Abstract—Network clustering enables us to view a complex network at the macro level, by grouping its nodes into units whose characteristics and interrelationships are easier to analyze and understand. State-of-the-art network partitioning methods are unable to identify hubs and outliers. A recently proposed algorithm, SCAN, overcomes this difficulty. However, it requires a minimum similarity parameter ε but provides no automated way to find it. Thus, it must be rerun for each ε value and does not capture the variety or hierarchy of clusters. We propose a new algorithm, SCOT (or Structure-Connected Order of Traversal), that produces a length n sequence containing all possible ε-clusterings and provides an efficient automated method to find the optimal ε. Further, we transform the sequence into a heap and propose a new algorithm, HintClus (or Hierarchy-Induced Network Clustering), to hierarchically cluster the network without any parameter. Our method finds only best cluster boundaries; it is not agglomerative. Results on model-based synthetic network data and real data show that SCOT’s execution time is comparable to SCAN, that BuildContigHeap and HintClus run in negligible time, and that HintClus both matches and exceeds SCAN’s clustering ability, producing excellent clusters with low redundancy in the presence of noise.

I. INTRODUCTION

Large real-world networks, such as social networks, contain pockets of similar nodes called clusters. Network clustering enables us to view complex networks at the macro level, by grouping the raw network into units whose characteristics and interrelationships are easier to analyze and understand. For example, social networks may be organized into social groups, complex biological networks may be grouped into modules to create a biological map [1], and blog entries may be grouped into topical areas.

There are some nodes that we do not consider to be part of any cluster. These nodes have very special properties in real networks. In social networks, for example, hub nodes control the flow of information between communities [2]. These people are vitally essential for their ability to provide the right information at the right time, while outliers represent ineffective individuals. Hub nodes also play an important role in biological networks. Metabolites that connect to different modules are most common across species and if changed would have the greatest impact on the function of the metabolic process [1]. Hubs and outliers in functional genetic networks identify disease genomes [3].

Spectral partitioning methods, such as normalized cut [4], have been popularly applied to network clustering, but these methods use a cutting hyperplane and are not designed to find the optimal number of partitions. More recently, agglomerative hierarchical and probabilistic simulated annealing methods have been proposed [5], [1]. The agglomerative hierarchical algorithm runs in \(O(m d \log n)\) time, where \(m\) is the number of edges, \(n\) is the number of nodes, and \(d\) is the depth of the dendrogram, which is faster than a naive pairwise method having \(O(n^2)\) time. However, these methods cannot identify hubs and outliers, but rather include them in the clusters.

In light of these shortcomings, Xu et al. propose a network connectivity-based algorithm, SCAN [6], by extension of a density-based clustering approach, such as DBSCAN [7]. It uses a structural similarity measure and defines clusters using a minimum cluster size \(\mu\) and a minimum similarity \(\varepsilon\). SCAN runs in average \(O(n)\) time and not only finds clusters, but also identifies outliers and hubs. Nevertheless, there are several difficulties associated with SCAN:

- There is no automated way to find a good \(\varepsilon\).
- The algorithm must be rerun for each \(\varepsilon\).
- Epsilon is a global threshold, which implies
  - There are no hierarchical clusters, and
  - There is no variation in cluster subtlety.

In response, we make the following contributions by developing:

- The algorithm SCOT, which spans a tree over the network in structure-connected order of traversal (SCOT). In average \(O(n)\) time, it produces a sequence of length \(n\) that, we prove, contains all possible \(\varepsilon\)-clusterings by SCAN;
- An efficient method to find the optimal global \(\varepsilon\);
- A new Contiguous Subinterval Heap structure (ContigHeap) of size at most \(n\) that stores the cluster boundary hypotheses obtained from SCOT’s output and is built in \(O(n)\) time; and
- The new, progressive heap-clustering algorithm HintClus, which is epsilon-free and produces hierarchical clusters. The nodes in our cluster hierarchy contain only the optimal cluster boundaries; it is not an agglomerative method.

Progressive Clustering of Networks Using Structure-Connected Order of Traversal

Dustin Bortner 1, Jiawei Han 2

Department of Computer Science
University of Illinois at Urbana-Champaign

1bortner1@illinois.edu 2hanj@illinois.edu
The paper is organized as follows. In Section III, we discuss the concept of structure-connected clusters, introduce structure-connected order of traversal, and present an efficient method to find the optimal global $\varepsilon$. In Section IV, we present the ContigHeap and our algorithm HintClus to cluster without the $\varepsilon$ parameter. Section V contains an empirical analysis of our method, and Section VI concludes and presents ideas for future work.

II. RELATED WORK

The spectral partitioning method introduced by Fiedler [8] divides a network into two sets of nodes such that the number of edges spanning the sets, the edge cut, is minimized. Edges can be weighted to indicate similarity between nodes, in which case the edge cut is the sum of the weights of the spanning edges. A trivial solution is to make no cuts, resulting in one set containing all the vertices and the other set being empty. Adding an additional constraint, such as fixing the cluster sizes, is necessary.

This method has resulted in numerous variants, which seek to minimize some quantity involving edge cut. The ratio cut method [9] minimizes the ratio $K/(|V_1||V_2|)$ where $K$ is the edge cut and $V_1, V_2$ are the sets of vertices. However, this method essentially fixes the cluster sizes, since it is biased toward clusters such that $|V_1| = |V_2|$. The min-max cut method [10] both minimizes the inter-cluster edge cut constraint and maximizes the sum of intra-cluster edge weights. The normalized cut method [4] normalizes the edge cut by the link ratio, the proportion of links to other clusters to total links of a cluster, and seeks to minimize that quantity.

In general, all of the spectral clustering methods partition the nodes into two sets using a cutting hyperplane. To find an arbitrary number $k$ of clusters, spectral clustering is applied recursively. However, it has been shown in [11] that the normalized cut increases monotonically as $k$ increases, and it is biased toward $k = 2$. Thus, the optimal $k$ cannot be determined by comparing the approximate solutions obtained by normalized cut. Recursive spectral partitioning in general does not offer a means to stop recursion.

III. TRAVERSING NETWORKS IN STRUCTURE-CONNECTED ORDER

A. Motivation

Intuitively, for a pair of nodes, we may define similarity as the ratio of shared neighbors to total neighbors. This is known as structural similarity. Clusters vary in cardinality and average similarity, and to use a global similarity threshold is unlikely to accurately find all the clusters. Some will be excluded and others will contain outliers.

Figure 1 shows a network containing two clusters. Cluster $A$ contains fewer nodes and has lower average similarity than cluster $B$, yet we consider both of them to be clusters. Figure 2 exploits the geometric arrangement of the nodes to represent the similarity of the nodes as the height of a surface. A global similarity threshold $\varepsilon$ represents the height of a plane parallel to the $xy$-plane. The value of this parameter will greatly affect the outcome of the clustering, since any contiguous “island” above the plane is considered a single cluster.

Among the non-cluster nodes, we find two types, outlier nodes, which have connections to only one cluster ($\alpha, \beta_1, \beta_2$), and hub nodes, which bridge multiple clusters ($\gamma$). Network partitioning methods in general are not designed to find hubs and outliers, and will assign them to some cluster.

Therefore, we avoid using a partitioning method, and instead turn to connectivity-based methods. To capture clusters with differing degrees of structural similarity using a global connectivity threshold parameter, we would need to run the clustering algorithm many times with different parameter values. However, there is an unlimited number of possible values. If we wish to try a large number of different $\varepsilon$ values to compare the clusterings, we would need to store all of the results from each run, and it is not clear how to combine the results of numerous runs. To solve this problem, we propose an algorithm that produces a sequence of length $n$ that contains all possible $\varepsilon$-clusterings.

Our algorithm produces the sequence by traversing the network nodes according to structural similarity order. If we turned the surface in Figure 2 upside-down, and began filling the space by pouring liquid onto a fixed point, this illustrates how our algorithm traverses the nodes (although we
do not actually utilize any geometric interpretation). We begin at some point and traverse the nodes to a local maximum similarity, where we then iteratively visit all nodes in order of decreasing similarity, essentially generating a spanning tree from any previously visited node. When we reach a saddle point, such as γ, we “spill over”, find the local maximum, and span the nodes in decreasing similarity order. The sequence produced by this traversal order contains clusters at all possible similarity thresholds.

B. Structure-Connected Clusters

The main idea of structure-connected clusters is that for each node in a cluster, the node must have at least μ neighbors whose structural similarity is at least ε. In this subsection, we briefly survey the concepts of structure-connected cluster; see [6] for a full discussion.

Definition 1: Structural Similarity. Let \( G = (V, E) \) be a network. For a node \( v \in V \), the structure \( \Gamma(v) \) of \( v \) is the set containing \( v \) and its neighbors: \( \Gamma(v) = \{ w \in V \mid (v, w) \in E \} \cup \{ v \} \). The structural similarity between two nodes \( v, w \) is then

\[
\sigma(v, w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}.
\]

For a node \( v \), the set of neighbors having structural similarity greater than \( \varepsilon \) forms the \( \varepsilon \)-neighborhood of \( v \): \( N_\varepsilon(v) = \{ w \in \Gamma(v) \mid \sigma(v, w) \geq \varepsilon \} \). If \( |N_\varepsilon(v)| \geq \mu \), then \( v \) is considered a core node in the cluster, denoted \( CORE_{\varepsilon, \mu}(v) \). The fundamental unit of connectivity within a cluster is for a node to be in the \( \varepsilon \)-neighborhood of a core node:

Definition 2: Direct Structure Reachability. A node \( w \) is directly structure reachable from node \( v \) iff \( CORE_{\varepsilon, \mu}(v) \land w \in N_\varepsilon(V) \). This is denoted \( \text{DirREACH}_{\varepsilon, \mu}(v, w) \).

Structure reachability forms the transitive closure of a cluster with respect to direct structure reachability:

Definition 3: Structure Reachability. Given \( \varepsilon \in \mathbb{R}, \mu \in \mathbb{N} \), a node \( w \in V \) is structure reachable from \( v \in V \) iff \( \exists \{u_1, \ldots, u_n\} \subseteq V \) s.t. \( v = u_1, w = u_n \), and \( \forall i \in [1, n - 1], \text{DirREACH}_{\varepsilon, \mu}(u_i, u_{i+1}) \). This is denoted \( \text{REACH}_{\varepsilon, \mu}(v, w) \).

A cluster is coherent because each pair of nodes is structure reachable from some core node in the cluster. We say that the nodes are structure connected.

Definition 4: Structure Connectivity. Nodes \( v, w \in V \) are structure connected iff \( \exists e \in V \) s.t. \( \text{REACH}_{\varepsilon, \mu}(u, v) \land \text{REACH}_{\varepsilon, \mu}(u, w) \). This is denoted \( \text{CONNECT}_{\varepsilon, \mu}(v, w) \).

Finally, a cluster \( C \) is a maximal set of structure-connected nodes:

Definition 5: Structure-Connected Cluster. The set \( C \subseteq V \) is a cluster iff

1. \( \forall v, w \in C, \text{CONNECT}_{\varepsilon, \mu}(v, w) \); and
2. \( \forall v, w \in V, v \in C \land \text{REACH}_{\varepsilon, \mu}(v, w) \Rightarrow w \in C \).

A clustering of a network \( G \) is the set \( \mathcal{G}_{\varepsilon, \mu} \) of distinct structure-connected clusters found in a network, given parameters \( \varepsilon, \mu \). A node cannot belong to more than one cluster. Once a network is clustered, the nodes that do not belong to any cluster are either hubs or outliers.

Definition 6: Hub. Let \( \mathcal{G}_{\varepsilon, \mu} \) be a clustering, given parameters \( \varepsilon, \mu \). A node \( v \in V \) is a hub iff

1. \( v \) is an outlier: \( \forall C \in \mathcal{G}_{\varepsilon, \mu}, v \notin C \); and
2. \( v \) bridges clusters: \( \exists \{u, v\}, \{v, w\} \in E, \exists C, D \in \mathcal{G}_{\varepsilon, \mu} \) s.t. \( u \in C, w \in D \) and \( C \neq D \).

Any node that is not in a cluster and that is not a hub is called an outlier.

C. Structure-Connected Order of Traversal

Recall Figure 2, and observe that regions with higher structural similarity are contained within regions of lower structural similarity. By spanning the nodes in order of decreasing structural similarity, our algorithm SCOT (Structure-Connected Order of Traversal) discovers clusters at each possible \( \mu \) for some \( \varepsilon \). Essentially, we extend SCAN to efficiently calculate local \( \varepsilon \) from the data rather than requiring a global \( \varepsilon \) as a parameter. A local \( \varepsilon \) is found for each node by determining the minimum structural similarity that would make the current node a core node. We call this quantity the core similarity. We then calculate the structural similarity, the reachability similarity, from the current node to each \( \varepsilon \)-neighbor. At each iteration, we choose to visit the new node with the greatest reachability similarity to any previously visited node. This ensures that clusters with higher structural similarity are completely spanned before their surrounding regions of lower structural similarity.

For clarity in this subsection, we distinguish the optional algorithm parameter \( \hat{\varepsilon} \) from the contextual \( \varepsilon \).

Definition 7: Core Similarity. Given a node \( v \in V \) and an optional minimum structural similarity \( \hat{\varepsilon} \in \mathbb{R} \), the core similarity of \( v \) is \( \text{CoreSim}(v) \equiv \max\{\varepsilon \in \mathbb{R} : \{w \in \Gamma(v) : \sigma(v, w) \geq \varepsilon\}\geq \mu\} \mid N_\varepsilon(v)\geq \mu \) UNDEFINED else.

Definition 8: Reachability Similarity. Given nodes \( v, w \in V \) and an optional minimum structural similarity \( \hat{\varepsilon} \in \mathbb{R} \), the reachability similarity of \( w \) with respect to \( v \) is \( \text{ReachSim}(w, v) \equiv \min\{\text{CoreSim}(v), \sigma(v, w)\} \mid N_\varepsilon(v)\geq \mu \) UNDEFINED else.

Our algorithm SCOT is listed in Algorithm 1. The outer loop beginning at line 1 iterates over each node \( v \) in the network, in arbitrary order. EnqueueNeighbors determines whether \( v \)

---

1The parameter \( \hat{\varepsilon} \) is optional because for the default value \( \hat{\varepsilon} = 0 \), \( N_\varepsilon(v) = \Gamma(v) \). We include the parameter to relate SCOT to SCAN.
is a core node, sets its core similarity, and adds \( v \) to the sequence of visited nodes. If \( v \) is a core node, its \( \text{CoreSim}(v) \) -neighbors (neighbors in \( N_{\text{CoreSim}(v)}(v) \)) are added to the priority queue \( \text{visitQueue} \) or their priority therein is updated by calling \( \text{visitQueue}.\text{Update} \). The reachability similarity with respect to \( v \) is calculated and used as the visited. This update ensures that even if a node was previously enqueued, the edge with greatest structural similarity from \( \text{any previously visited} \) node is traversed when the node reaches the head of the queue. Returning to SCOT, if \( v \) is a core node, then the while statement at line 5 iteratively spans the network in structure-connected order of traversal, using the priority queue \( \text{visitQueue} \). Note that execution will not return to line 1 of SCOT unless (a) \( \bar{\epsilon} = 0 \) and the graph is not connected or (b) \( \bar{\epsilon} > 0 \) and this causes some node not to be a core node. If this occurs, any nodes already visited will be skipped, and the next node will be chosen arbitrarily. Therefore, the order of node traversal within the connected subgraphs (which is the entire graph if the graph is connected and \( \bar{\epsilon} = 0 \)) is determined by structure-connected order of traversal, but the order of disconnected subgraphs and the selection of the initial point for each such subgraph is arbitrary.

Figure 3 shows a network containing three clusters, each of which has an outlier, and a hub, which is connected to all three clusters.

D. Representation of Infinite Solutions

We now prove that the length-\( n \) sequence output by SCOT represents the structure-connected clusterings of a network for every possible \( \bar{\epsilon} \). This property allows us to easily generate the equivalent \( \bar{\epsilon} \)-clusterings of SCAN from the compact output of SCOT.

Definition 9: \((\text{Idx}, \text{Sim}, \eta_i)\) Let \( \text{SCOT}_G : V \rightarrow \mathbb{N} \times \mathbb{R} \) denote the algorithm SCOT as a mapping from a node in the network \( G = (V, E) \) to an index and reachability similarity pair. Let us represent \( \text{SCOT}_G \) by a combination of two mappings: \( \text{SCOT}_G : v \mapsto (\text{Idx}_G(v), (\text{Sim}_G \circ \text{Idx}_G)(v)) \), where \( \text{Idx}_G : V \rightarrow \{1, 2, \ldots, |V|\} \) is a bijective mapping from a node to an index, and \( \text{Sim}_G : \{1, 2, \ldots, |V|\} \rightarrow \mathbb{R} \) is a mapping from an index to a reachability similarity value. For brevity, we write \( \eta_i \equiv \text{Idx}^{-1}(i) \), which is a node indexed
by SCOT. We shall assume that we are in the context of some network $G$ and that $\mu$ is fixed, and omit these subscripts.

**Definition 10:** CONTIGUOUS INTERVAL. The interval $[a, b] \subseteq \text{Dom}(\text{Sim})$ is a contiguous interval with respect to $\text{Sim}$ iff $\exists \varepsilon > 0$ s.t. $\text{Sim}(i) \geq \varepsilon, \forall i \in [a, b]$ and $[a, b]$ is maximal. That is, $\exists a' < a$ s.t. $\text{Sim}(i) \geq \varepsilon, \forall i \in [a', a]$ and $\exists b' > b$ s.t. $\text{Sim}(i) \geq \varepsilon, \forall i \in [b, b']$. This is denoted by $\text{Contig}_\varepsilon(a, b)$.

A contiguous interval represents a contiguous, maximal region of the SCOT plot above a horizontal slice.

**Lemma 1:** Given $\varepsilon > 0$, if (a) $\text{Sim}(a-1) < \varepsilon$ or $\text{Sim}(a-1) = \text{UNDEFINED}$ and (b) $\text{Sim}(i) \geq \varepsilon, \forall i \in [a, b]$, then $\text{REACH}_{\varepsilon}(\eta_{a-1}, \eta_{b})$.

**Proof:** $\text{Sim}(b) \geq \varepsilon \Rightarrow \exists k$ s.t. $k < b$ and $\text{DirREACH}_{\varepsilon}(\eta_k, \eta_{b})$. Suppose for contradiction that $k < a-1$. Then SCOT visited $\eta_{a-1}$ while $\eta_k$ was in the queue, which is a contradiction. Thus, $a-1 \leq k < b$. By induction and bounding, $\exists k_1, k_2, \ldots, k_n$ s.t. $b = k_1 > k_2 > \ldots > k_n = a-1$ and $\text{Sim}(i) \geq \varepsilon, \forall i \in [1, n-1]$. By definition, $\text{REACH}_{\varepsilon}(\eta_{a-1}, \eta_{b})$.

**Theorem 2:** Any contiguous interval $\text{Contig}_\varepsilon(a, b)$ corresponds one-to-one to a structurally connected cluster.\(^2\)

That is,

$$C = \{ \eta_i | i \in [a-1, b] \} \wedge \text{Contig}_\varepsilon(a, b) \Leftrightarrow \text{CLUSTER}_\varepsilon(C)$$

**Proof:** ($\Rightarrow$) We must show that $C$ is both connected and maximal.

(Connected) Since $a$ is a maximal lower bound, $\text{Sim}(a-1) = \text{UNDEFINED}$ or $\text{Sim}(a-1) < \varepsilon$. Let $v, w \in C$. By self-reachability and the lemma, $\text{REACH}_{\varepsilon}(\eta_{a-1}, v)$ and $\text{REACH}_{\varepsilon}(\eta_{a-1}, w)$. It follows that $\text{CONNECT}_{\varepsilon}(v, w)$.

(Maximal) Let $v \in C, w \in V$, and $\text{REACH}_{\varepsilon}(v, w)$. Since $\text{Sim}(a-1) = \text{UNDEFINED}$ or $\text{Sim}(a-1) < \varepsilon$, it follows that $\text{Idx}(w) \geq a-1$. Similarly, $\text{Idx}(w) \leq b$. Thus, $w \in C$.

($\Leftarrow$) Let $\exists a_2 \notin C$ and $\eta_{a-1} \in C$. Thus, $\text{Sim}(a-1) = \text{UNDEFINED}$ or $\text{Sim}(a-1) < \varepsilon$. Assume (amendment)

\(^2\)By the definition of structure-connected cluster in \([6]\), cluster membership is ambiguous in the case that the node is not a core node but is directly reachable by core nodes in at least two clusters. In this case, SCAN determines the cluster membership by the arbitrary order in which the nodes are input. Further, SCAN may misclassify a node as an outlier if a non-core node that is directly connected to a core node in exactly one cluster is input before any core node of the cluster. Note that misclassification can only occur when processing the first nodes of a non-connected subgraph and is likely negligible. We therefore assume the definitions to be amended by the original implementation of SCAN.

that $\text{CORE}_{\varepsilon}(\eta_{a-1})$, then $\text{Sim}(a) \geq \varepsilon$, and $a$ is a maximal lower bound. According to Lemma 1 in \([6]\), any core node in $C$ uniquely corresponds to $C$. Since structure-connected order of traversal will span $C$ contiguously and by the lemma, $\forall v \in C, \text{REACH}_{\varepsilon}(\eta_{a-1}, v)$, and so $\forall i \in [a, a-1 + |C|], \text{Sim}(i) \geq \varepsilon$. The last node in $C$ to be visited must be a non-core node or the last node in $V$; otherwise, a future node to be visited would also be in $C$. Thus, $\text{Sim}(a + |C|) < \varepsilon$ or $a - 1 + |C| = |V|$, and $b = a - 1 + |C|$ is the maximal upper bound.

**E. Optimal Global Similarity Threshold**

In \([7]\), \([6]\), a method using sampling and supervised graphical identification is presented to determine the best $\varepsilon$. A $k$-nearest neighbor query is performed for each sampled point. The structural similarities are calculated, and the points are sorted by increasing structural similarity and then plotted. This procedure has computational complexity $O(n d + n \log n)$, where $d$ is the average degree of the nodes, and the optimal $\varepsilon$ then has to be discerned manually. This is in addition to the time to run SCAN, which is done separately. The knee hypothesis stated in \([7]\), \([6]\] asserts that the knee in the data is fixed, and omit these subscripts.

When processing the first nodes of a non-connected subgraph and is likely negligible. We therefore assume the definitions to be amended by the original implementation of SCAN.

The plot is monotonically increasing, and the knee hypothesis implies that the plot is approximately concave. Suppose we select one of the sorted values $\varepsilon_i$, and we draw a line segment from both the minimum and maximum values $\varepsilon_1$ and $\varepsilon_n$ to $\varepsilon_i$. Then the knee is the point at which the obtuse angle between the line segments is minimized, or where the acute angle is maximized. We shall use the following formula to determine the acute angle $\theta$:

$$\tan \theta = \frac{m_1 - m_2}{1 + m_1 m_2},$$

where $m_1 = \frac{\varepsilon_1 - \varepsilon_i}{i}$ and $m_2 = \frac{\varepsilon_n - \varepsilon_i}{n-1}$ are the slopes of the line segments, and we expect that $m_1 > m_2$.

We first present a slower naive method and then improve its complexity. Sorting the data by reachability similarity takes $O(n \log n)$ time. Using the point-to-point method, we have

$$\theta_i = \arctan \left( \frac{(n-i)(\varepsilon_i - \varepsilon_1) - i(\varepsilon_n - \varepsilon_1)}{(n-i) + (\varepsilon_i - \varepsilon_1)(\varepsilon_n - \varepsilon_1)} \right).$$

The last node in $C$ to be visited must be a non-core node or the last node in $V$; otherwise, a future node to be visited would also be in $C$. Thus, $\text{Sim}(a + |C|) < \varepsilon$ or $a - 1 + |C| = |V|$, and $b = a - 1 + |C|$ is the maximal upper bound.

**Fig. 4.** A bar graph of the output of SCOT for Figure 3

- **By the definition of structure-connected cluster in [6], cluster membership is ambiguous in the case that the node is not a core node but is directly reachable by core nodes in at least two clusters. In this case, SCAN determines the cluster membership by the arbitrary order in which the nodes are input. Further, SCAN may misclassify a node as an outlier if a non-core node that is directly connected to a core node in exactly one cluster is input before any core node of the cluster. Note that misclassification can only occur when processing the first nodes of a non-connected subgraph and is likely negligible. We therefore assume the definitions to be amended by the original implementation of SCAN.**

- **That $\text{CORE}_{\varepsilon}(\eta_{a-1})$, then $\text{Sim}(a) \geq \varepsilon$, and $a$ is a maximal lower bound. According to Lemma 1 in [6], any core node in $C$ uniquely corresponds to $C$. Since structure-connected order of traversal will span $C$ contiguously and by the lemma, $\forall v \in C, \text{REACH}_{\varepsilon}(\eta_{a-1}, v)$, and so $\forall i \in [a, a-1 + |C|], \text{Sim}(i) \geq \varepsilon$. The last node in $C$ to be visited must be a non-core node or the last node in $V$; otherwise, a future node to be visited would also be in $C$. Thus, $\text{Sim}(a + |C|) < \varepsilon$ or $a - 1 + |C| = |V|$, and $b = a - 1 + |C|$ is the maximal upper bound.**
We can perform a binary search to find the maximum $\theta_j$ in $O(\log n)$ time. For robustness, we may want to choose several different starting points for the search, and then select the most popular answer as the knee. Unlike, [7], [6], we do not need to count the computation of similarity between the nodes, because unlike their methods that cannot reuse the results, ours has already been done as part of SCOT. Thus, the optimal global $\varepsilon$ can be found automatically in $O(n \log n)$ time.

The complexity of this approach is further decreased to $O(n)$ by using a modified equal-width histogram over the range of reachability similarity values, instead of sorting the nodes. The histogram can be constructed in $O(n)$ time and $O(n)$ space by scanning the nodes and storing the count of nodes having reachability values within each bin, then scanning the histogram and storing the count of nodes below each bin boundary. Since the value of $\text{Sim}$ is in $[0,1]$ and difference between the reachability similarity of non-cluster nodes is relatively large, the number of bins required is much smaller than the number of nodes. This can be directly applied to the point-to-point formula for $\theta_j$. Plus, the binary search is over the number of bins instead of the number of nodes.

IV. PARAMETER-FREE CLUSTERING

In the previous section, we showed that structure-connected order of traversal can be used to generate a length $n$ sequence containing all possible $\varepsilon$-clusterings of SCAN. This allows us to obtain the clusterings for any $\varepsilon$ with little effort, by reusing the SCOT results. In addition, we provided an automated way to find the optimal global $\varepsilon$ under the knee hypothesis.

In this section, we surpass the limitation of a global $\varepsilon$ and find clusterings without any parameter. The $\varepsilon$ of each cluster is independent of others and appropriate for that cluster. In addition, we obtain hierarchical clusters, without producing many redundant clusters. Our technique relies on two novel concepts: (i) transforming the SCOT sequence into a heap of contiguous subintervals, and (ii) using a divide-and-conquer statistical pruning method to identify the clusters satisfying the boundary criterion by progressing both root-to-leaf and leaf-to-root in a single traversal of the heap. Both concepts are extremely fast in practice, relative to SCAN and SCOT execution times, as we show in Section 5.

A. Building the Contiguous Subinterval Heap

Theorem 2 states that each contiguous interval in the SCOT sequence is equivalent to a structure-connected cluster. There exists a natural partial ordering over the set of all contiguous intervals, which is stated in the following definition.

**Definition 11: CONTIGUOUS SUBINTERVAL.** Given a contiguous interval $\text{Contig}_i(a,b)$, the interval $[c,d]$ is a contiguous subinterval of $[a,b]$ iff $[c,d] \subset [a,b]$ and $3\varepsilon' > \varepsilon \text{ s.t. } \text{Contig}_{\varepsilon'}(c,d)$. This is denoted by $[c,d] \prec [a,b]$.

Contiguous subintervals can be stored in a tree structure. The partial ordering of contiguous subintervals makes this structure a heap, which we call a ContigHeap. In this subsection, we construct the ContigHeap, and in the next subsection, we prune the heap to contain only the contiguous subintervals that satisfy the cluster boundary criterion. Algorithm 4 lists our stack-based BuildContigHeap, which builds the ContigHeap in $O(n)$ time. BuildContigHeap can be integrated into SCOT by processing the nodes as they are visited.

The algorithm works as follows. The first item in the sequence becomes a contiguous interval with a lower bound as the current index and an unknown upper bound, and is placed on the stack. As increasing items are read from the sequence, they also become contiguous intervals with lower bounds as the current index and unknown upper bounds and are placed on the stack. If an item with the same $\varepsilon$ as the last item is read, it is ignored. If an item with a lesser $\varepsilon$ than the last item is read, each contiguous interval on the stack with greater $\varepsilon$ is popped, its upper bound is set as the current index, and it is added as a child of the next contiguous interval on the stack. In this way, the stack gradually grows subheaps of the final heap. Once the last item is read from the sequence, all remaining contiguous intervals on the stack are collapsed as descendants to form a single heap.

**Algorithm 4 BuildContigHeap($P$)**

**Input:** The sequence $P$ of reachability values $p_i$

**Output:** A heap of contiguous subintervals ($\text{CS}$).

1: Push $CS \langle \varepsilon = p_1, a = 1, b = \text{UNDEFINED} \rangle$ onto stack $S$
2: for $i := 2$ to $|P|$ do
3: if $S$.peek().$\varepsilon < p_i$ then
4: Push $\langle p_i, i, \text{UNDEFINED} \rangle$ onto $S$
5: else if $S$.peek().$\varepsilon > p_i$ then
6: $current\text{CSReached} := \text{false}$
7: // Close all contained contiguous subintervals
8: repeat
9: if $S$.peek().$\varepsilon = p_i$ then
10: $current\text{CSReached} := \text{true}$
11: else
12: Pop child off of $S$
13: child.$b := i - 1$
14: if $S = \emptyset$ or $S$.peek().$\varepsilon < p_i$ then
15: // Create hypothesis for $p_i$ if not exists
16: Push $\langle p_i, child$.a, \text{UNDEFINED} $\rangle$ onto $S$
17: $current\text{CSReached} := \text{true}$
18: Add child to children of $S$.peek()
19: until $current\text{CSReached}$
20: // Close all open contiguous subintervals
21: while true do
22: Pop child off of $S$
23: child.$b := |P|$
24: if $S = \emptyset$ then return child
25: Add child to children of $S$.peek()
B. Determining Cluster Boundaries

We have shown that it is possible to extract clusters of any threshold $\varepsilon$ from the results of SCOT, but how can we use the data to determine an appropriate local $\varepsilon$ for each cluster? For the DBSCAN algorithm [7], Ankerst et al. [12] provide a clustering method that requires the global parameter $\xi$. This algorithm identifies steep-up and -down regions, and $\varepsilon$ is effectively determined in the regions where $|\text{slope}| \geq \xi$.

Similarly, we argue that local $\varepsilon$ are naturally determined by where the data has a clear boundary, the cluster boundary, at which the change in $\sigma$ has a local extremum. In contrast to [12], however, we shall find the cluster boundaries without a global $\xi$ parameter. By this argument, we make the following definitions.

**Definition 12:** DERIVATIVE of Sim. The derivative of $\text{Sim}(i)$ is

$$\text{Sim}'(i) \equiv \text{Sim}(i) - \text{Sim}(i - 1)$$

**Definition 13:** CLUSTER BOUNDARY CRITERION. A cluster corresponds via SCOT to the interval $[a, b]$ only if $\text{Contig}_c(a, b)$, $\text{Sim}'(a)$ is a local maximum and $\text{Sim}'(b)$ is a local minimum.

Without such extrema, we lack sufficient evidence to distinguish from arbitrary boundaries. This is part of the reason why a global $\varepsilon$ may not produce a good clustering. In practice, $\text{Sim}'(i)$ oscillates due to noise and because the data space is discrete, hence nearly every point of $\text{Sim}'(i)$ is a local extremum. We cannot simply find roots of the second derivative of Sim. Further, we need a criterion to distinguish a “best” $\varepsilon$ within some neighborhood of $i$. By sliding a window over $\text{Sim}'(i)$ we can attempt to limit our focus to local events when evaluating the relative change in $\text{Sim}'(i)$. We can use this technique to validate that the clustering criterion can be satisfied somewhere on the reachability plot, by noting large differences; however, the accuracy and resolution of any clustering result depends on a window size parameter. Nakamura and Small [13] propose the small shuffle surrogate method to identify short-term trends in a time series, but this technique is computationally expensive and experimental. For these reasons, we propose an alternative method that considers the $\varepsilon$ of the nested contiguous subintervals as hypotheses, and prunes the poor hypotheses.

For some network, Figure 5 shows the contiguous subintervals $\text{Contig}_c(a, b)$ found by BuildContigHeap as black lines drawn across the plot from $a$ to $b$ at height $\varepsilon$. In some regions, the $\varepsilon$ of the contiguous subintervals are so close together that they form solid black regions. In this particular plot, the lines are adjacent when the difference between their $\varepsilon$, $\Delta \varepsilon \approx 0.0025$. We observe that this implies that the slope of the plot on both sides of the contiguous subinterval is below some threshold, which depends on the plotted width of the nodes. Therefore, in relation to the cluster boundary criterion, these contiguous subintervals are not bounded at local extrema in $\text{Sim}'$ and are not likely to represent cluster boundaries.

We also see subclusters within larger clusters, and it seems reasonable that these child clusters are independent of their parent clusters, to a degree proportional to the fraction of their width with respect to the parent. This allows child clusters to have cluster boundaries defined by locally maximum $\Delta \varepsilon$ that differ from their parents. In the case that contiguous subintervals form a chain, i.e., when there is a path in the heap along which the widths of the contiguous subintervals are changing very little, then the $\Delta \varepsilon$ in this chain competes for local maximum. This concept of chains provides the basis for the scope in which the cluster boundary criterion is evaluated.

Consider the following weighting scheme to describe the relevance of some ancestor node in determining whether a node is a cluster boundary in the ContigHeap. Let $n_k$ be the node under evaluation. Then the relevance of the parent node $n_{k-1}$ to $n_k$ is given by the ratio of the node’s width to its parent’s width, $w_k/w_{k-1}$. Therefore, if the ratio is high, e.g., 0.95, then $n_{k-1}$ will have a strong relevance in determining whether $n_k$ is a cluster boundary, but if the ratio is low, e.g., 0.40, then $w_{k-1}$ will have less relevance to $n_k$. Further, let the relevance of some ancestor node $n_{k-d}$ with respect to $n_k$ be given by

$$\text{Rel}(w_{k-d}, w_k) \equiv \frac{w_k}{w_{k-1}} \cdots \frac{w_{k-d+1}}{w_{k-d}} = \frac{w_k}{w_{k-d}}$$

for $d < k$. Then the relevance of $n_{k-d}$ to $n_k$ is the product of the pairwise relevances of all intervening node pairs in the path from $n_k$ to $n_{k-d}$. Thus, if all intervening node pairs have high pairwise relevance, then they form a chain and $n_{k-d}$ is highly relevant to $n_k$. However, if any of the intervening node pairs has a low pairwise relevance, then this breaks the chain, for all following nodes will have diminished relevance.

To satisfy the cluster boundary criterion, we define a cluster boundary as a contiguous subinterval having $\Delta \varepsilon$ at least one weighted standard deviation away from the weighted mean $\Delta \varepsilon$, calculated during progression in both the root-to-leaf and leaf-to-root directions, using the relevances as the weights. In the definition of $\text{Rel}$, we see that the iterated products collapse to the ratio of $w_k/w_{k-d}$. Thus the weighted mean and weighted standard deviation in the root-to-leaf direction are

$$m_k \equiv \sum_{i=1}^{k-1} \frac{w_i}{w_k} \Delta \varepsilon_i = \frac{1}{\sum_{i=1}^{k-1} \frac{1}{w_i}} \sum_{i=1}^{k-1} \frac{1}{w_i} \Delta \varepsilon_i$$

and

$$s_k \equiv \sqrt{\sum_{i=1}^{k-1} \frac{w_i}{w_k} (\Delta \varepsilon_i - m_k)^2 = \frac{1}{\sum_{i=1}^{k-1} \frac{1}{w_i}}} \sum_{i=1}^{k-1} \frac{1}{w_i} (\Delta \varepsilon_i - m_k)^2$$

Because $w_k$ divides out of $m_k$ and $s_k$, we can use the running zeroth, first, and second moments to calculate $m_k$ and $s_k$ efficiently for each node in the ContigHeap. A similar argument applies to calculations in the leaf-to-root direction.
Our algorithm HintClus, which determines the cluster boundaries by progressing in both directions, is given in Algorithm 5. We call HintClus($\mu$, $0$, $0$, $0$, ref $ts_0 := 0$, ref $ts_1 := 0$, ref $ts_2 := 0$) on the root node of the ContigHeap, and the algorithm removes all unwanted contiguous subintervals from the ContigHeap. Figure 6 shows the hierarchical cluster boundaries produced by HintClus.

Algorithm 5 CHNode.HintClus($\mu$, $s_0$, $s_1$, $s_2$, $parEps$, ref $cs_0$, ref $cs_1$, ref $cs_2$)

Input: Minimum interesting cluster size $\mu$, root-to-leaf weighted moments $s_i$, $\varepsilon$ of parent node

Output: Leaf-to-root weighted moments $cs_i$ by reference

1: $\Delta \varepsilon := \varepsilon - parEps$
2: $w := b - a$
3: active := $(w + 1 \geq \mu)$
4: if active and $s_0 \neq 0$ then
5:   $m := s_1/s_0$
6:   $s := (1/s_0) \sqrt{s_0s_2 - s_1^2}$
7:   active := $(\Delta \varepsilon - m \geq s)$
8: $s_0 := 1/w$
9: $s_1 := \Delta \varepsilon \cdot s_0$
10: $s_2 := \Delta \varepsilon \cdot s_1$
11: $ts_0 := ts_1 := ts_2 := 0$
12: for all child in children do
13:     child.HintClus($minPts$, $s_0$, $s_1$, $s_2$, ref $ts_0$, ref $ts_1$, ref $ts_2$)
14: if active and $ts_0 \neq 0$ then
15:   $m := ts_1/ts_0$
16:   $s := (1/ts_0) \sqrt{ts_0ts_2 - ts_1^2}$
17:   active := $(\Delta \varepsilon - m \geq s)$
18: $cs_0 := cs_0 + ts_0 + w$
19: $cs_1 := cs_1 + ts_1 + \Delta \varepsilon \cdot w$
20: $cs_2 := cs_2 + ts_2 + \Delta \varepsilon^2 \cdot w$

C. Hubs and Outliers

Unlike SCAN, in which hubs and outliers are simply those nodes with reachability similarity less than the global $\varepsilon$, HintClus produces hierarchical clusters, and hubs and outliers are relative to the clusters in which we are interested. Since HintClus effectively reduces the ContigHeap to only the few best cluster boundaries, and since the clusters vary in $\varepsilon$, it may be inaccurate to globally classify nodes as hubs and outliers. It is reasonable to expect that domain knowledge is required to determine which clusters are interesting, and this will vary depending on the type of data, so we cannot provide the specification for “interesting clusters”. However, HintClus has done the difficult task of producing the best cluster boundaries, so this requirement of the user is more than reasonable.

If a set of non-overlapping cluster boundaries are selected as the interesting primary cluster boundaries, the hubs and outliers are identified among those nodes not included in the selected cluster boundaries. Further, we can provide the ability to select one cluster boundary as the scope for the hub and outlier identification, and only allow the contiguous intervals within this scope to be selected as the primary interesting cluster boundaries. This allows us to discover hubs and outliers in different contexts within the network.

Once the primary cluster boundaries (and optional scope) are selected, we identify hubs and outliers using the following procedure. For each node within the scope and not included in a selected primary cluster boundary, lookup the neighbors of the node. This same lookup is performed by SCAN. Since each node has been assigned an index by SCOT, we can easily determine whether the neighbors are within the scope and whether they are in any of the selected primary cluster boundaries. If the node has neighbors in more than one primary cluster boundary, then the node is a hub. If the node has neighbors in exactly one primary cluster boundary, then the node is an outlier. Otherwise, the node is not connected to any of the primary clusters, is not of interest, and is not classified.

V. EXPERIMENTAL RESULTS

In this section, we compare the execution time of SCOT to SCAN and examine the computational complexity. Using the Enron email dataset, we demonstrate the ability of HintClus to find the hierarchical clusters with various $\varepsilon$ that SCAN cannot find in large, real-world datasets.

A. Computational Complexity

We ran trials of both SCOT and SCAN on generated datasets of three popular network models: Erdős-Rényi (ER) random graph [14], Watts-Strogatz (WS) small-world [15], and Barabási-Albert (BA) scale-free [16] models. For each model, we ran ten trials with increasing numbers of nodes (and edges). We repeated this for different values of $\varepsilon$. The system on which experiments were conducted is a Windows XP Professional PC with a Pentium 4 3.20 GHz CPU and 2 GB RAM.

Figure 7 shows the running times for the ER model ($\mu = 2$). The SCAN series are indicated by the label SN followed by the value of $\varepsilon$, and the SCOT series are indicated by the label ST followed by the value of $\varepsilon$. In each network the number of edges increases linearly with the number of nodes, at a rate of edges $\approx 0.5 \times$ nodes. Execution times of both SCAN and SCOT change negligibly with $\varepsilon$, and execution times scale linearly with the number of edges. Table I lists the relative differences in running times of SCOT relative to SCAN. SCOT’s running time is comparable to SCAN’s, at 3.1% slower for $\mu = 2$ and 12.3% faster for $\mu = 5$. Both BuildContigHeap and HintClus take negligible time, at a maximum time of 0.0156 s (minimum time resolution) over all trials. In most trials, both algorithms took less than the minimum resolution.
Running times for the WS model are shown in Figure 8. In this case, the number of edges in the networks scales non-linearly with the number of nodes, as shown in Figure 9. Given this, we again see that execution times of both SCAN and SCOT scale linearly with the number of edges. SCAN’s execution times change little as \( \varepsilon \) changes, but SCOT’s times are relatively very large for \( \varepsilon = 0 \). However, the running time of SCOT quickly decreases to nearly that of SCAN as \( \varepsilon \) increases. Recall that \( \varepsilon \) determines the minimum reachability similarity; therefore, \( \varepsilon > 0 \) may prevent many points from being enqueued as neighbors in the algorithm. In this case, already at \( \varepsilon = 0.05 \), SCOT’s execution times have converged to a minimum. Since \( \varepsilon \in [0, 1] \) and a negligibly small number of nodes have such a low reachability similarity, at this value of \( \varepsilon \), nearly all of the cluster information in the graph is preserved, and certainly all of the useful cluster information is preserved. SCOT’s execution times are greater than SCAN’s, with a relative difference of 22.5% for 100,000 nodes, \( \mu = 5 \) and \( \varepsilon = 0.1 \). Over all trials, the maximum time for BuildContigHeap was 0.0469 s, and the maximum time for HintClus was 0.0156 s.

Figure 10 shows the execution times for the BA model (\( \mu = 25 \)). In each network, the number of edges increases linearly with the number of nodes, at a rate of edges \( \approx 50 \times \) nodes. While SCAN’s running time slightly increases, SCOT’s running time appreciably decreases, as \( \varepsilon \) increases. In this case, both SCAN and SCOT scale super-linearly, but sub-exponentially, to the number of edges. Notably, SCOT takes less time than SCAN in all trials, and for all tested \( \varepsilon \). At 10,000 nodes, SCOT runs 5.7% faster than SCAN. The maximum time for both BuildContigHeap and HintClus is 0.0156 s, and in most trials was less than the minimum resolution. The BA model results are particularly significant, as the large real-world datasets we analyzed have this model’s fat-tail distribution.

Overall, the results in this subsection indicate that SCOT competes with SCAN in terms of execution time. SCOT has the same computational complexity as SCAN. Recall that the \( \varepsilon \) parameter behaves differently for SCOT than for SCAN. For SCOT, it is simply the minimum reachability similarity in the output sequence; all \( \varepsilon \)-clusterings above this minimum can still be extracted. For SCAN, \( \varepsilon \) determines the single \( \varepsilon \)-clustering that is found. Given that SCOT contains all \( \varepsilon \)-clusterings and that the maximum relative difference in execution time with respect to SCAN was 29.3%, if SCAN must be executed
multiple times with different $\varepsilon$ values to find the clusters, then SCOT has already greatly reduced the time for analysis. Further, the finding that SCOT runs faster than SCAN for the BA model is encouraging, since we observe that the large real-world datasets resemble this model. Finally, we see that BuildContigHeap and HintClus run in negligible time.

B. Cluster Evaluation

To evaluate the clustering quality of HintClus, we use the Enron email dataset [17]. This dataset contains 517,431 email messages, as files in the original folder structures, from 150 employees of Enron Corp. The From and To email addresses were extracted from the 1.7 GB of files, filtered to remove garbage characters and addresses used only once, and assigned IDs. The resulting network consists of 56,985 nodes (email addresses) and 219,368 edges.

1) General Clustering Behavior: For $\mu = 30$ and $\varepsilon = 0$, SCOT executes in 68 s. (We select $\mu = 30$ because it produces a smoother sequence, which provides easier explanation here. However, we find similar execution times for $\mu = 2$ and $\mu = 5$.) BuildContigHeap and HintClus execute in less than the minimum time resolution. Figure 11 shows the SCOT sequence plot with HintClus clusters indicated as horizontal black lines for the Enron email dataset. The nodes are distributed with a fat tail, along which we encounter spikes whose base $\varepsilon$ decreases with the tail. Many of these spikes correspond to clusters, depending on whether there is a corresponding node in the ContigHeap that satisfies the cluster boundary criterion.

If we had plotted all contiguous subintervals active in the original ContigHeap, the plot would look almost solid black. HintClus appropriately eliminates the contiguous subintervals throughout the body of the plot, since the overall slope of the tail does not satisfy the cluster boundary criterion. This is a feat, considering the variety and number of the $\Delta \varepsilon$ throughout the entire plot and the fact that HintClus runs in negligible time. Figure 12 shows the SCOT plot and HintClus clusterings of the first 3,100 nodes from Figure 11. Here, it is easier to see that the spikes actually contain very irregular features. HintClus is able to eliminate those contiguous subintervals that do not meet the cluster boundary criterion. HintClus has also produced hierarchical clusters. Close inspection of the HintClus clusterings reveals that they do correspond with intuitively sharp changes in the steepness of the plot, relative to the surrounding regions. In terms of $\varepsilon$, the closeness of the clusters within a path of the ContigHeap is commensurate with the length of the chain. HintClus is able to capture both hierarchy and variety in the clusters, while finding only the best cluster boundaries. In contrast, SCAN would have to be run many times to produce these clusters. By arbitrarily selecting the $\varepsilon$ of one of the HintClus clusters, we have $\varepsilon \approx 0.297$. At this value, SCAN executes in 44 s, already more than half the execution time of SCOT, which means an explosion in the analysis time using SCAN in comparison to SCOT. Further, without the SCOT sequence, it would be infeasible to find the same optimal hierarchy and cluster variety.

2) Clusters Discovered: To illustrate the legitimacy of the clusters discovered by HintClus, we select five small, non-overlapping clusters and analyze the relationship of the nodes using the contents of the emails sent between the users in that cluster.

The first cluster contains 23 individuals, and a sampling of 10 of their email addresses are listed in Table II. Using email signatures, topics under discussion, and references to new hires and resignations, we determined that the individuals are in the Houston, Texas office at 1400 Smith St. They are legal staff, most of whom work in the Enron Wholesale Services (EWS) department, and some of whom work in the Enron...
Cluster 2 consists of 18 members of the marketing staff in the Enron Transportation Services (ETS) department of the Omaha office. Some of these individuals fall on the technical side of marketing, working on software projects, such as the ETS Marketing Dashboard, Pipeline Profile Pricing View or LIM database. Others work on the financial side of marketing, and consume these software resources in analyzing natural gas pipeline throughput at various storage sites, including Transwestern (TW).

The third cluster contains 21 individuals with non-enron.com email addresses. These are Pacific Gas and Electric (PG&E) legal staff and legal consultants corresponding on pending adjudication, such as the extension of the “Gas Accord II” period, during which PG&E would use the California Gas Transmission (CGT) Risk Management Program. Other matters include Gas Industry Restructuring (GIR) settlement, the gas strategy investigation by the California Public Utilities Commission (CPUC) that lead to natural gas rate increases for PG&E customers, and a complaint filed by Enron against PG&E regarding tariff violations.

Cluster 4 is comprised of 31 researchers employed by Enron at the Houston office. The members are mainly split between the Enron Research Group (ERG), the Weather Risk Management Group, and the Enron Asset Management (EAM) group. Interestingly, of employees that resigned from Enron in 2001/2002, emails containing formal resignations and providing information about new employment appeared most prominently by researchers.

Finally, the fifth cluster contains 33 individuals related to the GE Wind Energy (GEWE) Systems Performance Group. Discussions generally involve wind turbine performance at various client sites, such as whether turbine uptime is measured correctly when turbines are in virtual mode for extended periods of time, and claims by clients that they are not billed properly and that preventive maintenance is not being performed on the turbines as agreed.

There are many other clusters identified by HintClus, and these examples illustrate that the clusters have legitimate
intracluster similarity and intercluster dissimilarity. They also demonstrate that one can learn interesting aspects of the operation of various Enron departments and groups, using the clusters to easily find related emails.

VI. CONCLUSION AND FUTURE WORK

By traversing the network in structure-connected order, SCOT is able to represent all of SCAN’s clusterings for any value of $\varepsilon$ in one length $n$ sequence. In effect, we span a tree over the network by choosing an arbitrary starting point and then iteratively selecting the most similar node to any previously visited node. Using the SCOT sequence, we presented an efficient, automated method to determine the optimal global $\varepsilon$. Unlike the sampling method, we are able to use the reachability similarities of all nodes, and by using a modified histogram instead of sorting, we are able to find the knee in $O(n)$ time.

We transformed the SCOT sequence into a ContigHeap, a heap of contiguous subintervals, in which each node represents a structure-connected cluster for a particular $\varepsilon$. The ContigHeap is constructed in $O(n)$ time. We presented the algorithm HintClus, which traverses the heap to prune invalid cluster boundaries without a similarity threshold $\varepsilon$ or a steepness threshold $\xi$. This method produces hierarchical clusters, which capture the subtlety and variety of the clusters.

Using three popular network models, we compared the computational complexity of SCOT and SCAN, and found them to be identical. With the Enron dataset, we demonstrated the hierarchical and varied clusters produced by HintClus, and explained why it is infeasible to replicate such results with SCAN. In all experiments BuildContigHeap and HintClus ran in negligible time. We showed visually that the clusters satisfy the cluster boundary criterion, and thus are best cluster boundaries. Finally, we demonstrated the legitimacy of the HintClus clusters by analyzing a subset of the clusters and observing intracluster similarity.

In future, we would like to investigate relaxing the definition of hubs so that clusters may be connected through paths of multiple nodes that may be considered hubs. Since SCOT has identified the maximally structurally connected nodes, it may be possible to use the clustering found by HintClus to propagate beginning at the maximal nodes to find where clusters meet. Under certain conditions, these meeting points can be considered hubs.

ACKNOWLEDGEMENT

The work was supported in part by the U.S. National Science Foundation grants IIS-08-42769 and BDI-05-15813, Office of Naval Research (ONR) grant N00014-08-1-0565, and the Air Force Office of Scientific Research MURI award FA9550-08-1-0265. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES