Hierarchical Community Decomposition Via Oblivious Routing Techniques

W. Sean Kennedy*
Bell Labs, Alcatel Lucent
Murray Hill, NJ
kennedy@research.bell-labs.com

Jamie Morgenstern†
Carnegie Mellon University
Pittsburgh, PA
jamiemmt.cs@gmail.com

Gordon Wilfong
Bell Labs, Alcatel Lucent
Murray Hill, NJ
gtw@research.bell-labs.com

Lisa Zhang‡
Bell Labs, Alcatel Lucent
Murray Hill, NJ
ylz@research.bell-labs.com

ABSTRACT

The detection of communities in real-world large-scale complex networks is a fundamental step in many applications, such as describing community structure and predicting the dissemination of information. Unfortunately, community detection is a computationally expensive task. Indeed, many approaches with strong theoretic guarantees are infeasible when applied to networks of large scale. While numerous approaches have been designed to scale community detection algorithms to cope with large-scale networks many of these proposed techniques leverage local optimizations or local greedy decisions to iteratively find the communities. By solely relying on these techniques to locally optimize the choice of communities, it is easy to miss the global structure of the network.

We take a different approach by formulating a notion of a hierarchical community decomposition (HCD) that takes a more global view of community structure. Our main contributions are as follows. We formally define A (λ, δ)-HCD where λ parameterizes the connectivity within each subcommunity at the same hierarchical level and δ parameterizes the relationship between communities across two consecutive levels. Based on a method of Racke originally designed for oblivious routing, we provide an algorithm to construct a HCD and prove that an $(O(\log n), O(1))$-HCD can always be found for any n-node input graph. Further, our algorithm does not rely on a pre-specified number of communities or depth of decomposition. Since the algorithm can take exponential time, we also describe a practical efficient implementation and perform an experimental validation of this heuristic on synthetic and real-world networks. We start with "friendly" synthetic networks for which the quality of the decompositions produced by our practical implementation can be verified. This gives us confidence in our experimental findings on the global community structure of the more complex real-world networks.

*Partially supported by NSERC PDF and the NIST Grant No. 60NANB10D128.
†Work partially done while at Bell Labs.
‡Work partially supported by the NIST Grant No. 60NANB10D128.

1. INTRODUCTION

The recent explosion of accessible information driven, in part, by the popularity of online social networks, has led many research, business and marketing communities to examine the resultant large social data sets. Mining these data sets promises to create a wealth of information about how societies, objects and ideas interact. These interactions are naturally represented by a graph. For example, in Facebook's recently announced graph search, users are nodes and a subset of the friend relationships yields the links [16]. Of the multitude of questions surrounding the analysis of these networks, one of the most noticeable efforts focuses on *community detection*. Understanding community structure is a fundamental step in many applications, such as web search, biology, social network analysis, music, business structure. See for example [13, 22, 20, 23, 33, 34, 36].

The community structure of many networks is naturally hierarchical. Communities can be iteratively divided into smaller sub-communities each of which are also good communities. Such hierarchical structures have been observed in nature, social networks, and more, see for e.g., [38, 15, 12, 40, 22]. Our goal is to capture the inherent hierarchical structure of a graph. Towards this end we define a *hierarchical community decomposition* (HCD) of a network. Given a network $G = (V, E)$, an HCD of G may be best described by an associated *decomposition tree* $T = (\mathcal{V}, \mathcal{E})$, where T is a rooted

tree such that each node $a \in \mathcal{V}$ is labeled by a subset $V_a \subseteq V$ where these labels satisfy two properties. First, the root r is labeled by $V_r = V$, and second, for each node $a \in \mathcal{V}$ with children $c_1, c_2, ..., c_k$, the subsets $V_{c_1}, V_{c_2}, ..., V_{c_k}$ form a partition of V_a.

Informally, every node $a \in \mathcal{V}$ corresponds to a "good" community induced on V_a in G. (We discuss the question of what it means to be good below.) Furthermore, the decomposition tree also reflects the global structure of G. For example in Figure 1, community B consists of A_1, A_2, A_3 and A_4 each of which is obviously a good sub-community individually. However, $A = A_1 \cup A_2$ appears to form a natural sub-community as well. Therefore the decomposition tree shown in Figure 1.c captures the global structure more completely than the one in Figure 1.b.

This leads to a basic question, *what is a good community*? Towards answering this question, many notions have been explored. From a graph theoretic view, a good community is likely to have a small diameter, i.e. the shortest path distance between any node pairs is small. Natural variations of small diameter based community detection include the k-center, k-median and k-mean problems. Alternatively, a good community is likely to be sufficiently dense, where the density reflects the nodes within the subgraph interact more with one another than with those outside the subgraph. Examples of this include high maximum and average node degrees within the community [26]. Despite simplicity these simple classes of measures do not consistently lead to good communities. As we discuss now, more sophisticated measures have proven to be more successful.

One of the most intensively studied notions of a community stems from the rich theory of graph cuts. Given a subset S of the nodes, a *cut* is the set of edges each of which has one endpoint inside S and one outside of S. Generally speaking, *conductance* (and its many variants) measures the ratio of the cut size to the number of edges inside each side of the cut. Formally, the *conductance* of $S \subseteq V$ in graph $G = (V, E)$, is

$$\varphi(S) := \frac{|\{xy \in E | x \in S, y \notin S\}|}{\min\{\sum_{v \in S} \deg(v), \sum_{v \in V - S} \deg(v)\}}.$$

A min-conductance subgraph is intuitively associated with a good community as it captures the notion of more edges within the subgraph than leaving the subgraph.

We construct hierarchical community decompositions (HCD) by adopting a scheme for oblivious routing originally designed by Harold Räcke [37]. A byproduct of Räcke's routing algorithm is a pair of connectivity conditions, each of which corresponds to our intuition of community structure. The first rule is similar to conductance and insists that any community must have ample internal connections relative to external connections. The second rule can be thought of as a conductance type measure on the sub-communities ensuring that global structure is represented in the decomposition. It insists that any subset of these sub-communities must not have many inter-community connections; otherwise, this set of sub-communities should together form a larger community. These two opposing forces guide each iteration of the algorithm, balancing between individual community quality and overall global structure.

One method to find a HCD is to determine one level of the hierarchy by applying a community detection algorithm to the original graph and then iteratively applying the algorithm to each of the sub-communities to extend the hierarchy. However, by focusing on local optimization of the choice of communities, it is easy to miss the global structure of the network. For example in Figure 1, if A_1, A_2, A_3 and A_4 are the best communities, an iterative algorithm may well find them all and put them in the same level of hierarchy, but miss the global structure that includes an extra level in which $A_1 \cup A_2$ also forms a good community A, though A is somewhat weaker than $A_1 \dots A_4$. On the other hand, the second rule in our HCD approach based on Räcke decomposition is designed to capture such structure.

Since determining a subgraph of minimum conductance is NP-hard [41], several well-studied heuristic algorithms have been introduced. Recent work [32] makes an extensive empirical comparison of a number of these methods, and notes that they seem to present particular biases. For example spectral methods tend to find very dense subgraphs as communities which may still be well-connected to the rest of the graph, whereas some other methods may result in a small number of edges crossing the cuts between communities, but unfortunately poor internal connectivity. If a community of less desirable bias is found in one iteration, it is likely that an iterative application for community decomposition will propagate an early error. In our HCD approach, we allow multiple iterations in breaking up the potential sub-communities and recombining them until we converge to a satisfactory partition. Thus, we believe the opposing forces of partitioning and recombining provide an opportunity for patching errors that min-conductance heuristics could introduce.

1.1 Contribution and Organization

- In Section 2.1 we formally define a hierarchical community decomposition, (λ, δ)-HCD, where λ parameterizes the connectivity within each sub-community at the same hierarchical level and δ parameterizes the relationship between communities across two consecutive levels.

- In Section 2.2 we provide an algorithm to construct a (λ, δ)-HCD and prove that an $(O(\log |V|), O(1))$-

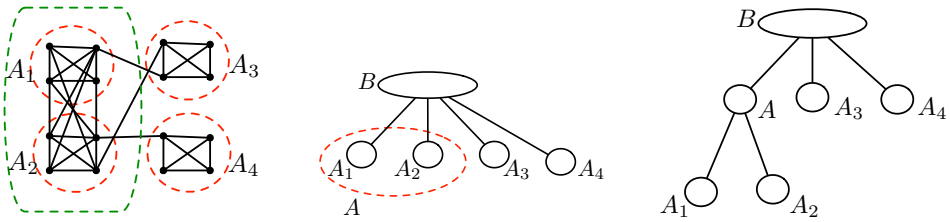

Figure 1: (a). Network B. (b) A decomposition tree that misses A in the global structure. (c) A desirable hierarchy formed by A, A_1 and A_2.

HCD can always be found for any input graph. A similar and slightly weaker result is stated and proved by Racke for oblivious routing [37], although our proof is simpler and is geared towards HCD not routing. Since this algorithm takes exponential time to examine an exponential number of cuts, we propose an efficient heuristic implementation via spectral cuts in Section 2.3.

- In Section 3 we detail our experimental results. We first use a small network to verify our earlier point that our algorithm is able to recover from a less-than-ideal partition provided by spectral cuts whereas performing iterative spectral cuts may not. In addition, for small enough examples that the exact algorithm is feasible and we observe that our efficient heuristic in fact returns a HCD very close to the solution given by the exact algorithm. We then proceed to "friendly" synthetic networks for which the quality of the decompositions produced by our efficient heuristics can be verified. We finally explore several real-world complex networks and explore their global community structure via our efficient heuristic.

1.2 Related Work

We turn to a brief overview on methods for determining communities. We do not attempt an extensive and complete review, instead referring the interested reader to [19, 9, 29, 39]. Also, Leskovek et al. [32] present a comprehensive empirical study of numerous methods for determining communities.

Communities: Hardness, Approximations & Heuristics.

Many of the cut-based criterion for measuring community quality give rise to NP-hard optimization problems. See for example [21] or [24] for a survey up to 1996 of approximating minimum-conductance, expansion, normalized or sparsest cuts. Flow-inspired algorithms for determining communities have been the subject of considerable study [17, 18]. Of particular interest to this work is minimum-conductance cuts. Leighton and Rao prove an $O(\log(n))$-approximation [30]; Arora *et al.* show an improved approximation of $O(\sqrt{\log(n)})$ [5]. Heuristics for attacking this problem use the Cheeger inequality which describes the relationship between the edge expansion of a graph and the second eigenvalue of the normalized Laplacian matrix graph [10]. This leads to the use of spectral methods to find low-conductance cuts [11].

The Nibble method by Spielman and Teng [42] finds a community of low conductance containing a starting vertex v, in time nearly linear in the size of the output community. The algorithm starts by processing a truncated distribution similar to that of a short random walk starting at v: those vertices that occur in short random walks are more likely to be a part of a low-conductance community with v. Formally, if one selects a vertex v from a community C with conductance $O(\phi^2/\log^3(n))$, the Nibble method returns a community that is mostly contained in C with conductance at most ϕ, with a constant probability.

Local spectral methods [2] find low-conductance cuts by computing approximate PageRank vectors and conducting sweeps over those vectors. They show a mixing result for approximate PageRank vectors that implies a ϕ-conductance cut can be found provided an $\Omega(\phi^2/\log(m))$-conductance cut exists. Extending the Nibble method, the algorithm also starts from a single vertex v and finds a low-conductance community that usually contains v. A follow-up paper [3] begins with set of well-connected "seed" vertices rather than a single vertex. This was inspired by the HITS algorithm [28] which begins with a seed set generated by a search engine and then does a constant-depth breadth-first-search to construct a community containing that set.

Kannan, Vempala, and Vetta [26] generalize the conductance-based objective to a bicriterial one. An (α, ϵ)-clustering has conductance at least α for each cluster, and has at most an ϵ-fraction of the edge weights crossing between clusters. They show that finding a spectral cut and recursing on the two sides achieves the best (α, ϵ)-approximation guarantee.

Hierarchical Community Decomposition.

Other than iteratively applying a community detec-

tion algorithm to find a hierarchical decomposition, a variety of more global approaches have been developed.

Consider a basic algorithm that begins with an initially empty graph on n vertices and adds edges one at a time, in the order of decreasing weight. Whenever two disconnected components are joined, they can be thought of as subcommunities in the larger community which is their union. Therefore, this basic algorithm produces decompositions that are hierarchical in nature. This approach tries to bridge the gap between scalable methods and global ones: the weights can be defined in terms of more global properties of the graph, but the algorithm itself is polynomial time. For example, the weights on these edges can be the original weights from the community graph, or a function of the number of paths between two vertices [35, 27], where long paths are usually weighed significantly less important than short paths. Girvan *et al.* [22] propose an almost dual approach by starting with the full graph and iteratively removing edges.

Balcan and Liang [7] formalize a metric to measure hierarchical community structure. They describe a bottom-up construction of stable communities: each community C must be sufficiently large, points in C must have most of their neighbors in C, and points outside C can only have a few of their neighbors in C. The authors propose an algorithm that begins with singleton sets as communities, and merges sets if there is sufficient overlap in their neighborhoods, and show the algorithm produces a hierarchy that satisfies their definition of stability. Huang *et al.* [25] describe a modularity-based bottom-up approach to detect hierarchical communities. Good communities are iteratively contracted into larger super-communities.

Work of [8, 4] also consider potentially overlapping communities.

2. HIERARCHICAL COMMUNITY DECOMPOSITIONS

2.1 (λ, δ)-HCD

We now give a formal definition of HCD, which stems from the first work [37] in the sequence of papers on oblivious routing by Racke and coauthors. Let $G = (V, E)$ be an undirected weighted input graph, where w_{ij} or w_e represents the weight of an edge $e = ij \in E$. We interpret w_{ij} as similarity between the two nodes i and j, the heavier the weight the more alike the two nodes. Recall an HCD of G is described by the associated *decomposition tree* $T = (\mathcal{V}, \mathcal{E})$, where T is a rooted tree such that each node $a \in \mathcal{V}$ is labeled by a subset $V_a \subseteq V$ where these labels satisfy the following two properties:

- the root r is labelled by $V_r = V$, and
- for each node $a \in \mathcal{V}$ with children $c_1, c_2, ..., c_k$, the subsets $V_{c_1}, V_{c_2}, ..., V_{c_k}$ form a partition of V_a.

We enforce two conditions. First, if A is a good community then any subset of A must have ample connections within A relative to connections external to A; otherwise, this subset is not sufficiently connected to other nodes in A, and A is not a good community.

DEFINITION 1. *The total connections, or* capacity, *between two (not necessarily disjoint) subsets X and Y of V is*

$$\mathbf{cap}(X, Y) = \sum_{x \in X, y \in Y} w_{xy},$$

where $w_{xy} = 0$ whenever xy is not an edge of G.

DEFINITION 2. *Let $V' \subset V$. For $\lambda \geq 1$, a subset of $U \subset V'$ is* λ-detached with respect to V' *if $|U| \leq \frac{1}{2}|V'|$ and if*

$$\frac{\mathbf{cap}(U, V \setminus U)}{\mathbf{cap}(U, V' \setminus U)} \geq \lambda.$$

We say V' is λ-detachable if some *subset $U \subset V'$ is λ-detached w.r.t. V'.*

As argued above, for an appropriate value of $\lambda \geq 1$, no good community should contain a λ-detached subgraph. Hence, our first condition on the hierarchical decomposition tree T is that for each $a \in \mathcal{V}$, V_a is not λ-detachable.

The second condition takes the hierarchical view and tries to ensure that all levels of the hierarchy are present even when good smaller communities exist. Recall the example in Figure 1. Let A be a good community containing two sub-communities A_1 and A_2. Intuitively, there should exist nodes labelled by A, A_1 and A_2 in T where A is an ancestor of A_1 and A_2. Figure 1.c shows an example of a good hierarchy. In contrast, Figure 1.b demonstrates a less than satisfactory hierarchy, as it does not capture A_1 and A_2 together form a community in addition to being separate communities themselves. We capture this notion more formally in Definitions 3 and 4 below.

Let a be any non-leaf node of a hierarchical decomposition tree T, and let $c_1, c_2, ..., c_k$ be its children. The above intuition suggests that the only combination of the communities $V_{c_1}, V_{c_2}, ..., V_{c_k}$ that results in a good community is $V_a = \bigcup_i V_{c_i}$. To formalize this, we define the following.

DEFINITION 3. *Let $V' \subset V$. Let $\delta \geq 1$ and $P_1, P_2, ..., P_k$ be a partition of V'. A subgraph $R = \bigcup_{i \in I} P_i \subset V'$, where I is a subset of $\{1, 2, ..., k\}$ is* δ-linked with respect to V' *if $|R| \leq \frac{1}{2}|V'|$ and if*

$$\frac{\sum_{i,j \in I, i \neq j} \mathbf{cap}(P_i, P_j)}{\sum_{i \in I, j \notin I} \mathbf{cap}(P_i, P_j)} > \delta.$$

We say V' with a partition of $P_1, P_2, ..., P_k$ is δ-linkable if $R = \bigcup_{i \in I} P_i$ is δ-linked w.r.t. V' for some *subset I.*

Our second condition is that if $c_1, c_2, \ldots, c_k$ are the children of a in T then V_a with a partition of $V_{c_1}, \ldots, V_{c_k}$ is not δ-linkable for an appropriately chosen $\delta \geq 1$.

We are now able to describe the decomposition we are interested in finding.

DEFINITION 4. *A (λ, δ)-*hierarchical decomposition *of a graph $G = (V, E)$ is a decomposition tree $T = (\mathcal{V}, \mathcal{E})$ satisfying the following three properties:*

(1) For each node $a \in \mathcal{V}$, V_a is not λ-detachable.

(2) For each non-leaf node $a \in \mathcal{V}$ with children $c_1, c_2, ..., c_k$, V_a with the partition of $V_{c_1}, \ldots, V_{c_k}$ is not δ-linkable.

(3) For each leaf node $\ell \in \mathcal{V}$, $|V_\ell| = 1$.

Finding a decomposition satisfying only Conditions (1) and (2) is trivial. Indeed, a single node labelled by V will suffice. As we discuss in the next section, for carefully chosen λ and δ there always exists a decomposition also satisfying Condition (3). Clearly, such values exist since a $(1, |E|)$-hierarchical decomposition of a graph $G = (V, E)$ is a root r labelled by $V_r = V$ and adjacent $|V|$ leaves each labeled by a unique vertex of G. In addition, any binary decomposition tree T defines a $(|E|, 1)$-hierarchical decomposition of a graph, since each non-leaf node a with two children c_1 and c_2 imply vacuously that V_a with partition V_{c_1} and V_{c_2} is not δ-linkable. As we discuss below, Räcke showed that we can do a lot better. Indeed, there always exists an $(O(\log |V|), O(\log |V|))$-hierarchical decomposition [37]. In the next section, we strengthen his result by showing that there always exists an $(O(\log |V|), O(1))$-hierarchical decomposition. Our improvement plays a key role in our practical implementation of this algorithm described in Section 2.3.

2.2 Exact Algorithms for HCD

For fixed $\lambda \geq 1$ and $\delta \geq 1$, a potential algorithm to construct a (λ, δ)-hierarchical decomposition of a graph $G = (V, E)$ follows in a straightforward way from Definition 4. We iteratively build decomposition trees $T_i = (\mathcal{V}_i, \mathcal{E}_i)$, $i = 0, 1, ...$, each satisfying Conditions (1) and (2) starting with T_0 and stopping when T_i also satisfies Condition (3).

We let T_0 be a single node r labelled by V. Note that (1) holds since $V_r = V$ and so $\mathbf{cap}(U, V \setminus U) = \mathbf{cap}(U, V_a \setminus U)$ for all proper subsets U of V, and (2) holds since the unique node of T_0 is a leaf. For $i \geq 0$, assume T_i satisfies (1) and (2), but not (3). Let ℓ be a leaf node with $|V_\ell| > 1$. Now, the key is to find a partition $P_1, ..., P_k$ of V_ℓ such that V_ℓ is not δ-linkable and that no P_i is λ-detachable. (In fact, a stronger statement is captured in Lemma 6 below.) It follows that the decomposition tree T_{i+1} found by adding children $P_1, ..., P_k$ to node ℓ in T_i, respectively labelled by $P_1, ..., P_k$, satisfies (1) and (2).

Therefore, there must exist some i' for which $T_{i'}$ also satisfies (3) since in each iteration we partition some leaf node a into non-empty parts each of size at most $\frac{1}{2}|V_a|$.

More rigorously, the following Theorem 5 states a (λ, δ)-hierarchical decomposition can always be found for logarithmic values of λ and δ. A similar and slightly weaker result is stated and proved in [37], although our proof is simpler and is geared towards HCD not routing.

THEOREM 5. *Let $\lambda \geq 4 \log |V|$ and $\delta \geq (1 - \frac{2}{\lambda} \log |V|)^{-1}$. For any graph $G = (V, E)$ there exists an (λ, δ)-hierarchical decomposition $T = (\mathcal{V}, \mathcal{E})$.*

To prove this theorem, it is enough to prove Lemma 6 which iteratively builds the decomposition tree starting from the root, as described above. The algorithm described in pseudocode 1 and 2 implement the process.

LEMMA 6. *Assume $V' \subseteq V$ satisfies $|V'| > 1$ and is not λ-detachable. Then, there exists a partition $P_1, P_2, ..., P_k$ of V' such that*

(i) for each $i = 1...k$, $|P_i| \leq \frac{1}{2}|V'|$,

(ii) no P_i is λ-detachable, and

(iii) no subset $I \subset \{1, 2, ..., k\}$ is such that $\bigcup_{i \in I} P_i$ is δ-linkable w.r.t. V'.

In fact, Algorithms 1 and 2 find such a partition.

PROOF OF LEMMA 6. Let $\mathcal{P}_0 = \{\{v\} \mid v \in V'\}$, that is, each node of V' is in its own part. Clearly, $\mathcal{P}_0$ satisfies (i) and (ii). We iteratively construct partitions $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_j, ...$ of V' each of which satisfy (i) and (ii). We stop when we find a partition $\mathcal{P}_{j'}$ also satisfying (iii). We now describe and prove an algorithm which does this. The reader may wish to refer to the pseudocode given by Algorithm 1 and Algorithm 2.

Algorithm 1: PARTITION

Input: $V' \subset V$ such that V' is not λ-detachable.
Output: Partition $P_1, ..., P_k$ of V' satisfying (i), (ii) and (iii).

```
begin
    set P = {{v} | v ∈ V'}
    while ∃I ⊂ {1, 2, ..., |P|} ∈ P such that
    ⋃_{i∈I} P_i is δ-linked w.r.t. V' do
        let R = DETACH(⋃_{i∈I} P_i)
        set P = (P \ ⋃_{i∈I} {P_i}) ∪ R
    end
    return P
end
```

We prove the following.

LEMMA 7. *Let $\mathcal{P}_j = \{P_1, P_2, ..., P_k\}$ satisfy (i) and (ii) but not (iii) for $j \geq 0$. Then, there exists $\mathcal{P}_{j+1}$ satisfying (i) and (ii), and such that*

$$\sum_{P \in \mathcal{P}_{j+1}} \mathbf{cap}(P, V \setminus P) < \sum_{P \in \mathcal{P}_j} \mathbf{cap}(P, V \setminus P). \quad (1)$$

Since the capacity of any $W \subset V$, $\mathbf{cap}(W, V \setminus W)$ is always positive, it follows that eventually, for some $\mathcal{P}_{j'}$, (iii) must also be satisfied. Hence to complete the proof of Lemma 6, we need only prove Lemma 7.

PROOF OF LEMMA 7. Since (iii) is not satisfied there exists $I \subset \{1, 2, ..., k\}$ such that $R = \mathbf{cap}_{i \in I} P_i$ is δ-linked w.r.t. V'. We show that we can partition R so that the resultant partition $R_1, ..., R_t$ satisfies (i), (ii) and

$$\sum_{j=1}^{t} \mathbf{cap}(R_t, R \setminus R_t) < \sum_{i \in I} \mathbf{cap}(P_i, R \setminus P_i). \quad (2)$$

This implies Lemma 7, since

$$\begin{aligned}
&\sum_{P \in \mathcal{P}_{j+1}} \mathbf{cap}(P, V \setminus P) \\
&= \sum_{j=1}^{t} \mathbf{cap}(R_t, V \setminus R_t) + \sum_{i \notin I} \mathbf{cap}(P_i, V \setminus P_i) \\
&= \sum_{j=1}^{t} \mathbf{cap}(R_t, V \setminus R) + \sum_{j=1}^{t} \mathbf{cap}(R_t, R \setminus R_t) \\
&\qquad + \sum_{i \notin I} \mathbf{cap}(P_i, V \setminus P_i) \\
&< \sum_{i \in I} \mathbf{cap}(P_i, V \setminus R) + \sum_{i \in I} \mathbf{cap}(P_i, R \setminus P_i) \\
&\qquad + \sum_{i \notin I} \mathbf{cap}(P_i, V \setminus P_i) \\
&= \sum_{i \in I} \mathbf{cap}(P_i, V \setminus P_i) + \sum_{i \notin I} \mathbf{cap}(P_i, V \setminus P_i) \\
&= \sum_{P \in \mathcal{P}_j} \mathbf{cap}(P, V \setminus P),
\end{aligned}$$

where the inequality follow from Eqn. 2 and since $R_1, ..., R_t$ and $\{P_i | i \in I\}$ are both partitions of R.

Given R we find the desired $R_1, ..., R_t$ by applying Algorithm 2. Starting with $\mathcal{R}_0 = \{R\}$, we iteratively find partitions $\mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_\ell, ...$ Given $\mathcal{R}_\ell, \ell \geq 0$ we find $\mathcal{R}_{\ell+1}$ by choosing some part R^ℓ that contains a λ-detached subgraph U with respect to R^ℓ. If no such part exists, then $\mathcal{R}_\ell$ is the desired partition. Otherwise, we set $\mathcal{R}_{\ell+1} = (\mathcal{R}_\ell \setminus \{R^\ell\}) \cup \{U^\ell, R^\ell \setminus U^\ell\}$. Since, $1 \leq U^\ell \leq \frac{1}{2}|R^\ell|$ each part is nonempty and so this process must eventually terminate.

Assuming this process terminated at $\ell = \ell'$, note that we can describe this process as a set of ordered pairs $\mathcal{U} = \{(U^1, R^1), (U^2, R^2), ..., (U^{\ell'}, R^{\ell'})\}$. For each pair

Algorithm 2: DETACH

Input: $R \subset V$
Output: Partition $R_1, ..., R_t$ of R such that each R_i, $1 \leq i \leq t$, is not λ-detachable.

```
begin
    set R = {R}
    while ∃U ⊂ R_i ∈ R such that U is λ-detached w.r.t. R_i do
        set R = (R \ R_i) ∪ {U, R_i \ U}
    end
    return R
end
```

of parts $R_i \neq R_j$, any edge $e \in E(R_i, R_j)$ is in exactly one of the sets $E(U^\ell, R^\ell \setminus U^\ell)$ and so,

$$\begin{aligned}
\sum_{j=1}^{t} \mathbf{cap}(R_t, R \setminus R_t) &= 2 \sum_{(U^\ell, R^\ell) \in \mathcal{U}} \mathbf{cap}(U^\ell, R^\ell \setminus U^\ell) \\
&\leq \frac{2}{\lambda} \sum_{(U^\ell, R^\ell) \in \mathcal{U}} \mathbf{cap}(U^\ell, V - U^\ell),
\end{aligned}$$

since each U^ℓ is λ-detached with respect to R^ℓ. We can split this sum as follows.

$$\begin{aligned}
\frac{2}{\lambda} \sum_{(U^\ell, R^\ell) \in \mathcal{U}} \mathbf{cap}(U^\ell, V - U^\ell) &= \frac{2}{\lambda} \sum_{(U^\ell, R^\ell) \in \mathcal{U}} \mathbf{cap}(U^\ell, R - U^\ell) \\
&\quad + \frac{2}{\lambda} \sum_{(U^\ell, R^\ell) \in \mathcal{U}} \mathbf{cap}(U^\ell, V - R)
\end{aligned}$$

To finish we need a simple observation.

OBSERVATION 8. *Each vertex $v \in R$ is in at most $\log |R|$ of the U^ℓ.*

PROOF. The set of U^ℓ's containing v form a nested family of subsets each at least twice a big as the previous and all contained in R. Hence, there can be at most $\log |R|$ of them. □

Observation 8 implies

$$\begin{aligned}
&\sum_{(U^\ell, R^\ell) \in \mathcal{U}} \mathbf{cap}(U^\ell, V - R) \\
&= \sum_{(U^\ell, R^\ell) \in \mathcal{U}} \sum_{v \in U^\ell} \mathbf{cap}(\{v\}, V - R) \\
&= \sum_{v \in R} \sum_{\substack{(U^\ell, R^\ell) \in \mathcal{U} \\ U^\ell \ni v}} \mathbf{cap}(\{v\}, V - R) \\
&\leq (\log |R|) \sum_{v \in R} \mathbf{cap}(\{v\}, V - R) \\
&= (\log |R|) \mathbf{cap}(R, V - R).
\end{aligned}$$

In a similar way, Observation 8 implies that every part of the final partition $R_1, ..., R_t$ is contained in at most

$\log|R|$ of the U^ℓ's. Hence,

$$\begin{aligned}
&\sum_{(U^\ell,R^\ell)\in\mathcal{U}} \mathbf{cap}(U^\ell, R-U^\ell)\\
\leq\ &\sum_{(U^\ell,R^\ell)\in\mathcal{U}}\ \sum_{R_i\subseteq U^\ell} \mathbf{cap}(R_i, R-R_i)\\
\leq\ &\log|R|\sum_{i\in I}\mathbf{cap}(R_i, R\setminus R_i).
\end{aligned}$$

So, together we have

$$\begin{aligned}
&\sum_{j=1}^{t}\mathbf{cap}(R_t, R\setminus R_t)\\
\leq\ &\frac{2\log|R|}{\lambda}\left[\mathbf{cap}(R_i, R\setminus R_i)+\mathbf{cap}(R, V-R)\right] \quad (3)
\end{aligned}$$

By assumption $R=\bigcup_{i\in I}P_i$ is δ-linked w.r.t. V' and so $\mathbf{cap}(R, V-R) < \frac{1}{\delta}\sum_{i\neq j\in I}\mathbf{cap}(P_i,P_j)$. This together with our choice of λ and δ, Eqn. 3 is then equivalent to the following.

$$\begin{aligned}
&\sum_{j=1}^{t}\mathbf{cap}(R_t, R\setminus R_t)\\
<\ &\left(1-\frac{2\log|R|}{\lambda}\right)^{-1}\frac{1}{\delta}\sum_{i\neq j\in I}\mathbf{cap}(P_i,P_j)\\
<\ &\sum_{i\neq j\in I}\mathbf{cap}(P_i,P_j).
\end{aligned}$$

This completes the proof of Lemma 7, and hence, Lemma 6.

Remark. Our proof shares many ideas with a proof of Räcke's showing that there always exists such a $(O(\log|V|), O(\log|V|))$-hierarchical decomposition tree $T=(\mathcal{V},\mathcal{E})$ satisfying the height of T is at most $O(\log|V|)$. This additional height constraint was important in the design of his oblivious routing scheme. Note that since Condition (i) insists that each child node is at most half the size of its parent, the height of our decomposition tree is also $O(\log|V|)$. For the purpose of hierarchical decompositions, a height constraint seems unnatural, and hence, we have not included it.

2.3 Efficient Implementation

As described, Algorithms 1 and 2 are inefficient since an exponential number of subsets may need to be considered in the while-loop in line 3 of both algorithms. Later work of Racke and coauthors, e.g. [6], achieves a polynomial running time via approximating the cuts, though their solutions are still too complex to be implementable. Further, these new formulations diverge from the initial formulation, whose mathematical formulation corresponds directly with the intuition of what makes a community, namely strong internal connectivity with poor external connectivity.

In this section we focus on a scalable implementation.Line 3 of Algorithm 1 and Line 3 of Algorithm 2 are the crux of this method. Instead of checking *each* subset in Line 3 of the two algorithms, we propose checking subsets generated by spectral cuts in the graph. In the case of Algorithm 1, consider the reduced multigraph $G_{\mathcal{P}}$, where each piece of the current partition is collapsed into a single node. A spectral cut in $G_{\mathcal{P}}$ is likely to be well-connected internally, implying the sets on one side of the cut are good candidates for δ-linkedness. Algorithm 3 is randomized and only a heuristic for finding low conductance cuts. Hence, we require that Algorithm 3 fails a fixed number of times before we assume that no δ-linkable set exists. Initially we set this failure rate to be 10, but experimentation showed that 3 was sufficient, greatly improving the runtime of our heuristic.

Similarly, a spectral cut of R_i approximates the minimum conductance cut in a graph [26]. This implies there is some measure by which the intra-connectivity of a spectral cut is low. Thus, we propose replacing checking all subsets of each R_i for λ-detachability by only checking *spectral cuts* of each R_i. Algorithm 3 outlines a simple spectral cut method which replace Line 3, run on $G_{\mathcal{P}}$ for 1, and run on $G|_{R_i}$ for 2. Again, we require that Algorithm 3 fails a fixed number of times before we assume that no λ-detachable set exists.

The proof of Lemma 7 shows that the choice of δ bounds the number of iterations of Algorithm 1 since

$$\tau(\{R_1,...,R_t\}) := \sum_{j=1}^{t}\mathbf{cap}(R_t, R\setminus R_t)$$

decreases by at least a factor of $\frac{1}{\delta}$ in each iteration. Since $\tau(\{R_1,...,R_t\}) \leq |E|$, by choosing $\delta\geq 2$ we can ensure their are at most $\log_2|E|$ iterations. Additionally, we can ensure our heuristic is not stuck in an infinite loop by insisting that $\tau(\{R_1,...,R_t\})$ is strictly decreasing in each iteration.

Algorithm 3: SpectralCut

Input: $V'\subset V$
Output: Cut (P_1,P_2) of V'.

```
1 begin
2   let L = Laplacian of subgraph induced on V'
3   let (u1, u2) = smallest 2 eigenvectors of L
4   let (C1, C2) be the 2-means clustering of the n
      columns of (u1, u2)
5   let (P1, P2) be the partition of V' according to
      (C1, C2)
6   return (P1, P2)
7 end
```

3. EXPERIMENTS

In this section, we describe some experimental results obtained by applying our method for constructing a hierarchical community decomposition (HCD). We consider the exact implementation of Algorithm 1 PARTITION and Algorithm 2 DETACH in which the all cuts are exhaustively enumerated in the while-loops, and a practical implementation in which the exhaustive enumeration is replaced by Algorithm 3 SPECTRALCUT.

There are three components to our experiments. First, we consider a very small graph for which the exact implementation of our algorithm is feasible. We observe that the decomposition produced by our practical implementation matches that produced by the exact implementation, therefore achieving the theoretical guarantees. We also observe that the first spectral cut in the practical implementation is in fact a bad cut. Our algorithm is able to correct such a mistake through the interplay of gluing together and breaking apart communities. On the other hand a straight-forward iterative implementation of spectral cut would not be able to recover from such a mistake. Second, we present a set of synthetic graphs for which an exact implementation of our algorithm is already infeasible. However, each synthetic example has a planted hierarchical decomposition that serves as a reference point. We observe that the HCDs provided by our practical algorithms are close to the planted decompositions. Finally, we use a set of complex real-world networks for which we do not know what desired decompositions look like. We present the decompositions we find and some observations. However, our validation on the smaller networks give us confidence in our experimental findings on the global community structure of these large networks.

3.1 A Small Example

We begin with very small networks. Other than the exact and practical implementation of our decomposition algorithm, we also consider applying spectral cuts iteratively. By iterative spectral cuts, we mean applying SPECTRALCUT to subgraphs returned from previous applications of SPECTRALCUT.

Consider the graph illustrated in Figure 2. This graph G consists of four clique-like subgraphs C_{11}, C_{12}, C_{21} and C_{22}, each with 5 nodes, and a small number of links between these four subgraphs.

The set $S = C_{11} \cup \{x\}$ of Figure 2 corresponds to the first spectral cut that incorrectly placed one red vertex x on the wrong side of the partition. Specifically, vertex x has a single neighbor inside S and three neighbors in $G - S$, thus the cut $(S - \{x\}, (G - S) \cup \{x\})$ has better conductance. If we apply spectral cut iteratively, we will never recover from this mistake of selecting S as a community.

On the other hand the practical implementation of

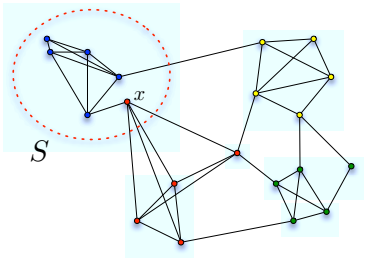

Figure 2: A 20-node graph G consists of four clique-like subgraphs. The first spectral cut yields the cut $(S, G - S)$, placing vertex x not in its "best" community, that is, the one with red colored vertices.

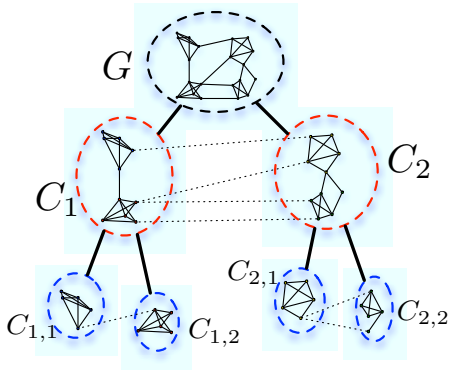

Figure 3: Applying PARTITION / DETACH to the graph G of Figure 2. Edges between communities at the same level are shown as dotted lines.

our algorithm also initially suffers from the bad cut S, but manages to correct it due to the interplay of δ-linkedness and λ-detachness. The former glues together nodes into one potential community and the latter breaks apart a potential community. In this example, $\{x\}$ is λ-detached with respect to S and therefore S can be corrected. Figure 3 describes our decomposition as applied to G. This highlights a key benefit of our algorithm: *the δ-linked and λ-detached conditions can correct mistakes made by spectral clustering, even when using spectral clustering as a subroutine!* Here is the progression of our algorithm.

1. $\mathcal{P} = \{\{v\} \mid v \in V\}$.
2. PARTITION discovers via SPECTRALCUT that $S = C_{1,1} \cup \{x\}$ is δ-linked. DETACH then discovers S is λ-detachable and SPECTRALCUT returns $(C_{1,1}, \{x\})$. Subsequently, P is updated to $\{C_{1,1}, \{v\} | v \in V \setminus C_{1,1}\}$.
3. PARTITION discovers via SPECTRALCUT that $C_{1,2}$ is δ-linked. $C_{1,2}$ is not λ-detachable. P is updated to $\{C_{1,1}, C_{1,2}, \{v\} | v \in V \setminus C_{1,1} \setminus C_{1,2}\}$. Similarly,

PARTITION discovers via SPECTRALCUT that $C_{2,1}$ and $C_{2,2}$ are δ-linked, and P is updated to $\{C_{1,1}, C_{1,2}, C_{2,1}, C_{2,2}\}$.

4. PARTITION further discovers $\{C_{1,1}, C_{1,2}\}$ are linked which form C_1, and $\{C_{2,1}, C_{2,2}\}$ are linked which form C_2.

Finally, the exact implementation of our algorithm results in the same decomposition in this example. Hence, we know the resulting decomposition satisfies the theoretical guarantees of Theorem 5.

3.2 Larger Synthetic Examples

We now consider a set of synthetic examples, drawing inspiration from Condon and Karp's *planted partition model* [14]. These examples are sufficiently large that our exact decomposition algorithm is no longer feasible. When a graph has a clear unique hierarchical structure, we observe that the practical implementation of our algorithm returns a decomposition that matches the planted decomposition nearly perfectly. On the other hand, when a graph does not have a unique good decomposition, we observe that our algorithm may return decompositions that are different from the planted ones. However, we verify these computed decompositions are in fact sensible judging by their conductance values.

Let us first describe the construction of the synthetic examples. The *planted ℓ-partition model with probabilities $p_1 > p_2$* generates a graph G whose vertex set V can be partitioned into ℓ subsets each of size n/ℓ such that two nodes within the same partition are adjacent with probability p_1 and two nodes in different partitions are adjacent with probability p_2. For $\ell = 2$, if p_1 is substantially larger than p_2 then the minimum bisection of the graph is expected to be the planted 2-part partition [14].

To test the validity of our hierarchical decomposition algorithm, we generalize the Condon-Karp by model defining the *k-level hierarchical planted ℓ-partition model*, or *(k, ℓ)-HPPM* recursively as follows. The $(1, \ell)$-HPPM with probabilities $p_1 > p_2$ is identical to the planted ℓ-partition model with probabilities $p_1 > p_2$. A graph generated from $(2, \ell)$-HPPM with probabilities $p_1 > p_2 > p_3$ consists of ℓ graphs $G_1, G_2, ..., G_\ell$ each of which is independently generated from $(1, \ell)$-HPPM with probabilities $p_1 > p_2$, and additionally each node of G_i is adjacent to each node of G_j, $i \neq j$ with probability $p_3 < p_2$. In general, a graph G generated from (k, ℓ)-HPPM with probabilities $p_1 > p_2 > \cdots > p_{k+1}$ consists of ℓ graphs $G_1, G_2, ..., G_\ell$ each of which is independently generated from $(k-1, \ell)$-HPPM with probabilities $p_1 > p_2 > \cdots > p_k$ and additionally each node of G_i is adjacent to each node of G_j, $i \neq j$, with probability $p_{k+1} < p_k$.

Given a graph G generated from $(k, 2)$-HPPM with probabilities $p_1 \gg p_2 \gg ... \gg p_{k+1}$, i.e., each p_i is much smaller than its predecessor in the order, the ideal hierarchical partitioning is clear: the root should be labeled by the vertices of G where its children should be labeled by the 2 $(k-1, 2)$-HPPM graphs used in generating G. A full decomposition is found by iterating this process.

Figure 4 gives an example of a 64-node graph $G_{1,2}$ generated using the $(1, 2)$-HPPM with $p_1 = 0.2$ and $p_2 = 0.1$. In general, spectral methods have no trouble recovering the two planted communities. It is not surprising that the practical implementation of our algorithm closely recovers the planted hierarchical decomposition. Note that the k-means procedure in Algorithm 3 SPECTRALCUT is randomized. The output therefore varies between trials. In addition the parameter choices for λ and δ also contribute to the varying output. However, none of the trials deviate far from the planted decomposition. For $\lambda = 3.0$ and $\delta = 1.0$, we see a 3-level hierarchy: the top labeled by the nodes $G_{1,2}$, its children two communities each of size $|G_{1,2}|/2 = 32$, of conductance 0.231 and 0.224, where each of these nodes have 32 singleton children. More generally, for each

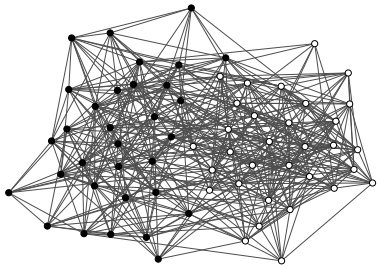

Figure 4: $G_{1,2}$: a 64-node graph generated using the $(1, 2)$-HPPM with $p_1 = 0.2$ and $p_2 = 0.1$. The planted partition is indicated using the black and white colored nodes.

$k = 2, 3, ..., 8$ we generated $G_{k,2}$ using the $(k, 2)$-HPPM where $|G_\ell| = 32 \times 2^k$ and $p_i = \frac{0.2}{2^{i-1}}$, for $i = 1, 2, ..., k$. For each generated instance, our algorithm always recovered the planted hierarchical decomposition.

For a graph G generated from (k, ℓ)-HPPM for $\ell > 2$, even if the probabilities satisfy $p_1 \gg p_2 \gg ... \gg p_{k+1}$, good hierarchical decompositions may not be unique. For example, Figure 6 shows a 160-node graph $G_{1,5}$ generated from the $(1, 5)$-HPPM with $p_1 = 0.2$ and $p_2 = 0.05$. The first level of the hierarchical planted partition is indicated using the colored nodes, each of these five 32-node planted communities having conductance between 0.094 and 0.138. Note that each of the ten 64-node communities found by taking any two of these planted communities has conductance between 0.0712 and 0.096. Therefore, for such an example, there is no

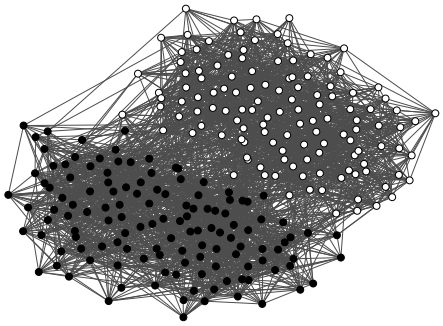

Figure 5: $G_{3,2}$: a 256-node graph generated using the $(3,2)$-HPPM with $p_1 = 0.2$, $p_2 = 0.1$, $p_3 = 0.05$ and $p_4 = 0.025$. The first level of the hierarchical planted partition is indicated using the black and white colored nodes.

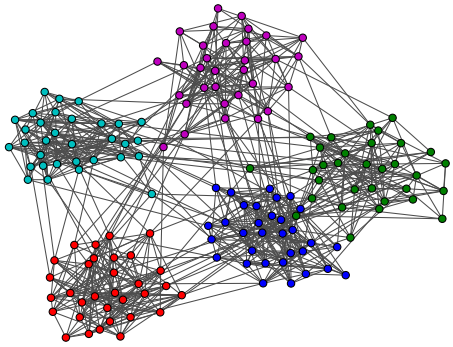

Figure 6: $G_{1,5}$: a 160-node graph generated using the $(1,5)$-HPPM with $p_1 = 0.2$ and $p_2 = 0.05$. The first level of the hierarchical planted partition is indicated using the colored nodes.

obvious unique good hierarchical structure. Specifically, it is unclear whether a flat hierarchy where each of these five planted communities hanging off the root node is superior to alternatives which include some of the larger low-conductance communities in the hierarchy. The trials of our algorithm reflect this by returning different decompositions. In each of these runs our algorithm does not recover the planted flat hierarchy, but instead finding a deeper decomposition which includes some of these larger low conductance communities. One such trial is depicted in Figure 7.

3.3 Real-World Experiments

We consider a set of three real-world networks each of roughly comparable size. We study one social network from Facebook, the popular online networking site. Here each node represents a unique user of the

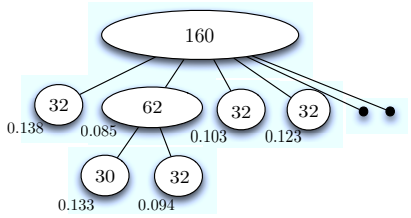

Figure 7: The hierarchical decomposition generated with $\lambda = 5$ and $\delta = 2$. Each of the non-singleton communities is labeled by its size and its conductance score. The singleton communities hanging off of the communities of size 30 and 32 are not shown.

social network and two nodes are connected with a link if the two users are "friends". This network is denoted `FB1`, has 6,116 vertices, 31,374 edges and one connected component, and is an anonymized Facebook dataset derived from [43]. We study two collaboration networks from the SNAP data set [1, 31]. Each of these is built from the e-print service arXiv.org for five different scientific research communities that use this service. Nodes in these networks correspond to authors of papers, and nodes are connected by an edge whenever the corresponding authors co-authored an article. The two research communities are High Energy Physics—Phenomenology (`CA-HepPh`) and General Relativity and Quantum Cosmology (`CA-GrQc`). `CA-HepPh` has 12,008 vertices, 118,521 edges and 278 connected components, where the largest of these contains the vast majority of the vertices, specifically, 11204. `CA-GrQc` has 5,242 vertices, 14,496 edges and 355 connected components, where the largest of these is large, containing 4158 vertices.

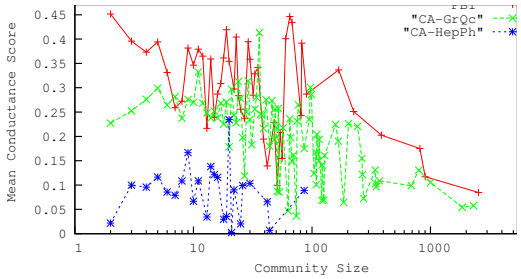

Figure 8: The mean conductance score for all non-singleton communities plotted against community size.

Applying our heuristic with $\lambda = 3$ and $\delta = 1$ to `FB1` gives a rich 10-level decomposition, the average internal node in the decomposition tree having 13.4 children. In addition, the mean conductance of the score is generally high for communities of specific sizes (see

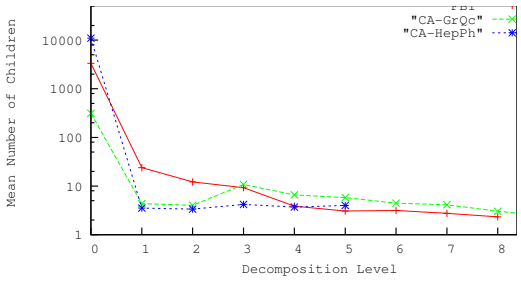

Figure 9: Number of children plotted against decomposition level.

Figure 8). In applying our heuristic with $\lambda = 3$ and $\delta = 1$ to `CA-HepPh` and `CA-GrQc` revealed constrasting behaviour. Both of the collaboration networks seemed to be characterized by stronger smaller communities that are more isolated in general. Indeed, the majority of `CA-HepPh` 12,008 vertices were placed as singletons on the first level of the decomposition. We believe this indicates the the majority of researchers in this community either published very little or collaborated with a small number of coauthors.

4. CONCLUSION

In this paper we formally define a (λ, δ)-HCD to capture the inherent hierarchical structure of a network, where λ parameterizes the connectivity within each subcommunity at the same hierarchical level and δ parameterizes the relationship between communities across two consecutive levels. Strengthening and simplifying Racke's algorithm for oblivious routing, we offer an algorithm that computes an $(O(\log V), O(1))$-HCD for any input graph. This algorithm examines an exponential number of cuts and therefore is inherently inefficient. We propose a heuristic that replaces the exponential enumeration of all cuts by spectral methods. This heuristic performs well on graphs with planted communities attached in a hierarchical fashion. In addition, it is capable of handling complex real-world networks. We remark that this replacement with spectral methods can be thought of as a black box. In fact, any efficient method for finding good communities can be used. For future work, we intend to look into more scalable methods, such as local spectral methods, for even better scaling for larger real-world networks.

5. REFERENCES

[1] Data archive at http://http://snap.stanford.edu/.

[2] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 475–486. IEEE, 2006.

[3] Reid Andersen and Kevin J Lang. Communities from seed sets. In *Proceedings of the 15th international conference on World Wide Web*, pages 223–232. ACM, 2006.

[4] Sanjeev Arora, Rong Ge, Sushant Sachdeva, and Grant Schoenebeck. Finding overlapping communities in social networks: toward a rigorous approach. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 37–54. ACM, 2012.

[5] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 222–231. ACM, 2004.

[6] Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Racke. Optimal oblivious routing in polynomial time. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, STOC '03, pages 383–388, New York, NY, USA, 2003. ACM.

[7] Maria Florina Balcan and Yingyu Liang. Modeling and detecting community hierarchies. *International Workshop on Similarity Based Pattern Analysis and Recognition.*, 9(10):29–30, 2013.

[8] M.F. Balcan, C. Borgs, M. Braverman, J. Chayes, and S.H. Teng. Finding endogenously formed communities. *arXiv preprint arXiv:1201.4899*, 2012.

[9] Ulrik Brandes and Thomas Erlebach. *Network analysis: methodological foundations*, volume 3418. Springer, 2005.

[10] J. Cheeger. A lower bound for the smallest eigenvalue of the Laplacian. Probl. Analysis, Sympos. in Honor of Salomon Bochner, Princeton Univ. 1969, 195-199, 1970.

[11] Fan RK Chung. Spectral graph theory. *CBMS Regional Conference Series in Math, No. 92*, 1997.

[12] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

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

[14] Anne Condon and Richard M Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2):116–140, 2001.

[15] M Cosentino Lagomarsino, P Jona, B Bassetti, and H Isambert. Hierarchy and feedback in the evolution of the escherichia coli transcription network. *Proceedings of the National Academy of Sciences*, 104(13):5516–5520, 2007.

[16] Facebook. https://www.facebook.com/about/graphsearch.

[17] Gary William Flake, Steve Lawrence, and C Lee Giles. Efficient identification of web communities. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–160. ACM, 2000.

[18] Gary William Flake, Robert E Tarjan, and Kostas Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408, 2004.

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

[20] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[21] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[22] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[23] Pablo M Gleiser and Leon Danon. Community structure in jazz. *Advances in complex systems*, 6(04):565–573, 2003.

[24] Dorit S Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1996.

[25] Jianbin Huang, Heli Sun, Jiawei Han, Hongbo Deng, Yizhou Sun, and Yaguang Liu. Shrink: a structural clustering algorithm for detecting hierarchical communities in networks. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 219–228. ACM, 2010.

[26] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM*, 51(3):497 – 515, May 2004.

[27] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[28] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[29] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: A comparative analysis. *Physical review E*, 80(5):056117, 2009.

[30] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, November 1999.

[31] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2, 2007.

[32] Jure Leskovec, Kevin J Lang, and Michael Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web*, pages 631–640. ACM, 2010.

[33] David Lusseau, Karsten Schneider, Oliver J Boisseau, Patti Haase, Elisabeth Slooten, and Steve M Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

[34] Julian McAuley and Jure Leskovec. Learning to discover social circles in ego networks. In *Advances in Neural Information Processing Systems 25*, pages 548–556, 2012.

[35] Karl Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10(1):96–115, 1927.

[36] Nina Mishra, Robert Schreiber, Isabelle Stanton, and Robert E Tarjan. Clustering social networks. In *Algorithms and Models for the Web-Graph*, pages 56–67. Springer, 2007.

[37] H. Racke. Minimizing congestion in general networks. In *In Proc. of the 43rd FOCS*, pages 43 – 52, 2002.

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

[39] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[40] Michael Schweinberger and Tom AB Snijders. Settings in social networks: A measurement model. *Sociological Methodology*, 33(1):307–341, 2003.

[41] Jiří Šíma and Satu Schaeffer. On the NP-completeness of some graph cluster measures. *SOFSEM 2006: Theory and Practice of Computer Science*, pages 530–537, 2006.

[42] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2004.

[43] Christo Wilson, Alessandra Sala, Krishna P. N. Puttaswamy, and Ben Y. Zhao. Beyond social graphs: User interactions in online social networks and their implications. *ACM Trans. Web*, pages 17:1–17:31, 2012.