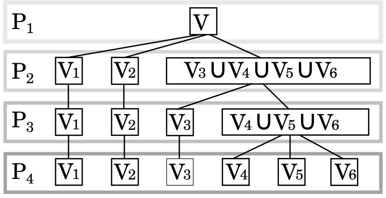


Figure 2: The cohesive hierarchy of the network in Fig. 1

The *distance* between two vertices u and v in a graph G, denoted by d^G_{uv}, is the length of the shortest path between them. We have $d^G_{vv} = 0$ for any vertex $v \in V$. The *diameter* of G is the length of the longest shortest path between any pair of vertices in G, denoted by Δ. The *information centrality* of G, denoted by $\mathcal{D}(G)$, is the inverse of mean distance between every pair of vertices u and v [10], i.e.,

$$\mathcal{D}(G) = \frac{n(n-1)}{\sum_{v \in V} \sum_{u \in V} d^G_{uv}}. \quad (1)$$

The measure in Eq. (1) has been widely adopted for modeling information centrality in many networks [10, 24].

3 PROBLEM DEFINITION

In this section, we formally define the problem of hierarchical community detection, based on information centrality.

Definition 3.1 (HCDP). Given a network G, the *hierarchical community detection problem* is to find a cohesive hierarchy $\mathcal{P}$ of communities, where the sum of information centralities of the subgraphs induced by all partitions in $\mathcal{P}$ is maximized, i.e.

$$\text{maximize} \sum_{P_i \in \mathcal{P}} \sum_{V' \in P_i \setminus P_{i-1}} \mathcal{D}\left(G[V']\right). \quad (2)$$

The flat community detection is a special case of this hierarchical community detection problem, when $|\mathcal{P}| = 1$. In the following, we show that the hierarchical community detection problem is NP-hard by a reduction from the maximum clique problem to its special case, i.e. the flat community detection problem. We first define the decision version of the flat community detection problem.

Definition 3.2 (CDP Decision). Given a network $G = (V, E)$, a rational number $\epsilon > 0$ and a positive integer k, the *community detection decision problem* is to determine whether there is a subset V' of vertices with size k, whose information centrality is no less than ϵ, i.e. $\mathcal{D}\left(G[V']\right) \geq \epsilon$.

The following lemma shows that the CDP Decision problem is NP-hard, using a reduction from the MaximumClique problem.

Lemma 3.3. *CDP Decision problem is NP-Complete.*

Proof. We first show that CDP Decision is in NP. Given a certificate of CDP Decision, which consists of a network G, and a set of vertices $V' \subseteq V$ with $|V'| = k$, we can use all-pairs shortest paths algorithm [6] to determine if $\mathcal{D}(G[V']) \geq \epsilon$, in polynomial time. Thus, CDP Decision is in NP.

We now show that CDP Decision is NP-hard by a reduction from the MaximumClique problem. Note that a subset of vertices V' form a clique in G, if and only if the information centrality of the induced subgraph $G[V']$ is 1, i.e. $\mathcal{D}(G[V']) = 1$. Therefore, it is implied that the MaximumClique problem is a special case of CDP Decision, where $\epsilon = 1$. Thus, given a network G and a positive integer k, one can decide the existence of a clique of the given size k by solving the CDP Decision. □

4 COHESIVE HIERARCHICAL COMMUNITY DETECTION

In this section, we propose two efficient, yet scalable algorithms for the cohesive hierarchical community detection problem: (i) CHD which is a basic cut-based algorithm to iteratively partition the network into densely connected communities, and (ii) FACH which is an optimized cut-based algorithm that relies on a sparsification technique to find sparse communities at high levels of a hierarchy.

4.1 The basic algorithm (CHD)

The CHD algorithm proceeds iteratively to identify a cohesive hierarchy $\mathcal{P}$ from a given network G. In each iteration, it finds one level of the hierarchy by creating a partition of each subset of vertices at its parent level. In iteration k, the algorithm finds a partition P_k by decomposing subsets of vertices in P_{k-1} into several subsets, where k is 1 initially, and incremented in each iteration.

Let $\mathcal{P}$ be a detected cohesive hierarchy of communities in network G, which is initialized by $\emptyset$ and let P_k be the k-th level of hierarchy $\mathcal{P}$. We assume that P_0 consists of all vertices as a community. Let k be the size of cuts detected by CHD in iteration k, which is initialized to 1 in the first iteration. In iteration k of the algorithm, for every set of vertices V' in P_{k-1}, which is currently a leaf in $\mathcal{P}$, an induced subgraph $G' = G[V']$ is constructed. Then a multi-cut of size no larger than k is detected in G' using the following procedure, called MAS-Decompose, and the result of removing this cut from G' is stored in another subgraph G''. The procedure MAS-Decompose [2] decomposes the subgraph G' into several connected components in G'', such that each connected component in G'' is connected to other vertices by at most k edges. The CHD algorithm then calculates the information centrality of G' and G'', and if the information centrality of G'' is no less than that of the initial subgraph G', it adds vertices in connected components of G'' to the k-th level of the hierarchy, i.e. P_k. Otherwise, the initial set of vertices V' is added to P_k. The CHD algorithm increments k and constructs levels of the hierarchy, until for all $V' \in P_{k-1}$, the algorithm cannot find a multi-cut of size k in G' that can increase the information centrality of G''. The detailed description of steps is given by Algorithm 1.

The rest is to show the time complexity of Algorithm 1 as follows.

Theorem 4.1. *Given a network $G = (V, E)$, the algorithm* CHD *delivers a feasible solution for the hierarchical community detection problem in time $O(n^2 m + n^3 \log(n))$.*

Algorithm 1 CHD(G)

```
Input: $G = (V, E)$
Output: A cohesive hierarchy $\mathcal{P}$
$\mathcal{P} \leftarrow \emptyset$; /* Initialize the cohesive hierarchy */
$k \leftarrow 0$;
$P_k \leftarrow \{V\}$; /* $P_k$ is the value of $k$-th level of the hierarchy */
$P_{k-1} \leftarrow \emptyset$;
while $(\mathcal{D}(P_k) - \mathcal{D}(P_{k-1}) > 0)$ do
    $k \leftarrow k + 1$; /* Increment the value of $k$ for this iteration */
    $P_k \leftarrow \emptyset$;
    for each $V' \in P_{k-1}$ do
        $G' \leftarrow G[V']$;
        $G'' \leftarrow$ MAS-Decompose$(G', k)$; /* $G''$ is a graph created by removing the edges in the multi-cut of size $k$ found by MAS from $G'$ */
        if $\mathcal{D}(G') \le \mathcal{D}(G'')$ then
            Add connected components of $G''$ to $P_k$;
        else
            $P_k \leftarrow P_k \cup \{V'\}$;
    $\mathcal{P} \leftarrow \{P_k\}$;
return $\mathcal{P}$;
```

Proof. The CHD algorithm proceeds iteratively. Within each iteration, for every subgraph $G[V']$ ($V' \in P_{k-1}$), a MAS-Decompose is applied in time $O(nm + n^2 \log(n))$, and the calculation of information centrality takes $O(nm)$ time.

Therefore, the time complexity of MAS-Decompose is dominant at each iteration of the CHD algorithm. It is also noted that the hierarchy $\mathcal{P}$ has at most n leaves, as the number of subgraphs is no larger than the number of vertices in the network (due to the Pigeonhole principle). Therefore, MAS-Decompose is called at most n times in each iteration of the CHD algorithm. However, in each iteration of the CHD algorithm, MAS-Decompose is run on partitions of the actual graph. Let n' and m' be the number of vertices and edges in the induced subgraph by each leaf V', i.e. $G[V']$. It is then implied that $\sum_{V' \in P_{k-1}} n'm' \le nm$ as $\sum_{V' \in P_{k-1}} n' \le n$ and $\sum_{V' \in P_{k-1}} m' \le m$. This means that the overall time complexity of each iteration is no larger than $O(nm + n^2 \log(n))$. Since the edge-connectivity of a network is at most $n - 1$, the time complexity of the CHD algorithm is $O(n^2 m + n^3 \log(n))$. □

4.2 The FACH algorithm

We here propose an algorithm, called FACH, that can reduce the number of edges from m in iteration k to $\min\{m, (k + 1)(n - 1)\}$, where n and m are the number of vertices and edges in a network.

In the following, we describe the construction of a sparse network $G_i = (V, E_i)$ $(1 \le i \le m)$ from $G = (V, E)$ which is noticeably smaller than G, while it can be used to find minimum cuts of G.

Lemma 4.2. *[18] Given a network $G = (V, E)$ and an integer $k > 0$, let $F_1 = (V, E_1)$ be a spanning forest in G and $F_i = (V, E_i)$ be a spanning forest in $G \setminus E_1 \cup E_2 \cdots E_{i-1}$ $(1 < i \le k - 1)$. For any two vertices $s \in V$ and $t \in V$, if they are k-edge-connected in G, then they must be k-edge-connected in $G_k = F_1 \cup \cdots \cup F_k$.*

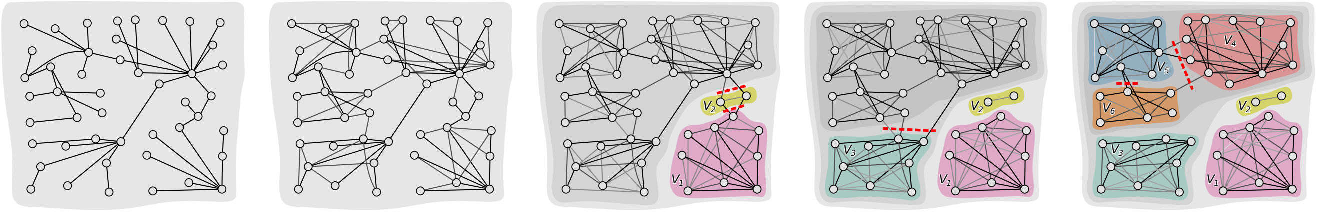
Lemma 4.2 states that in the second iteration, where $k = 2$, we can find a cut with size k in a network with at most $3(n - 1)$ edges, instead of the entire network with m edges. Similarly, in iteration k of the algorithm, a cut of size k can be found in a network with $\min((k + 1)(n - 1), m)$ edges.

Motivated by Lemma 4.2, we propose the FACH algorithm that runs significantly faster than the CHD algorithm by starting from a sparse network and gradually increasing the network size as a factor of k. Specifically, in iteration k of this algorithm, the MAS-Decompose is run on a subgraph of the initial network with at most $\min\{(k + 1)(n - 1), m\}$ edges. In other words, the network size in the first iteration is no larger than $2(n - 1)$. In the second iteration, the network size is no larger than $3(n - 1)$ and in iteration k, the network size is no larger than $(k + 1)(n - 1)$. Let G_k be the network in iteration k, on which the MAS-Decompose is run. The FACH algorithm starts with $G_0 = MSF(G)$, where $MSF(G)$ is the minimum spanning forest of the network G. The algorithm then constructs G_k by adding the minimum spanning forest of the residual network $G \setminus G_{k-1}$ to G_{k-1}.

Let $\mathcal{P}$ be a cohesive hierarchy, which is initially empty. The algorithm iteratively increments the value of k and finds P_k, the partition representing the k-th level of the cohesive hierarchy $\mathcal{P}$, until the information centrality of P_k cannot be increased. Let $P_0 = \{V\}$. In iteration k, the FACH algorithm finds a minimum spanning forest F_k and form $G \setminus G_{k-1}$, where G_{k-1} is the union of spanning forests found in iterations 1 to $k - 1$. For each community $V' \in P_{k-1}$, the FACH algorithm creates an induced subgraph $G_k[V']$ and finds a multi-cut in it using the MAS-Decompose procedure. If the removal of the multi-cut can increase the information centrality, the FACH algorithm applies the cut; otherwise, it proceeds to the next community in P_{k-1}. The detailed steps of the FACH algorithm is given in Algorithm 2.

Algorithm 2 FACH(G)

```
Input: $G = (V, E)$
Output: A cohesive hierarchy $\mathcal{P}$
$\mathcal{P} \leftarrow \emptyset$; /* Initialize the cohesive hierarchy */
$k \leftarrow 0$;
$G_k \leftarrow MSF(G)$; /* Find a Minimum Spanning Forest of $G$ */
$P_k \leftarrow \{V\}$; /* $P_k$ is the $k$-th level of the hierarchy */
$P_{k-1} \leftarrow \emptyset$;
while $(\mathcal{D}(P_k) - \mathcal{D}(P_{k-1}) > 0)$ do
    $k \leftarrow k + 1$; /* Increment the value of $k$ for this iteration */
    /* Find a Minimum Spanning Forest of $G \setminus G_{k-1}$ */
    $F_k \leftarrow MSF(G \setminus G_{k-1})$;
    $G_k \leftarrow F_k \cup G_{k-1}$;
    $P_k \leftarrow \emptyset$;
    for each $V' \in P_{k-1}$ do
        $G' \leftarrow G_k[V']$;
        $G'' \leftarrow$ MAS-Decompose$(G', k)$;
        if $\mathcal{D}(G') \le \mathcal{D}(G'')$ then
            Add connected components of $G''$ to $P_k$;
        else
            $P_k \leftarrow P_k \cup \{V'\}$;
    $\mathcal{P} \leftarrow \{P_k\}$;
return $\mathcal{P}$;
```



(a) $k = 0$ **(b)** $k = 1$ **(c)** $k = 2$ **(d)** $k = 3$ **(e)** $k = 4$

Figure 3: A running example of the FACH **algorithm with the sparsification technique.**

Example 4.3. Fig. 3 illustrates a running example of the FACH algorithm, in which network is sparsified in early iterations for finding cohesive communities. In Fig. 3b, it can be seen that in the first iteration, the edge-cuts are found in a network that has only $81 = 43 + 38$ edges, which is sparser than the original network that has 124 edges. The number of edges is increased to $105 = 77 + 28$ and $113 = 102 + 11$ in the third and fourth iterations, respectively.

We now provide the time complexity of the FACH algorithm.

Theorem 4.4. *Given a network $G = (V, E)$, there is an algorithm for the hierarchical community detection problem, i.e. the* FACH *algorithm, which delivers a feasible solution in time $O(nm)$.*

Proof. The FACH algorithm consists of several iterations, where in iteration k, the union of spanning forests for levels 1 to $k-1$, i.e. G_i is constructed and for every induced subgraph $G_i[V']$ ($V' \in P_{k-1}$), a MAS-Decompose is applied, and the value of the information centrality is calculated in $O(nm)$ time.

The time complexity of the MAS-Decompose is $O(nm + n^2)$ and the time complexity of information centrality estimation is $O(nm)$. Since the network is connected $m \geq n - 1$, the time complexity of the FACH algorithm is $O(nm)$. □

5 APPROXIMATING INFORMATION CENTRALITY

To efficiently detect a cohesive hierarchy of communities in a large-scale network, one challenge is to calculate the value of information centrality. One straightforward way to calculate the information centrality is to discover all-pairs shortest paths, which is very time-consuming or even infeasible for real-world networks. Thus, we devise a simple, yet scalable algorithm for approximating the information centrality in polylogarithmic time.

We here build on top of the results obtained by Eppstein et al. [8] and devise a randomized algorithm, Algorithm 3, that finds the information centrality of a network using a small number of randomly selected vertices from the network. The algorithm first selects a set S ($|S| = s$) of vertices from V, uniformly at random. It then runs the single-source shortest path algorithm (BFS), starting from each randomly selected vertex in S. Finally, it estimates the value of information centrality of the network using the following equation,

$$\hat{\mathcal{D}}(G) = \frac{s \cdot (n-1)}{\sum\limits_{u \in S} \sum\limits_{v \in V} d_{uv}^G}. \tag{3}$$

Algorithm 3 describes an overal view of the approximation algorithm for information centrality in a given network.

Algorithm 3 Estimated information centrality

```
Input: $G = (V, E)$, $s$
Output: $\hat{\mathcal{D}}(G)$
  Let $S$ be a set of $s$ vertices selected uniformly at random;
  for each vertex $u \in S$ do
      Calculate shortest paths $d_{uv}^G$ from $u$ to all $v \in V$;
  for each vertex $v \in V$ do
      $\hat{C}(v) \leftarrow \sum_{u \in S} d_{uv}^G$;
  $\hat{\mathcal{D}}(G) \leftarrow s \cdot (n-1) / \sum_{v \in V} \hat{C}(v)$;
  return $\hat{\mathcal{D}}(G)$;
```

5.1 Theoretical analysis

We now analyse the approximation algorithm for information centrality. We first review the Hoeffding lemma [14] as follows.

Lemma 5.1 (Hoeffding [14]). *If $x_1, x_2, \cdots, x_n$ are independent variables, such that variable x_i ($1 \leq i \leq n$) is bounded by a_i and b_i, and $\mu = E[\sum x_i/n]$ is the expected mean, then for $\xi > 0$,*

$$\Pr\left\{ \left| \frac{\sum_{i=1}^n x_i}{n} - \mu \right| \geq \xi \right\} \leq 2e^{-2n^2\xi^2 / \sum_{i=1}^n (b_i - a_i)^2}. \tag{4}$$

Due to the small-world characteristic of complex networks [30], the diameter in such networks is usually small. Therefore, we show that the error of Algorithm 3 for approximating the information centrality of networks is no larger than $\epsilon\Delta$ with a high probability, where Δ is the diameter of network and ϵ is a given approximation ratio.

Theorem 5.2. *Given a network $G = (V, E)$, and a set S of randomly selected vertices with size $\Theta(\log(n)/\epsilon^2)$, Algorithm 3 approximates the reciprocal of information centrality such that $|1/\hat{\mathcal{D}}(G) - 1/\mathcal{D}(G)| \leq \epsilon\Delta$, with high probability of at least $1 - 1/n$.*

Proof. Recall that Algorithm 3 chooses $s = |S|$ random samples from vertices and the approximated value of information centrality is

$$\hat{\mathcal{D}}(G) = \frac{s \cdot (n-1)}{\sum\limits_{u \in S} \sum\limits_{v \in V} d_{uv}^G} = \frac{s \cdot n \cdot (n-1)}{n \cdot \sum\limits_{u \in S} \sum\limits_{v \in V} d_{uv}^G}. \tag{5}$$

It is noted that the expected reciprocal of estimation, i.e. $1/\hat{\mathcal{D}}(G)$, is the reciprocal of actual information centrality, i.e. $1/\mathcal{D}(G)$. Thus, in Hoeffding lemma, considering $x_i = \frac{n \sum_{v \in V} d_{iv}^G}{n(n-1)}$, we can safely assume that $\mu = 1/\mathcal{D}(G)$, $a_i = 0$ and $b_i = n\Delta/(n-1)$. Therefore, the probability that the difference between the reciprocal estimated information centrality $1/\hat{\mathcal{D}}(G)$ and the actual reciprocal information

Table 1: Real datasets with their characteristics.

Dataset	# vertices	# edges	# communities	Δ
Facebook	4,039	88,234	308	8
Amazon	334,863	925,872	14,529	44
DBLP	317,080	1,049,866	7,556	21
LiveJournal	3,997,962	34,681,189	12,115	17
Orkut	3,072,441	117,185,083	9,120	9

centrality $1/\mathcal{D}(G)$ being more than ξ is

$$\Pr\left\{\left|\frac{1}{\hat{\mathcal{D}}(G)} - \frac{1}{\mathcal{D}(G)}\right| \geq \xi\right\} \quad \leq \quad 2e^{-2s^2\xi^2/s(\frac{n\Delta}{n-1})^2}.$$

Considering the error being a small fraction of the diameter of G, i.e. $\xi = \epsilon\Delta \ll \Delta$, and using $s = \frac{\log n}{\epsilon^2}$ random samples, it can be seen that the probability of error is bounded above by $1/n$. Therefore, the approximation error of Algorithm 3 is smaller than $\epsilon\Delta$ with a high probability of at least $(n-1)/n$. □

The following theorem shows that the time complexity of the FACH algorithm with the proposed randomized approximation algorithm can be reduced to $O(n^2)$.

Theorem 5.3. *Given a network $G = (V, E)$, where $m = cn$ with constant c, the* FACH *algorithm delivers a feasible solution for the hierarchical community detection problem (HCDP), in time $O(n^2)$.*

Proof. The FACH algorithm proceeds iteratively, where in iteration k, the union of spanning forests for levels 1 to $k-1$, i.e. G_i is constructed and for every induced subgraph $G_i[V']$ ($V' \in P_{k-1}$), a MAS-Decompose is applied, and the value of the information centrality is calculated, using Algorithm 3.

As mentioned above, the time complexity of the MAS-Decompose is $O(nm+n^2)$ and the time complexity of information centrality estimation is $O(m \log n + n \log^2 n)$. It can be seen that MAS-Decompose dominates the time-complexities of each iteration of the FACH algorithm. In iteration k, the network size is $k(n-1)$. Since the number of edges in real-world social networks is usually a constant factor of the number of vertices, i.e. $m = cn$, for a constant c, the hierarchies $\mathcal{P}$ will have at most a constant number $k \leq c$ of levels. Therefore, the overall time complexity of the FACH algorithm is $O(n^2)$. □

6 EXPERIMENTAL RESULTS

In this section, we discuss the performance of the proposed algorithms, i.e. CHD and FACH, on several real datasets by comparing against several state-of-the-art algorithms. We first describe the experimental settings and then evaluate the performance of the proposed algorithms in detecting the hierarchical structure of communities. We finally investigate the performance of the CHD and FACH algorithms in finding different levels of a hierarchy.

6.1 Experimental settings

We introduce the benchmark algorithms, datasets and measures that were adopted for evaluating the proposed algorithms.

Benchmark algorithms. We compare the performance of the proposed algorithms, i.e. `CHD` and `FACH`, with the following state-of-the-art algorithms for hierarchical community detection: `LinkCluster` [1], `CNM` [4], `InfoMap` [26], and `OSLOM` [16].

Datasets. We used five real datasets that are publicly available[1], and have been widely used in the literature [32]: (1) Facebook is a subgraph of the social network facebook, where communities are groups of members identified by surveyed users, (2) Amazon is a network in which vertices are products and there is an edge between two vertices i and j if product i is frequently co-purchased with product j. Products in each category are considered as ground-truth communities, (3) DBLP is a collaboration network of researchers, where communities are defined as journals and conferences, (4) LiveJournal is a friendship network of users in the LiveJournal website. Users can create groups, and these groups are considered as the ground-truth communities. (5) Orkut is the friendship network of Orkut members. Communities in this network are groups created by users, where users can join each group.

Evaluation measures. Measuring the quality of detected communities is challenging, as different metrics lead to different interpretations of communities. We employ F-measure that is widely-adopted in the literature [12, 31–34] for quantifying the accuracy of detected communities. Let $\mathcal{C}^*$ be the set of ground-truth communities and let C be a detected community. The F-measure of C compared to $C^* \in \mathcal{C}^*$ is defined as follows,

$$F_k(C) = \max_{C^* \in \mathcal{C}^*} \{\frac{(k+1) \cdot p(C, C^*) \cdot r(C, C^*)}{k \cdot p(C, C^*) + r(C, C^*)}\}, \tag{6}$$

where $p(C, C^*) = |C \cap C^*|/|C|$ and $r(C, C^*) = |C \cap C^*|/|C^*|$ are the precision and recall, respectively. To calculate the accuracy of a flat community detection algorithm, one may calculate the average of F_1 and F_2-measures for all detected communities [12, 31–34]. However, the situation is different for hierarchical community detection algorithms, as communities detected at each level of a hierarchy have different characteristics and can be interpreted differently. One general rule in hierarchical community detection is that communities at the lower levels are smaller, more connected and more cohesive than the ones at the higher levels. Therefore, we suggest a weighting method in calculating the F-measure of communities at different levels of a hierarchy, which provides us with the ability to put more weight on communities at lower levels. Specifically, we incorporate a weight α_i, called the weight of level i, into F-measure of communities at level i of a hierarchy. Given a detected hierarchy $\mathcal{P} = \{P_1, \cdots, P_{|\mathcal{P}|}\}$, we define the F-measure of $\mathcal{P}$ as follows,

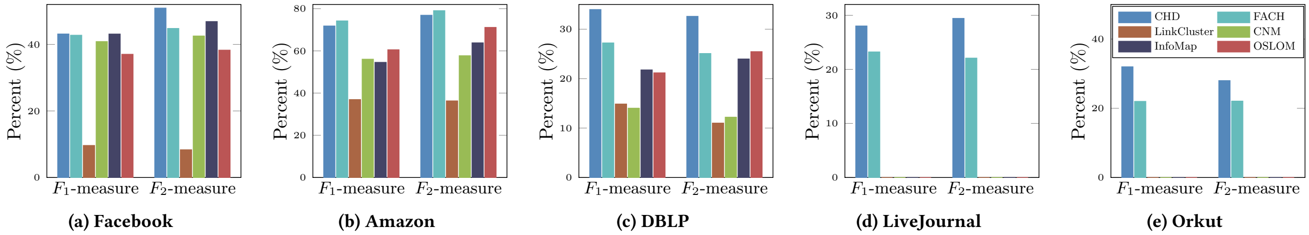
$$F_k(\mathcal{P}) = \sum_{1 \leq i \leq |\mathcal{P}|} \frac{1}{|\mathcal{P}|} \sum_{C \in P_i} \alpha_i \frac{F_k(C)}{|P_i|}, \tag{7}$$

$$\text{where } \alpha_i = \frac{i}{\sum_{1 \leq j \leq |\mathcal{P}|} j}.$$

Notice that the term α_i is called the weight of level i, which is used to emphasize on the accuracy of communities at the lower levels of the hierarchy. Although there are an infinite number of ways to define the weights α_i, we use a simple intuitive definition that makes both empirical and analytical sense.

All our experiments were run on a desktop computer with an Intel(R) Core(TM) i7-3770 CPU (3.40GHz) and 32GB of RAM.

[1] http://snap.stanford.edu/data/index.html



(a) Facebook (b) Amazon (c) DBLP (d) LiveJournal (e) Orkut

Figure 4: F_1 and F_2-measures of the hierarchical community detection algorithms.

6.2 Accuracy and efficiency

We study the accuracy and efficiency of our proposed algorithms CHD and FACH in hierarchical community detection by calculating the F-measures of communities found in each network and comparing them against the benchmark algorithms.

Fig. 4 shows the accuracy of different hierarchical community detection algorithms in terms of F_1 and F_2-measures. It can be seen in Fig. 4a that algorithms CHD and FACH outperform the benchmark algorithms by nearly 1% in F_1-measure for dataset Facebook. Fig. 4a also shows that algorithm CHD outperforms all other algorithms by at least 5% in F_2-measure using dataset Facebook. In Fig. 4b, it can be seen that the F_1 measures of algorithms CHD and FACH are at least 10% higher than the other algorithms in the benchmark using dataset Amazon. Similarly, F_2-measures for both CHD and FACH in this dataset are at least 5% higher than all other algorithms for dataset Amazon. Fig 4c plots the results for dataset DBLP, where it can be observed that both F_1 and F_2-measures of algorithm CHD is at least 20% higher than all other benchmark algorithms. However, the accuracy of algorithm FACH is slightly less than that of algorithm CHD for dataset DBLP. Fig. 4d presents the results for dataset LiveJournal and Fig. 4e presents the results for dataset Orkut, where due to the large size of the datasets only algorithms CHD and FACH terminated. It is noted that we waited for 150 hours for all algorithms to terminate, but only CHD and FACH found the results. Yet, the accuracy of the results is reasonable (above 20% for both datasets LiveJournal and Orkut). It is noted in Fig. 4 that the performance of algorithm CHD is better than that of algorithm FACH in datasets LiveJournal and Orkut. The reason is that algorithm CHD spends more time in finding edge-cuts in the networks.

Fig. 5 depicts the running times of different hierarchical community detection algorithms. Although dataset Facebook is small, Fig. 5 shows that the running time of algorithms CHD and FACH is only a fraction of all other benchmark algorithms. In Fig. 5, for dataset Amazon, the running time of algorithm CHD is at most 1/50 of that of all benchmark algorithms, and the running time of algorithm FACH is only 80% of the running time of algorithm CHD. Similarly, for dataset DBLP, the running time of algorithm CHD is at most 1/75 of the running time of benchmark algorithms, and the running time of algorithm FACH is only 60% of the running time of algorithm CHD. However, for datasets LiveJournal and Orkut, only algorithms CHD and F̂ACH terminated within 150 hours, and their running time was less than 30 minutes for such a large datasets. Fig. 5 illustrates that the running time of algorithm FACH is less than that of algorithm CHD, because of the sparsification technique. However, for dataset Orkut, both algorithms CHD and FACH have similar running times, because these algorithms discovers more levels of the hierarchy.

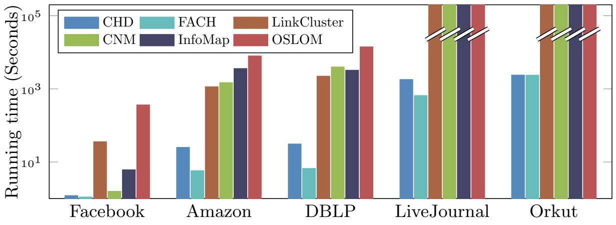
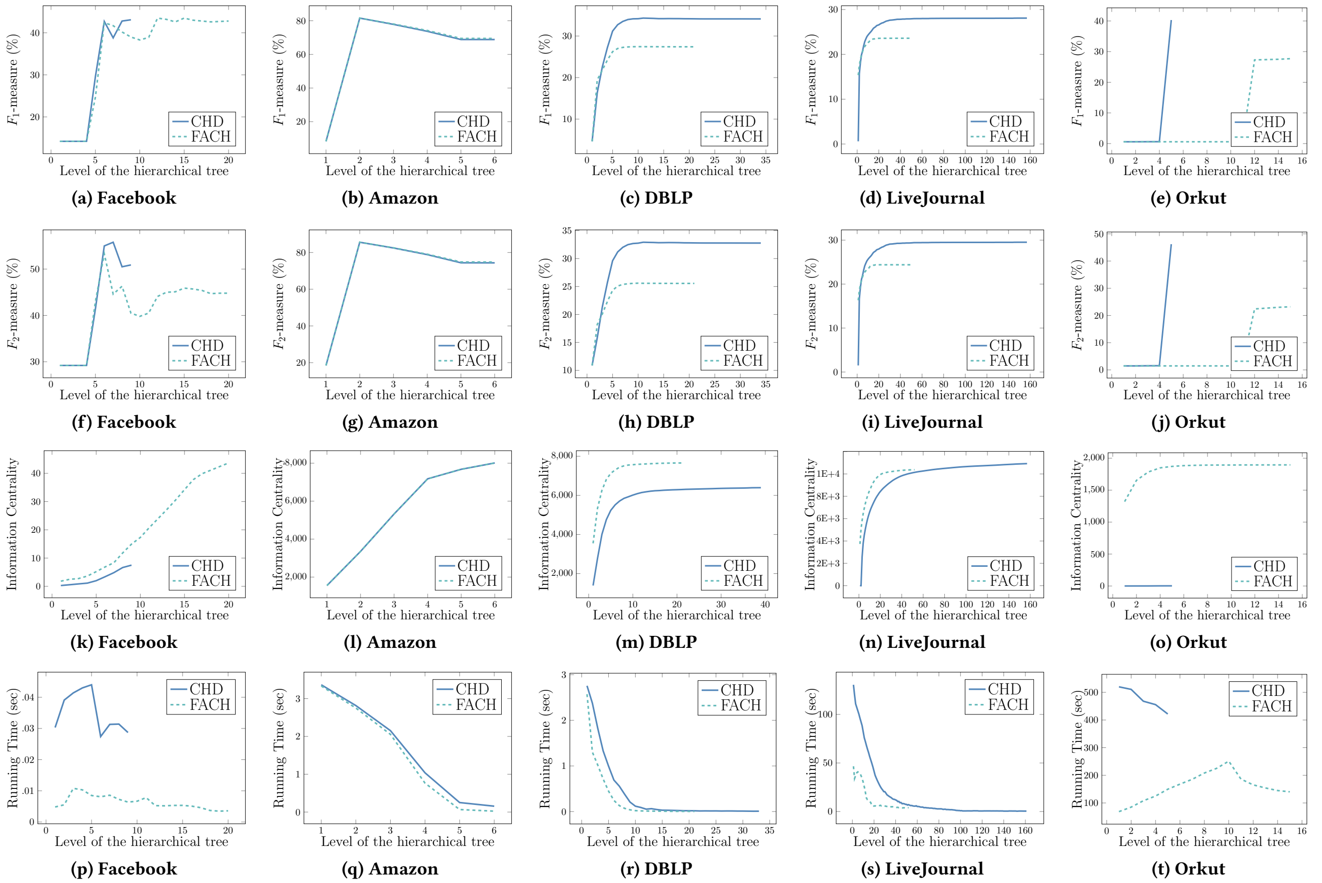


Figure 5: Running times of hierarchical community detection algorithms, where bars with parallel lines indicate that the corresponding algorithm did not terminate in 150 hours.

6.3 Hierarchies: level by level

Fig. 6 illustrates the values of F_1-measure, F_2-measure, information centrality and running times for the communities at each level of hierarchies detected by algorithms CHD and FACH. Fig. 6a shows that for dataset facebook, the value of F_1-measure increases as we move towards the lower levels of a hierarchy using algorithms CHD and FACH, with minor fluctuations. However, it can be seen in Fig. 6f that the value of $F2$-measure has a sudden drop at level 5 of the hierarchy, which indicates that the communities beyond level 5 of the hierarchy are too fine. In Fig. 6b, the values of F_1-measure of communities detected by algorithm CHD for dataset Amazon are increasing, as we move towards lower levels of hierarchy of communities. While there are some fluctuations in F_1-measure for dataset Amazon, the values are stable at the last two levels of hierarchy, i.e. levels 5 and 6. It is noted that the number of levels of the hierarchy for dataset Amazon is 6. Similarly, Fig. 6c shows that for dataset DBLP, the values of F_1-measure and F_2-measure have an increasing trend as we move towards the bottom of a hierarchy.

Fig. 6k-6o present the values of information centrality for communities at different levels of a hierarchy. Fig. 6k shows that for dataset facebook, the value of information centrality by algorithm CHD has a sudden increase and it stabilizes after that. However, in the same figure, the value of information centrality has a slower increase using algorithm FACH. It is noted in Fig. 6l that as the algorithm iterates, it finds more levels of the hierarchy and increases the values of the information centrality of communities, for dataset Amazon. Similarly for dataset DBLP, Fig. 6m shows that the information centrality of the detected communities is strictly increasing. In Fig. 6n, for dataset LiveJournal, the value of information centrality is increasing, as we move downwards in the hierarchy.



(a) Facebook (b) Amazon (c) DBLP (d) LiveJournal (e) Orkut

(f) Facebook (g) Amazon (h) DBLP (i) LiveJournal (j) Orkut

(k) Facebook (l) Amazon (m) DBLP (n) LiveJournal (o) Orkut

(p) Facebook (q) Amazon (r) DBLP (s) LiveJournal (t) Orkut

Figure 6: Running times for each level of hierarchies detected by algorithms CHD and FACH

Fig. 6p-6t plot the amounts of time that algorithms CHD and FACH need for detecting each level of a hierarchy in different datasets. Fig. 6p shows that the running time of algorithm CHD is mainly dominated by its early iterations, where k is small. However, the running time of the algorithm FACH is almost stable with minor fluctuations across different levels of the hierarchy, which is due to the fast sparsification in algorithm FACH. It can be seen in Fig. 6q that for dataset Amazon, the running time of algorithm CHD drops significantly, as the algorithm moves to lower levels of the hierarchy. The reason behind is that algorithm CHD breaks the network into smaller and smaller subgraphs at each level and therefore, the time spent for detecting lower levels of a hierarchy is much less than that of the higher one. Fig. 6r shows the running times of algorithm CHD for different levels of the hierarchy in dataset DBLP, from which it can be seen that the running time significantly drops. Similarly, Fig. 6s shows that for dataset LiveJournal, the running time drops, as we move downwards the hierarchy of communities. However algorithm FACH spends significantly less amount of time in both early and late iterations, resulting in a faster outcome. In Fig. 6t, while the running time of algorithm CHD is dominated by its early iterations, algorithm FACH performs very fast in those early iterations. The conclusion from Fig. 6p-6t is that algorithm CHD is promising in hierarchical community detection, but algorithm FACH is much more efficient, particularly in early iterations.

7 RELATED WORKS

Existing methods on hierarchical community detection can be categorized into two categories: *top-down* and *bottom-up* [28] approaches. While top-down approaches start with a given network and partition it into denser communities, bottom-up ones start with seeds, and expand those seeds gradually by merging them with each other. We here review the most influential works in each category.

Top-down approaches. Newman et al. [11, 20] proposed partitioning a network into communities by removing the edges in the order of their betweenness centralities. In order to reduce the running time of betweenness centrality calculations in [11, 20], Radicchi et al. [22] proposed to remove edges that disconnect the network, thereby, reducing the network size for the next rounds of betweenness centrality calculations. Radicchi et al. [22] also suggested replacing the betweenness centrality with the edge-clustering coefficient, which can be calculated faster. However, calculating the edge-coefficient clustering value is still very time consuming to be calculated repeatedly in a large network. Similarly, Fortunato et al. [10] used the information centrality of a network as a measure for removing edges at each iteration. Fortunato et al. [10] proposed to remove the edges, whose removal can result in the maximum decrease in the information centrality of network, where calculating the information centrality in a large-scale network is impractical.

Bottom-up approaches. Newman [19] proposed a bottom-up approach for hierarchical community detection, where communities are merged, if the modularity of the resulting community can be increased, starting from vertices as seeds. In an attempt to improve the efficiency of this algorithm [19], Clauset et al. [4] proposed several optimization techniques, which is yet unable to scale to networks with millions of vertices. Expansion of every single vertex in a network can be very time consuming and it can result in many redundant communities, therefore in LFM [15] only a randomly selected set of vertices are expanded until the value of modularity is locally maximal. Similarly, Sales-Pardo et al [27] considered a set of local maxima communities according to modularity fitness metric. However, the modularity used in [15], [19] and [27] does not allow single vertices to be a community. Haveman et al. [13] modified the fitness function to allow a single vertex to be a community. Many researchers [7, 17, 21, 29] have attempted to use small subsets of vertices as seeds. For example, in the works EAGLE [29] and COCD [7] it was proposed to start with small cliques as seeds and merge two communities with the maximum similarity into one, where the similarity between two communities is proportional to the number of edges between them. However, cliques are a very strict condition for real-world communities. Lancichinetti et al. [16] presented the algorithm OSLOM, which uses an estimated statistical significance of a community as a fitness metric and locally optimizes the statistical significance of communities. Rosvall et al. [26] proposed a method called Informap, which is a generalization of their flow-based clustering method [25] to uncover hierarchical communities. Ahn et al. [1], proposed a different approach called link clustering, where edges are partitioned via hierarchical clustering.

8 CONCLUSIONS

In this paper, we studied the hierarchical community detection problem. We formally defined the problem of hierarchical community detection, as finding a rooted tree of communities where each community is a subset of its parent in the tree, and the information centrality of communities is no less than that of their parent in the hierarchical tree. We showed that the problem of finding hierarchical communities is NP-hard and devised an efficient and scalable heuristic algorithms for this problem. We further incorporated a fast sparsification method to reduce the network size for finding global cuts. We also proposed a fast randomized algorithm to estimate the value of information centrality in large-scale networks. We finally validate the effectiveness of our proposed algorithms using extensive experiments over five large-scale real-world datasets.

9 ACKNOWLEDGEMENT

This work is supported by the grant of Australian Research Council Discovery Project No. DP120102627.

REFERENCES

[1] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. 2010. Link communities reveal multiscale complexity in networks. *Nature* 466, 7307 (2010), 761–764.

[2] Lijun Chang, Jeffrey Xu Yu, Lu Qin, Xuemin Lin, Chengfei Liu, and Weifa Liang. 2013. Efficiently computing k-edge connected components via graph decomposition. In *SIGMOD'13*. 205–216.

[3] James Cheng, Yiping Ke, Shumo Chu, and M Tamer Özsu. 2011. Efficient core decomposition in massive networks. In *ICDE*. 51–62.

[4] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical review E* 70, 6 (2004), 066111.

[5] Jonathan Cohen. 2008. Trusses: Cohesive subgraphs for social network analysis. *National Security Agency Technical Report* (2008), 16.

[6] Thomas H.. Cormen, Charles Eric Leiserson, Ronald L Rivest, and Clifford Stein. 2001. *Introduction to algorithms*. Vol. 6. MIT press Cambridge.

[7] Nan Du, Bai Wang, and Bin Wu. 2008. Overlapping community structure detection in networks. In *CIKM'08*. 1371–1372.

[8] David Eppstein and Joseph Wang. 2001. Fast approximation of centrality. In *Proc. of SODA'01*. 228–229.

[9] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3 (2010), 75–174.

[10] Santo Fortunato, Vito Latora, and Massimo Marchiori. 2004. Method to find community structures based on information centrality. *Physical review E* 70, 5 (2004), 056104.

[11] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *PNAS'02* 99, 12 (2002), 7821–7826.

[12] Prem K Gopalan and David M Blei. 2013. Efficient discovery of overlapping communities in massive networks. *PNAS'13* 110, 36 (2013), 14534–14539.

[13] Frank Havemann, Michael Heinz, Alexander Struck, and Jochen Gläser. 2011. Identification of overlapping communities and their hierarchy by locally calculating community-changing resolution levels. *Journal of Statistical Mechanics: Theory and Experiment* 2011, 01 (2011), P01023.

[14] Wassily Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association* 58, 301 (1963), 13–30.

[15] Andrea Lancichinetti, Santo Fortunato, and János Kertész. 2009. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics* 11, 3 (2009), 033015.

[16] Andrea Lancichinetti, Filippo Radicchi, José J Ramasco, and Santo Fortunato. 2011. Finding statistically significant communities in networks. *PloS one* 6, 4 (2011), e18961.

[17] Conrad Lee, Fergal Reid, Aaron McDaid, and Neil Hurley. 2010. Detecting highly overlapping community structure by greedy clique expansion. *SNA/KDD'10* (2010), 33–42.

[18] Hiroshi Nagamochi and Toshihide Ibaraki. 1992. Computing edge-connectivity in multigraphs and capacitated graphs. *SIAM Journal on Discrete Mathematics* 5, 1 (1992), 54–66.

[19] Mark EJ Newman. 2004. Fast algorithm for detecting community structure in networks. *Physical review E* 69, 6 (2004), 066133.

[20] Mark EJ Newman and Michelle Girvan. 2003. Mixing patterns and community structure in networks. In *Statistical mechanics of complex networks*. Springer, 66–87.

[21] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* 435, 7043 (2005), 814–818.

[22] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. 2004. Defining and identifying communities in networks. *PNAS'04* 101, 9 (2004), 2658–2663.

[23] Erzsébet Ravasz, Anna Lisa Somera, Dale A Mongru, Zoltán N Oltvai, and A-L Barabási. 2002. Hierarchical organization of modularity in metabolic networks. *Science* 297, 5586 (2002), 1551–1555.

[24] Mojtaba Rezvani, Weifa Liang, Wenzheng Xu, and Chengfei Liu. 2015. Identifying top-k structural hole spanners in large-scale social networks. In *CIKM'15*. 263–272.

[25] Martin Rosvall and Carl T Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *PNAS'08* 105, 4 (2008), 1118–1123.

[26] Martin Rosvall and Carl T Bergstrom. 2011. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PloS one* 6, 4 (2011), e18209.

[27] Marta Sales-Pardo, Roger Guimera, André A Moreira, and Luís A Nunes Amaral. 2007. Extracting the hierarchical organization of complex systems. *PNAS'07* 104, 39 (2007), 15224–15229.

[28] Satu Elisa Schaeffer. 2007. Graph clustering. *Computer science review* 1, 1 (2007), 27–64.

[29] Huawei Shen, Xueqi Cheng, Kai Cai, and Mao-Bin Hu. 2009. Detect overlapping and hierarchical community structure in networks. *Physica A: Statistical Mechanics and its Applications* 388, 8 (2009), 1706–1712.

[30] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world' networks. *nature* 393, 6684 (1998), 440–442.

[31] Joyce Jiyoung Whang, David F Gleich, and Inderjit S Dhillon. 2013. Overlapping community detection using seed set expansion. In *CIKM'13*. 2099–2108.

[32] Jierui Xie, Stephen Kelley, and Boleslaw K Szymanski. 2013. Overlapping community detection in networks: The state-of-the-art and comparative study. *Comput. Surveys* 45, 4 (2013), 43.

[33] Jaewon Yang and Jure Leskovec. 2013. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *WSDM'13*. ACM, 587–596.

[34] Ying Zhao, George Karypis, and Usama Fayyad. 2005. Hierarchical clustering algorithms for document datasets. *Data mining and knowledge discovery* 10, 2 (2005), 141–168.