Scalable Hierarchical Agglomerative Clustering

Nicholas Monath[2], Kumar Avinava Dubey[1], Guru Guruganesh[1], Manzil Zaheer[1],
Amr Ahmed[1], Andrew McCallum[2], Gokhan Mergen[1], Marc Najork[1],
Mert Terzihan[4], Bryon Tjanaka[3], Yuan Wang[1], Yuchen Wu[1]
[1] Google LLC [2] University of Massachusetts Amherst
[3] University of Southern California [4] Facebook

ABSTRACT

The applicability of agglomerative clustering, for inferring both hierarchical and flat clustering, is limited by its scalability. Existing scalable hierarchical clustering methods sacrifice quality for speed and often lead to over-merging of clusters. In this paper, we present a scalable, agglomerative method for hierarchical clustering that does not sacrifice quality and scales to billions of data points. We perform a detailed theoretical analysis, showing that under mild separability conditions our algorithm can not only recover the optimal flat partition but also provide a two-approximation to non-parametric DP-Means objective [32]. This introduces a novel application of hierarchical clustering as an approximation algorithm for the non-parametric clustering objective. We additionally relate our algorithm to the classic hierarchical agglomerative clustering method. We perform extensive empirical experiments in both hierarchical and flat clustering settings and show that our proposed approach achieves state-of-the-art results on publicly available clustering benchmarks. Finally, we demonstrate our method's scalability by applying it to a dataset of 30 billion queries. Human evaluation of the discovered clusters show that our method finds better quality of clusters than the current state-of-the-art.

CCS CONCEPTS

• **Computing methodologies → Cluster analysis**.

KEYWORDS

Clustering, Hierarchical Clustering

ACM Reference Format:
Nicholas Monath, Kumar Avinava Dubey, Guru Guruganesh, Manzil Zaheer, Amr Ahmed, Andrew McCallum, Gokhan Mergen, Marc Najork, Mert Terzihan, Bryon Tjanaka, Yuan Wang, Yuchen Wu. 2021. Scalable Hierarchical Agglomerative Clustering. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467404

Work done while NM and BT were interns at Google. Work done while MT was at Google. Corresponding author emails: nmonath@cs.umass.edu and avinava.dubey@google.com.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08.
https://doi.org/10.1145/3447548.3467404

1 INTRODUCTION

Clustering is widely used for analyzing and visualizing large datasets (e.g, single cell genomics [52] users in social networks [7]), for solving tasks such as entity resolution [15, 26, 61], and for feature extraction (in debating systems [19] and knowledge base completion [17]). While it is the case that many clustering tasks are NP-hard [18, 43] and impossible to satisfy three simple properties [35], clustering is widely used and beneficial to the aforementioned applications in practice.

Hierarchical clustering, in which the leaves correspond to data points and the internal nodes correspond to clusters of their descendant leaves, can be useful to represent clusters of multiple granularity [60] or to automatically discover nested structures [16, 23]. Hierarchical clusterings represent multiple alternative *tree consistent partitions* [31]. Each tree consistent partition is a set of internal nodes that correspond to a flat clustering of the dataset. This illustrates how hierarchical clusterings can be used to represent uncertainty about a candidate flat clustering. Hierarchical clustering methods are often used to produce flat clustering, with a partition selected from relevant nodes from the tree structure. This is commonly done in entity resolution [26, 61]. Extracting a flat clustering from a hierarchical clustering, rather than directly performing flat clustering, has been show to be empirically [26, 37] as well as theoretically [28] beneficial. The hierarchical structure has also has proved useful in interactive settings where user feedback is provided to improve and extract a flat clustering [36, 54].

Best-first, bottom-up, hierarchical agglomerative clustering (HAC) is one of the most widely-used clustering algorithms [20, 26, 42, 52]. It is used as the basis for inference in many statistical models [9, 30, 31], as an approximation algorithm for hierarchical clustering costs [18, 47] as well as for supervised clustering [34, 55]. Interestingly, the hierarchical clustering algorithm has also been shown to be effective for flat clustering both theoretically in terms of K-means costs [28] as well as empirically [26, 37]. One capability that contributes significantly to HAC's prevalence is that it can be used to construct a clustering according to any cluster-level scoring function, also known as a *linkage function* [34, 55].

A key challenge in hierarchical clustering is scalability. For example, HAC takes $O(N^2 \log(N))$ time for N points. Competing methods attempt to achieve better scalability by operating in an online manner [45]. These incremental/online algorithms, while often effective empirically, are inherently sequential and so cannot utilize parallelism or scale to datasets larger than a few million points [45]. On the other hand, randomized algorithms [30] and parallel/distributed methods [6] typically achieve scalability at the cost of accuracy.

Affinity clustering [6], overcomes the main computational expense in HAC algorithm by leveraging distributed connected component algorithms. While efficient and a state-of-the-art method, Affinity clustering suffers from over-merging clusters as we empirically observe in this work.

In this paper, we design an accurate and scalable, bottom-up hierarchical clustering algorithm, the *Sub-Cluster Component Algorithm (SCC)*. We provide a detailed theoretical and empirical analysis of in terms hierarchical clustering as well as flat clustering (by selecting the partition from a particular round of the algorithm). We make the following contributions:

Theoretical Contributions (§3)

- SCC produces hierarchies containing the optimal flat partition for data that satisfies δ-separability [41] and achieves a constant factor approximation of *DP-means objective* [11], a flexible objective for flat clustering which adapts to different numbers of clusters. This is to our knowledge the first use of hierarchical clustering to provide an approximation algorithm for this non-parametric objective
- SCC generalizes HAC (in limit we can recover the same tree as HAC). We also show that SCC can recover the target partition of model-based separated data [45], which is recoverable by HAC.

Empirical Highlights (§4 & §5)

- State-of-the-art hierarchical and flat clustering results on publicly available benchmark datasets.
- Produces lower-cost clusterings in terms of DP-Means[49] than competing state-of-the-art methods.
- Web scale experiments examining the coherence of the clusters discovered by our algorithm, on **30 billion user queries**. Human evaluation shows SCC produced more coherent clusters than Affinity clustering. To the best of our knowledge, this is the largest evaluation of clustering algorithms, there by showcasing the scalability of SCC.

2 SUB-CLUSTER COMPONENT ALGORITHM

In this section, we first formally define notation and then describe our proposed SCC algorithm.

2.1 Definitions

Given a dataset of points $\boldsymbol{X} = \{x_i\}_1^N$, a flat clustering or partition is a set of disjoint subsets. Formally:

Definition 1. **[Flat clustering]**. *A flat clustering or partition of $\boldsymbol{X}$, denoted $\mathbb{S} = \{C_1, \ldots, C_K\}$, of $\boldsymbol{X}$, is a set of disjoint (and non-empty) subsets (i.e., $C_i \cap C_j = \emptyset$, $\forall C_i \neq C_j$) that covers $\boldsymbol{X}$ (i.e., $\bigcup_{i=0}^K C_i = \boldsymbol{X}$).*

We refer to the set of all partitions of a set $\boldsymbol{X}$ as $\mathcal{P}(\boldsymbol{X})$. Each flat clustering is a member of this set, $\mathbb{S} \in \mathcal{P}(\boldsymbol{X})$. A hierarchical clustering is a recursive partitioning of dataset $\boldsymbol{X} = \{x_i\}_1^N$ into a tree-structured set of nested partitions $\mathcal{T}$. Formally:

Definition 2. **[Hierarchical clustering [39]]**. *A hierarchical clustering, $\mathcal{T}$, of a dataset $X = x_1, x_2, \ldots, x_N$, is a set of clusters $C_0 = \{x_i\}_{i=1}^N$ and for each $C_i, C_j \in \mathcal{T}$ either $C_j \subset C_k$, $C_k \subset C_j$ or $C_j \cap C_k = \emptyset$. For any cluster $C \in \mathcal{T}$, if $\exists C'$ with $C' \subset C$, then there exists a set $\{C_j\}_{j=1}^\ell$ of disjoint clusters such that $\bigcup_{i=1}^\ell C_j = C$.*

Equivalently, a hierarchical clustering can be thought of as a tree structure such that leaves correspond to individual data points and internal nodes represent the cluster of their descendant leaves. In this work, we will refer to nodes of the tree structure by the cluster of points that the node represents, C_i, as in Definition 2. Parent/child edges in the tree structure can be inferred using this cluster-based notation. A hierarchical clustering encodes many different flat clusterings, known as *tree consistent partitions* [31]. A tree consistent partition is a set of internal nodes $\{C_1, \ldots C_K\} \subset \mathcal{T}$ that form a flat clustering.

Following HAC and previous work [6, 45], our algorithms will make use of *linkage functions*, which measure the dissimilarity of two sets of points: $d : \mathcal{P}(\boldsymbol{X}) \times \mathcal{P}(\boldsymbol{X}) \rightarrow \mathbb{R}^+$. Linkage functions are quite general in that they can support any function of the two sets [34, 42, 45]. Many commonly used linkage functions are defined in terms pairwise dissimilarities between points. For instance, the well-known *average* linkage is the average pairwise dissimilarities between points of one set to the other and *single* is the minimum pairwise dissimilarity. We will show how particular linkage functions can be used to achieve theoretical guarantees about our algorithm's performance.

2.2 Proposed Approach

SCC works in a best-first manner: determining which points should belong together in clusters in a sequence of rounds. The sequence of rounds begins with the decisions that are "easy to make" (e.g., points that are clearly in the same cluster) and prolongs the later, more difficult decisions until these confident decisions have been well established. SCC starts by putting each point into its own separate cluster. Then in each round, we merge together groups of clusters from the previous round that satisfy a given certain "condition". The merging operation continues, until there are no pairs of clusters remaining to be merged. Each round of our algorithm produces a partition (flat clustering) of the dataset at a different granularity and the collection of rounds together forms a hierarchical clustering (with non-parametric branching factor).

Let $\mathbb{S}^{(i)}$ be the partition produced after round i and the partition at the starting round be $\mathbb{S}^{(0)} = \{\{x\} | x \in \boldsymbol{X}\}$. Let a *sub-cluster* refer to a member a partition, i.e. $C \in \mathbb{S}^{(i)}$. Let $\tau_1, \ldots \tau_L$ be a series of L predefined increasing thresholds, given as hyperparameters to the algorithm. To specify the "condition" under which we merge sub-clusters in each round, we define *sub-cluster component* as:

Definition 3. **[Sub-cluster Component]** *Two sub-clusters $C_j, C_k \in \mathbb{S}$ are defined to be part of the same* sub-cluster component *according to a threshold τ and linkage $d : \mathcal{P}(\boldsymbol{X}) \times \mathcal{P}(\boldsymbol{X}) \rightarrow \mathbb{R}^+$, denoted $CH_d(C_j, C_k, \tau, \mathbb{S}) = 1$, if there exists a path $P \subseteq \mathbb{S}$ defined as $\{C_j = C_{s_0}, C_{s_1}, C_{s_2}, \ldots C_{s_{R-1}}, C_{s_R} = C_k\}$, where each the following two conditions are met:*

(1) $d(C_{s_r}, C_{s_{r-1}}) \leq \tau$ for $0 \leq r \leq R$, and
(2) either $C_{s_{r-1}} = \text{argmin}_{C \in \mathbb{S}} d(C_{s_r}, C)$ and/or $C_{s_r} = \text{argmin}_{C \in \mathbb{S}} d(C_{s_{r-1}}, C)$.

Inference at round i works by merging the sub-clusters in round $i-1$ that are in the same *sub-cluster component*. Computationally,

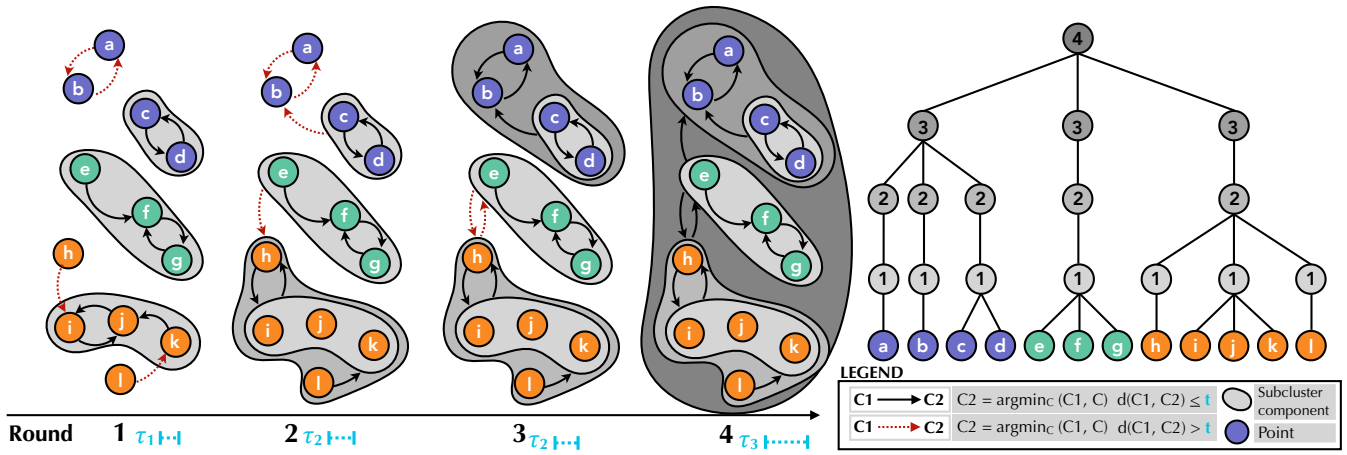

Figure 1: The Sub-Clustering Component Algorithm. We illustrate SCC on a small dataset. The formation of sub-clusters is shown with black arrows for pairs of points satisfying Def. 3. The direction indicates indicates the nearest neighbor relationship (Def. 3, condition 2). Red edges indicate the nearest neighbor relationships that are above the distance thresholds. The grey circles indicate the sub-cluster components created in that round. Best viewed in color.

the construction of sub-cluster components can be thought of as the connected components of a graph with nodes as the sub-clusters from the previous round and edges between pairs of nodes that are nearest neighbors and are less that τ from one-another.

We define, $\text{SC}_d(C_j, \mathbb{S}, \tau)$, as the union of all sub-clusters in $\mathbb{S}$ that are within the sub-cluster component of C_j, i.e.,

$$\text{SC}_d(C_j, \mathbb{S}, \tau) := \bigcup_{\substack{C \in \mathbb{S}, \\ \text{CH}_d(C_j, C, \tau, \mathbb{S})=1}} C \tag{1}$$

Thus, $\text{SC}_d(C_j, \mathbb{S}^{(i-1)}, \tau^{(i)})$ is a new cluster, created by taking a union of all clusters from round $i-1$ that are in the sub-cluster component of C_j. We create the flat partition at round i, $\mathbb{S}^{(i)}$, as the set of all of these newly found clusters:

$$\mathbb{S}^{(i)} := \{\text{SC}_d(C, \mathbb{S}^{(i-1)}, \tau^{(i)}) \mid C \in \mathbb{S}^{(i-1)}\} \tag{2}$$

We refer to our algorithm as the **Sub-Cluster Component algorithm (SCC)**. Alg. 1 gives pseudocode for SCC. We only increment the threshold if no clusters are merged in the previous round i.e. $\mathbb{S}^{(i-1)} = \mathbb{S}^{(i)}$. The sub-cluster componentw in a particular round can be found efficiently using a connected components algorithm [10]. Any of the rounds can be used as a predicted flat clustering. A hierarchical clustering is given by $\bigcup \text{SCC}(\boldsymbol{X}, d, \{\tau_1, \ldots, \tau_L\})$, the union of the sub-clusters produced by all rounds. Figure 1 provides an illustration of the SCC algorithm and the sub-cluster formation.

3 ANALYSIS

SCC is a simple, intuitive algorithm for clustering that has a number of desirable theoretical properties. We provide theoretical analysis of SCC used for both flat and hierarchical clustering. We analyze the separability conditions under which SCC recovers the target clustering and connect these results to the DP-Means objective as well as hierarchical clustering evaluation measures. Lastly, we show that in the limit of number of rounds our method will produce the same tree structure as agglomerative clustering.

Algorithm 1 Sub-Cluster Component Alg. (SCC)

Input: $\boldsymbol{X}$: dataset , d: set dissimilarity, $\{\tau_1, \ldots, \tau_L\}$: a set of thresholds in increasing order
Output: $(\mathbb{S}^{(0)}, \mathbb{S}^{(1)}, \ldots)$: One flat partition per round
$\mathbb{S}^{(0)} \leftarrow \{\{x\} \mid x \in \boldsymbol{X}\}$
idx $\leftarrow 1$, $i \leftarrow 1$
while idx $< L$ **do**
  Set $\text{SC}_d(C_i, \mathbb{S}^{(i-1)}, \tau^{(i)}), \forall C_i \in \mathbb{S}^{(i-1)}$, (Eq. 1)
  Set $\mathbb{S}^{(i)}$ (Eq. 2)
  idx $\leftarrow$ idx $+ \mathbb{I}[\mathbb{S}^{(i)} = \mathbb{S}^{(i-1)}]$
  $i \leftarrow i + \mathbb{I}[\mathbb{S}^{(i)} \neq \mathbb{S}^{(i-1)}]$
  $\tau^{(i)} \leftarrow \tau_{\text{idx}}$
return $(\mathbb{S}^{(0)}, \ldots, \mathbb{S}^{(i-1)})$

3.1 Recovering Target Clustering

Separability assumptions in clustering provide a mechanism to understand whether or not an algorithm effectively and efficiently recovers cluster structure under "reasonable" conditions. If we can define a *center* for a cluster of points, then we can define δ-separability, which expresses a ratio between the center-to-center dissimilarities and the point which is farthest from its assigned center.

Assumption 1. **(δ-Separability [41])** *We say that the input data $\boldsymbol{X}$ satisfies $\delta-$separation, with respect to some target clustering $\mathbb{S}^\star = \{C_1, C_2, \ldots, C_k\}$ if there exists centers $c_1^\star, \ldots c_k^\star$ such that for all $i \neq j$ $||c_i^\star - c_j^\star|| \geq \delta \cdot R$ where $R := \max_{l \in [k]} \max_{x \in C_l} ||x - c_l^\star||$.*

Each round of SCC produces a flat clustering of a dataset $\boldsymbol{X}$. We will show that if $\boldsymbol{X}$ satisfies the above δ-separability assumptions, one of the rounds of SCC will in fact be equal to the target clustering for the dataset, $\mathbb{S}^\star$, corresponding to the separated clusters. Formally, we make the statement:

Theorem 1. *Suppose the dataset $\boldsymbol{X}$ satisfies the δ-separability assumption with respect to the target clustering $\mathbb{S}^\star = \{C_1^\star, \ldots, C_k^\star\}$*

for $\delta \geq \gamma$. $SCC(\boldsymbol{X}, d, \{\tau_0, \ldots, \tau_L\})$ is set of partitions produced by SCC (Alg. 1) with $d(\cdot,\cdot)$ as average linkage and geometrically increasing thresholds i.e. $\tau_i = 2^i \cdot \tau_0$. The target clustering is equal to one of the clustering produced by one round of SCC, $\mathbb{S}^\star \in SCC(\boldsymbol{X}, d, \{\tau_0, \ldots, \tau_L\})$, where $\gamma = 6$ for all metrics and $\gamma = 30$ for the ℓ_2^2 distance and $\tau_0 \leq \min_{x,x' \in \boldsymbol{X}^2} ||x - x'||$.

The proof of Theorem 1 is given in the supplemental material (§A.1). Intuitively, we prove that, for the aforementioned bounds on within/across cluster distances, a geometric series will include a threshold that is larger than largest within cluster distance and smaller than the closest across cluster distance between any two sub-clusters. Having such a threshold $\tau^\star$, we will have a round i with a flat clustering, $\mathbb{S}^{(i)}$, equal to the target clustering $\mathbb{S}^\star$, $\mathbb{S}^{(i)} = \mathbb{S}^\star$.

We also consider a more general class of separable data, model-based separation [45], which specifies when a particular linkage function "separates" a dataset. In model-based separation, we view the dataset $\boldsymbol{X}$ as the nodes in an undirected graph with latent (unobserved) edges. The edges of this graph provide connected components, which correspond exactly to a target clustering $\mathbb{S}^\star$.

Assumption 2. **(Model-based Separation [45])** *Let $G = (\boldsymbol{X}, E)$ be a graph. Let the function $d : \mathcal{P}(\boldsymbol{X}) \times \mathcal{P}(\boldsymbol{X}) \to \mathbb{R}$ be a linkage function that computes the similarity of two groups of vertices and let $g : \mathcal{P}(\boldsymbol{X}) \times \mathcal{P}(\boldsymbol{X}) \to \{0, 1\}$ be a function that returns 1 if the union of its arguments is a connected subgraph of G. Dataset $\boldsymbol{X}$ is model-based separated with respect to f if:*

$$\forall C_0,\ C_1,\ C_2 \subseteq \boldsymbol{X},\ g(C_0, C_1) > g(C_0, C_2) \implies d(C_0, C_1) < d(C_0, C_2).$$

The target partition, $\mathbb{S}^\star$, which is model-based separated, corresponds to connected components in G.

We show that for datasets that are model separated our algorithm will contain the target clustering, i.e.,

Proposition 1. *Given a dataset $\boldsymbol{X}$ and a symmetric injective linkage function f s.t. $\boldsymbol{X}$ is model-based separated with respect to f, let $\mathbb{S}^\star$ be the target partition corresponding to the separated data. There exists a $\tau_1, \ldots, \tau_L$ such that the hierarchical clustering $\bigcup SCC(\boldsymbol{X}, \{\tau_1, \ldots, \tau_L\}, d)$ discovered by SCC contains the target partition $\mathbb{S}^\star$.*

Please refer to the supplemental material for the proof (§A.2).

3.2 Relation to Nonparametric Clustering

Next, we analyze the performance of our algorithm with respect to nonparametric, flat clustering cost functions. Nonparametric clustering, where the number of clusters is not known *a priori* and must be inferred from the data, is useful for many clustering applications [1, 32, 40]. DP-means [11, 32, 40] is an example of a widely used nonparametric cost function, that is obtained from the small variance asymptotics of Dirichlet Process mixture models.

Definition 4. **[DP-Means [32]]** *Given a dataset $\boldsymbol{X}$, a partition $\mathbb{S} = \{C_1, \ldots, C_K\}$, such that cluster C_l has center c_l and hyperparameter λ, the* DP-Means *objective is:*

$$DP(\boldsymbol{X}, \lambda, \mathbb{S}) = \sum_{C_l \in \mathbb{S}} \sum_{x \in C_l} ||x - c_l||^2 + \lambda |\mathbb{S}|. \tag{3}$$

Given a dataset $\boldsymbol{X}$ and hyperparameter λ, clustering according to DP-Means seeks to find: $\operatorname{argmin}_{\mathbb{S}, \boldsymbol{c}} DP(\boldsymbol{X}, \lambda, \mathbb{S})$ *where $\boldsymbol{c}$ are the centers for each cluster in $\mathbb{S}$.*

We show that SCC yields a constant factor approximation to DP-Means solution under δ-separation (Assumption 1).

Theorem 2. *Suppose the dataset $\boldsymbol{X}$ satisfies the δ-separability assumption with respect to the target clustering $\mathbb{S}^\star = \{C_1^\star, \ldots, C_k^\star\}$ for $\delta \geq \gamma$. $SCC(\boldsymbol{X}, d, \{\tau_0, \ldots, \tau_L\})$ is set of partitions produced by SCC with $d(\cdot,\cdot)$ as the average distance between points and geometrically increasing thresholds i.e. $\tau_i = 2^i \cdot \tau_0$. $SCC(\boldsymbol{X}, d, \{\tau_0, \ldots, \tau_L\})$ contains a 2-approximation solution to the DP-Means objective.*

See Appendix §A.1 for the proof. The two primary steps are: (1) Using theorem 1 and show that SCC can find optimal solution to the *facility location* problem and (2) show that this solution is a constant factor approximation of the DP-means solution.

Our analysis of SCC as an approximation algorithm for DP-Means shows the first connection of hierarchical bottom up clustering to non-parametric flat clustering objectives such as DP-means. This theoretical connection is also empirically useful as SCC achieves SOTA for the DP-means objective (§4.3).

3.3 Hierarchical Clustering Analysis

SCC is reminiscent of hierarchical agglomerative clustering (HAC), which, in each round, merges the two subtrees with minimum distance according to the linkage function. In the following statement, we show that there exists a sequence of thresholds $\tau_0, \ldots, \tau_L$, for which SCC will produce exactly the same tree structure as HAC. To formally make this statement, we need to make the additional assumption that the linkage function is both injective (so as there are no ties in the ordering of mergers) and reducible.

Proposition 2. *Let $d : \mathcal{P}(\boldsymbol{X}) \times \mathcal{P}(\boldsymbol{X}) \to \mathbb{R}^+$ be a linkage function that is symmetric and injective, $C_1, C_2, C_3, C_4 \subset \boldsymbol{X}$, $d(C_1, C_2) = d(C_3, C_4) \iff (C_1 = C_3 \wedge C_2 = C_4) \vee (C_1 = C_4 \wedge C_2 = C_3)$. Let $\mathcal{T}$ be the tree formed by HAC and let d also satisfy reducibility, $\forall C, C', C'' \in \mathcal{T}$, $f(C, C') \leq \min\{d(C, C''), d(C', C'')\} \implies \min\{d(C, C''), d(C', C'')\} \leq d(C \cup C', C'')$ then there exists a sequence of threshold $t_1, \ldots, t_r$ such that the tree formed by SCC, i.e., $\bigcup SCC(\boldsymbol{X}, f, \{\tau_1, \ldots, \tau_r\})$, is the same as $\mathcal{T}$.*

Proof. Each node in the HAC tree, $C \in \mathcal{T}$, has an associated linkage function score, denoted (with abuse of notation) as $d(C)$. We define the threshold-based rounds for SCC such that $t_1, \ldots, t_r$ to be the values, $\{d(C) + \epsilon | C \in \mathcal{T}\}$ sorted in ascending order. Because f is reducible and injective, there is a unique pair of nodes that will be merged in each round. This pair will correspond exactly to the pair that is merged by HAC in the corresponding round. It follows that the resulting tree structures will be identical. □

The connection between SCC and agglomerative clustering is used to prove Proposition 1. We note that SCC produces non-binary tree structures. And so SCC is not meant to directly approximate the binary tree produced by HAC, nor is it not well suited for objectives such as Dasgupta's cost [18].

3.4 Time Analysis

Worst Case. Building the sub-cluster components in each round is achieved by first finding the 1-nearest neighbor of each point that is less than the given round threshold. Then forming weakly connected components in the graph with edges of the aforementioned 1-nearest neighbor relationships. For a given round, let $O(T)$

be the time required to build a 1-nearest neighbor graph over the sub-clusters of a round. Cover Trees [8] allow this to be done in $O(c^{12}N \log N)$ for N elements where c is the expansion constant. Let $O(S)$ be the time required to run connected components. Since we are finding the connected components of a graph with N nodes and at most N edges, a worst case running time of connected components is $O(N)$. We note that each of the one nearest neighbor graph and the connected components algorithm are highly parallelizable in a map-reduce framework. In the worst case, SCC requires $2 * (N - 1)$ rounds (merging one pair of elements per round, multiplicative factor due to determining when to advance idx in Algorithm 1). In general this is a $O(N(T + S)) = O(NT)$ running time. We find that using a fixed number of rounds and simply advancing the threshold in each round experimentally works well.

Sparse Graphs. To speed up computation, we can pre-compute a k nearest-neighbor graph over the dataset. Once such a graph is constructed computing the 1-nearest neighbor of each node can be an $O(k)$ operation for many linkage functions (such as average, single, complete). We can use the sparse k nearest-neighbor graph to compute "top-k" approximation of the linkage functions. For instance, to compute the average linkage between two clusters of points according to the k-nearest-neighbor graph, we can define the distance between any two points that do not share an edge in the sparse k nearest neighbor graph to be a fixed constant (e.g., 4 in the case of ℓ_2^2 or 0 in the case similarities).

4 EXPERIMENTS

We empirically validate the effectiveness of SCC. Paralleling our theoretical contributions, we analyze the method both as hierarchical clustering approach as well as a flat clustering method and DP-Means approximation algorithm. Analysis on a variety of publicly available clustering benchmarks demonstrates that SCC:

- Recovers more accurate hierarchical clustering than state-of-the-art methods (§4.1).
- Produces high quality flat partitions of the data (§4.2).
- Produces lower DP-means cost solutions (§4.3).

Lastly, to demonstrate the scalability of SCC we evaluate on **30 billion** point web-scale dataset (§5).

4.1 Hierarchical Clustering

Datasets: We evaluate the SCC and competing methods on the following publicly available clustering benchmark datasets as in [37] (Table 1): **CovType** - forest cover types; **Speaker** - i-vector speaker recordings, ground truth clusters refer each unique speaker [27]; **ALOI** - 3D rendering of objects, ground truth clusters refer to each object type [24]; **ILSVRC (Sm.) (50K subset)** and **ILSVRC (Lg.) (1.2M Images)** images from the ImageNet ILSVRC 2012 dataset [50] with vector representations of images from the last layer of the Inception neural network.

Methods We analyze the performance of SCC compared to state-of-the-art hierarchical clustering algorithms: **Affinity** [6] - a distributed minimum spanning tree approach based on Borůvka's algorithm [10]; **Perch** [37] - an online hierarchical clustering algorithm that creates trees one point at a time by adding points next to their nearest neighbor and perform local tree re-arrangements in the form of rotations; **Grinch** [45] - similar to Perch this is an online tree building method, this work uses a grafting subroutine in addition to rotations. The graft subroutine allows for more global re-arrangements of the tree structure; **gHHC** [46] - a gradient-based hierarchical clustering method that uses a continuous tree representation in the unit ball; **hierarchical K-Means (HKM)** - the top down divisive algorithm; hierarchical agglomerative clustering (HAC) - the classic bottom up agglomerative approach; **Birch** [59] - a classic top-down hierarchical clustering method; **HDBSCAN** [12] - the hierarchical extension of the classic DBSCAN algorithm. For Perch, Grinch, HKM, Birch, gHHC we report results from previous work [37, 45, 46]. Cosine similarity is meaningful for each of the datasets. For all methods that use a linkage function, we use average linkage which was shown to be effective [45].

	CovType	ILSVRC (Sm.)	ALOI	Spkr.	ILSVRC (Lg.)
$\lvert\mathbb{S}^\star\rvert$	7	1000	1000	4958	1000
$\lvert\boldsymbol{X}\rvert$	500K	50K	108K	36.5K	1.3M
dim.	54	2048	128	6388	2048
BIRCH	0.44	0.26	0.32	0.22	0.11
Perch	0.448	0.531	0.445	0.372	0.207
HAC	-	0.641	0.524	0.518	-
Grinch	0.430	0.557	0.504	0.48	-
HKM	0.44	0.12	0.44	0.12	0.11
gHHC	0.444	0.381	0.462	-	0.367
HDBSCAN	**0.473**	0.414	0.599	0.396	-
Affinity	0.433	0.587	0.478	0.424	0.601
SCC	0.433	**0.644**	**0.619**	**0.528**	**0.614**

Table 1: Dendrogram Purity results on benchmark datasets. gHHC did not produce meaningful results on Speaker and Grinch and HDBSCAN did not scale to ILSVRC (Lg.).

Since the datasets use cosine similarity, for SCC, we use a geometric progression of thresholds for SCC based on the bounds of cosine similarity (switching the sign of threshold condition), which we approximate with minimum (0.001) to maximum (1.0) with 200 thresholds for the algorithm. We use the sparsified nearest neighbor graph approach with k=25 neighbors using we ScaNN [29].

Since each dataset has ground truth flat clusters, we evaluate the quality of hierarchy using *dendrogram purity* as in previous work [31, 37, 45]. Dendrogram purity is the average, over all pairs of points from the same ground truth cluster, of the purity of the least common ancestor of the pair. We hypothesize that SCC's improvement over Affinity clustering comes from its use of round-based thresholds to reduce the overmerging of clusters.

As seen in Table 2, we observe that SCC achieves the highest dendrogram purity on all datasets except CovType. Notably, both SCC and Affinity clustering scale much better to the largest 1.2M image dataset, with both methods having no degradation in performance from the 50K subset to the larger 1.2M point dataset.

We analyze the decision to use geometric progression for thresholds as compared to linear ones in Figure 5. We observe that SCC performs slightly better with geometric sequences compared to linear ones. We also observe that SCC requires only a few more rounds than Affinity to begin to achieve high quality results and that the performance saturates after a few hundred rounds.

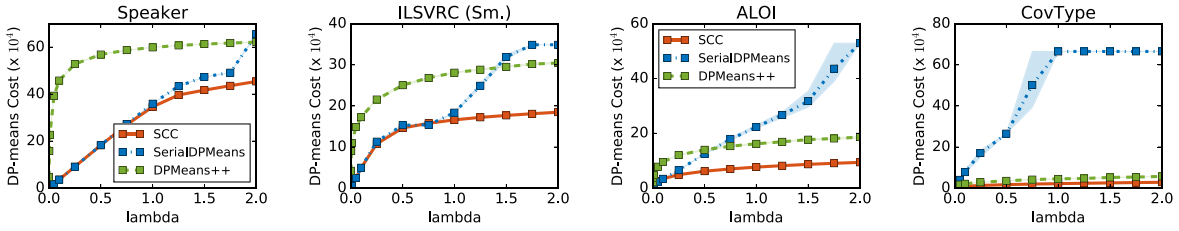

Figure 2: DP-Means Cost for the solutions found with a variety of methods for a range of λ values from close to 0 to 2. SCC produces lower cost solutions for different values of λ as compared to the other methods.

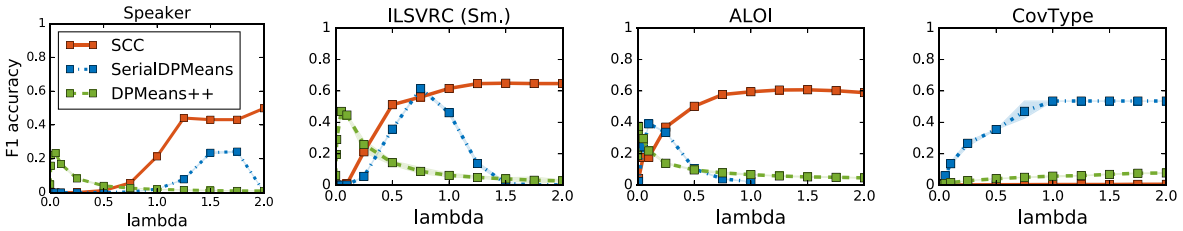

Figure 3: Pairwise F1 Evaluation. As each algorithm depends on λ in a different way, the settings of λ resulting in the best performance might differ between methods. We plot the performance of each method for each value of λ. When considering the best F1 achieved by each method for some value of λ, SCC is the top performing method on 4 of 5 datasets.

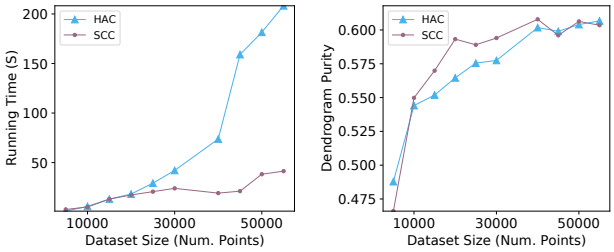

Figure 4: Comparison to HAC: We report running time and Dendrogram Purity of SCC compared to HAC, on a synthetic dataset.

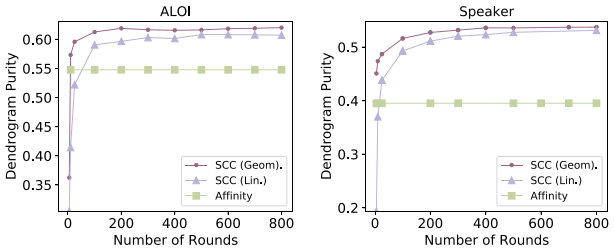

Figure 5: Number of Round & Kinds of Thresholds. We report SCC's performance with both a geometric progression of thresholds and a linear progression of thresholds. We also report Affinity's performance, which converges in about five rounds. Recall that HAC would require number of rounds equal to number of points (much more than 800).

Comparison to HAC We sample datasets of varying sizes from a Dirichlet Process Mixture Model. Figure 4 reports the running time and dendrogram purity as a function of the dataset size. We report SCC results with 200 rounds. We observe that while the time complexity of HAC grows quadratically, SCC remains much more constant. Despite being much more efficient than HAC we observe that SCC produces trees with comparable dendrogram purity.

4.2 Flat Clustering

In this section, we empirically evaluate SCC against state-of-the-art approaches in terms of pairwise F1 of flat clusters discovered. We use the same experimental setting as in previous works [37]. In this experiment, we use the ground truth number of clusters when selecting a flat clustering. We select the round of SCC which produced the closest number of clusters to the ground truth and report the flat clustering performance of that round. We do the same for baseline methods. We evaluate the quality of the flat clusterings using the pairwise F1 metric [37, 44], for which precision is defined as Prec $= \frac{|\mathcal{P}^* \cap \hat{\mathcal{P}}|}{|\hat{\mathcal{P}}|}$, and recall as Rec $= \frac{|\mathcal{P}^* \cap \hat{\mathcal{P}}|}{|\mathcal{P}^*|}$, where $\mathcal{P}^*$ is all pairs of points that are assigned to the same cluster according to the ground truth $\mathbb{S}^*$ and $\hat{\mathcal{P}}$ is similarly defined for the predicted clustering $\hat{\mathbb{S}}$: $\mathcal{P}^* = \{(x_i, x_j) \mid x_i, x_j \in \boldsymbol{X},\ \exists\, C^* \in \mathbb{S}^* \text{ s.t. } \{x_i, x_j\} \subseteq C^*\}$ and $\hat{\mathcal{P}} = \{(x_i, x_j) \mid x_i, x_j \in \boldsymbol{X},\ \exists\, C \in \hat{\mathbb{S}} \text{ s.t. } \{x_i, x_j\} \subseteq C\}$

Table 2 gives the F1 performance for each method on each of the datasets used in the hierarchical clustering experiments. We observe that both SCC and Affinity outperform the previous state-of-the-art results reported by [37]. SCC is the best performing method on Speaker and ALOI and is competitive with Affinity on the remaining datasets. We report the best F1 achieved in *any* round of our algorithm and Affinity, which is the next best performing method. SCC's best F1 is better than Affinity's.

	CovType	ILSVRC (Sm.)	ALOI	Spkr.	ILSVRC (Lg.)
Perch	0.230	0.543	0.442	0.318	0.257
K-Means	0.245	0.605	0.408	0.322	0.562
Affinity	0.536	**0.632**	0.439	0.299	**0.641**
SCC	**0.536**	0.609	**0.567**	**0.493**	0.602
Affinity (best F1)	0.536	0.632	0.465	0.3141	0.641
SCC (best F1)	*0.536*	*0.654*	*0.605*	*0.526*	*0.664*

Table 2: Pairwise F1 results on benchmark datasets.

4.3 Approximation of DP-Means Objective

Our analysis section (§3.2) showed that SCC is a DP-Means approximation algorithm. In this section, we empirically evaluate these claims. We measure the quality of the clustering discovered by our algorithm in terms of the DP-Means objective.

We compare SCC to the following state-of-the-art algorithms for obtaining solutions to the DP-means objective: **SerialDPMeans**, the classic iterative optimization algorithm for DP-Means [11, 32, 40, 49], in which data point is added to a cluster if it is within λ of that cluster center and otherwise starts a new cluster; **DPMeans++** [3] an initialization-only method which performs a K-Means++ [2] style sampling procedure. For each method, we record the assignment of points to clusters given by inference. We use this assignment of points to flat clusters to produce a DP-Means cost. We perform our analysis on the aforementioned clustering benchmarks.

Figure 2 shows the DP-Means objective as a function of the value of the parameter λ. Figure 3 shows the corresponding F1 performance for different values of λ (0.001, 0.005, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0). We report the min/max/average performance over multiple runs of the SerialDPMeans and DPMeans++ algorithms with different random seeds. Here, each method uses normalized ℓ_2^2 distance as the dissimilarity measure. SCC uses thresholds 0.001 to 4 with a geometric progression. We report the min/max/average performance over multiple runs of the SerialDPMeans and DPMeans++ algorithms.

For each value of λ, SCC achieves the lowest DP-Means cost, which we hypothesize is due to SCC discovering optimal clustering independently of λ via multiple alternative partitions in the tree. As each algorithm uses the value of λ differently, the value of λ that results in the best F1 score on the dataset could be quite different for each method and for each dataset. If we consider the best F1 value achieved, SCC is usually one of the best performing methods. On CovType, SerialDPMeans performs best. but F1 does not seem meaningful on CovType as we observed the highest F1 value (0.536) with all points in a single cluster. See Appendix §B for ILSVRC (Lg.) results and comparison to LowrankALBCD [57].

5 APPLICATION: LARGE SCALE CLUSTERING OF WEB QUERIES

We investigate the use of SCC for clustering web queries. We run our proposed clustering approach and Affinity clustering on a dataset

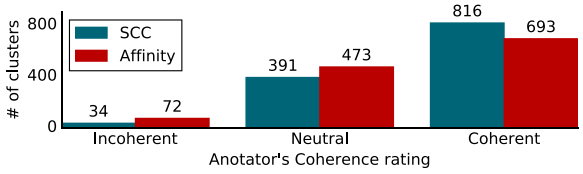

Figure 6: Human evaluation of clusters generated by SCC and Affinity

Tea Recipes	Electric Piano	Tennis Strategy
tea drinks	digital piano price	playing strategies in tennis
tea recipes	electric piano sale	understanding tennis tactics
fancy tea recipes	electric piano small	baseline tactics.
black tea flavors	best digital piano ebay	tennis strategy angles

Table 3: Fine-grained Query Clusters Discovered by SCC

Affinity Clustering	SCC
green velvet live	green velvet mix
green velvet talking	dj green velvet
kestra financial businesswire	tomorrowland green velvet
tomorrow land green velvet	green velvet 2015
dj green hair	green velvet music
rinvelt & david kestra businesswire	green velvet dj set

Table 4: SCC and Affinity Clustering for clusters corresponding to *green velvet* on the 30 billion query dataset.

of a random sample of **30 billion** queries. To the best of our knowledge this is one of the largest evaluations of any clustering algorithm. Due to the massive size of the data, we limit our evaluation to the two most highly performing methods, SCC and Affinity clustering. Distance computation between queries during algorithm execution are sped-up using hashing techniques to avoid the N^2 pairwise dissimilarity bottleneck (for both SCC and Affinity). Queries are represented using a set of proprietary features comprising lexical and behavioral signals among others. We extract manually a fine-grained level of flat clusterings and compared the clustering quality of the flat clusterings discovered by both algorithms.

Human Evaluation: To evaluate the quality of flat clusters discovered by SCC, we conducted an empirical evaluation with human annotators. We asked them to rate ~ 1200 randomly sampled clusters from -1 (incoherent) to +1 (coherent). For example, a annotator might receive a head query of `home improvement` and a tail query of `lowes near me` from the same cluster. The annotator then rates each of these pairs from -1 (incoherent) to +1 (coherent). We report the aggregated results in Figure 6. We found that the annotators labeled 6.0% of Affinity clustering's clusters and only 2.7% of SCC clusters as incoherent. The annotators labeled 55.8% of Affinity's clusters and 65.7% of SCC clusters as coherent. We hope this demonstrates the cogency of the clusters found by SCC.

Qualitative Evaluation: In Table 4, we show the clusters discovered by both SCC and Affinity that contain the query *Green Velvet* (the house/techno music artist). We observe that SCC's cluster is considerably more precise and on topic. Table 3, shows additional

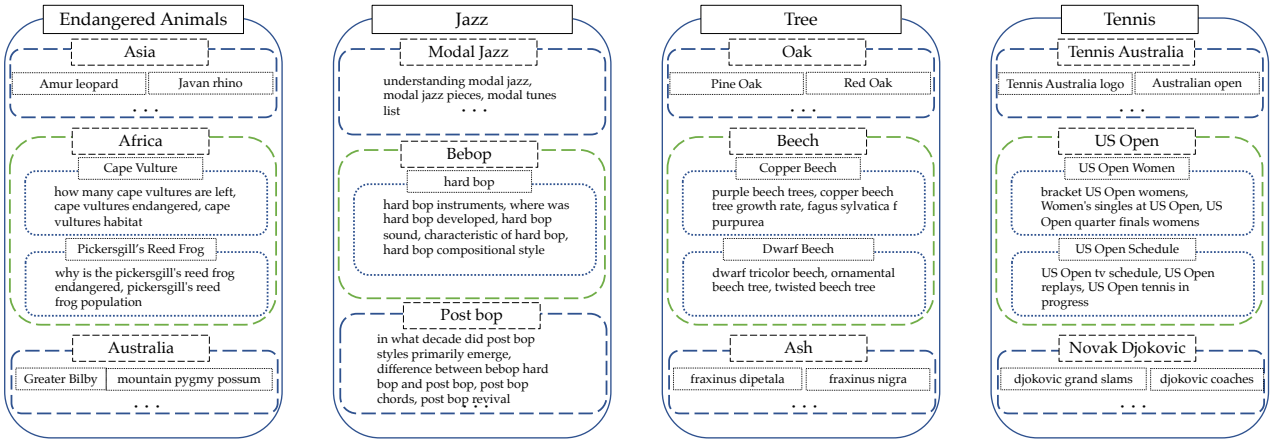

Figure 7: Hierarchy inferred on 30 billion user queries using SCC. We represent the hierarchy using rectangular boxes. The root with header is represented by the outer rectangular box (solid line). The second level of the hierarchy, with header, is shown in dashed (– –) rectangle withing the outer box. Finally the third level from the root is shown as the inner most dotted (...) rectangle. Plain text within each of the dotted rectangle are the top queries that belong to that cluster. For example, Endangered Animals, is the root of the left most hierarchy, Endangered Animals in Africa is a sub-cluster and pickergill's reed frog is the lowest level cluster containing queries such as why is the pickergill's reed frog endangered.

examples. We find the clusters to be precise and coherent. The algorithm discovers a clustering related to tennis strategies, which contains meaningful queries such as *baseline tactics* and *tennis strategy angles*. We investigate the hierarchical structure in Figure 7. We discover meaningful clusters at multiple granularities with the tree structure indicating meaningful relationships between topics. For instance, we discover subgenres of *jazz* such as *bebop* and *modal jazz*. We further discover that *hard bop* is a subgenre of *bebop*.

6 RELATED WORK

Parallel and distributed approaches for hierarchical clustering have been considered by previous work [6, 33, 48, 51, 56]. Yaroslavtsev and Vadapalli [56] use a graph sparsification approach along with parallel minimum spanning tree approach to achieve provably good approximate minimum spanning trees. Other work has proposed to achieve scalability by building hierarchical clusterings in an online, incremental, or streaming manner [37, 38, 45, 59]. BIRCH, one of the most widely used algorithms, builds a tree in a top down fashion splitting nodes under a condition on the mean/variance of the points assigned to a node. PERCH [37] and GRINCH [45] add points next to the nearest neighbor node in the tree structure and perform tree re-arrangements. Kranen et al. [38] build an adaptive index structure for streaming data with allows for recency weighting.

Our work is closely related to the tree-based clustering methods proposed by Balcan et al. [4, 5]. These methods also build a tree structure in a bottom up manner using round-specific thresholds to determine the mergers. These methods in fact recover clusterings under more flexible separation conditions than the one used in this paper. However, the linkage function computation used is much more computationally expensive than the one used in this paper. We show an empirical comparison of this to SCC in Appendix §B .

While this paper has focused on linkage-based hierarchical clustering in general, density-based clustering algorithms like DBSCAN [22] correspond to specific linkage functions. There have been several hierarchical variants of these approaches proposed including, most recently HDBSCAN* [12]. There are other related density and spanning-tree-based approaches [13, 62].

Other work has use techniques to reduce the number of distance computations required to perform hierarchical clustering [21, 39]. Krishnamurthy et al. [39] uses only $O(n \log^2(n))$ distance computations by repeatedly running a flat clustering algorithm on small subsets of the data to discover the children of each node in a top-down way. Other work uses nearest neighbor graph sparsification to reduce the complexity of algorithms [58].

A variety of objective functions have been proposed for hierarchical clustering. Some work has used integer linear programming to perform hierarchical agglomerative clustering [25]. Notably, Dasgupta's cost function [18] has been widely studied, including its connections to agglomerative clustering [14, 47].

7 CONCLUSION

We introduce the Sub-Cluster Component algorithm (SCC) for scalable hierarchical as well as flat clustering. SCC uses an agglomerative, round-based, approach in which a series of increasing distance thresholds are used to determine which sub-clusters can be merged in a given round. We provide a theoretical analysis of SCC under well studied separability assumptions and relate it to the non-parametric DP-means objective. We perform a comprehensive empirical analysis of SCC demonstrating its proficiency over state-of-the-art clustering algorithms on benchmark clustering datasets. We further provide an analysis of the method with respect to the DP-means objective. Finally, we demonstrate the scalability of SCC by running on an industrial web-scale dataset of 30B user queries. We evaluate clustering quality on this web-scale dataset with human annotations that indicate that the clusters produced by SCC are more coherent than state-of-the-art methods.

ACKNOWLEDGEMENTS

We thank Avrim Blum and Nina Balcan for their insightful comments and suggestions. Andrew McCallum and Nicholas Monath are supported in part by the Center for Data Science and the Center for Intelligent Information Retrieval and in part by the National Science Foundation under Grants No. 1763618. Some of the work reported here was performed using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

REFERENCES

[1] N. Andrews, J. Eisner, and M. Dredze. 2014. Robust entity clustering via phylogenetic inference. *ACL* (2014).
[2] D. Arthur and S. Vassilvitskii. 2007. k-means++: The advantages of careful seeding. *SODA* (2007).
[3] O. Bachem, M. Lucic, and A. Krause. 2015. Coresets for Nonparametric Estimation-the Case of DP-Means. *ICML* (2015).
[4] M.F. Balcan, A. Blum, and S. Vempala. 2008. A discriminative framework for clustering via similarity functions. *STOC* (2008).
[5] M.F. Balcan, Y. Liang, and P. Gupta. 2014. Robust hierarchical clustering. *JMLR* (2014).
[6] M. Bateni, S. Behnezhad, M. Derakhshan, M. Hajiaghayi, R. Kiveris, S. Lattanzi, and V. Mirrokni. 2017. Affinity Clustering: Hierarchical Clustering at Scale. *NeurIPS* (2017).
[7] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida. 2012. Characterizing user navigation and interactions in online social networks. *Inf. Sci.* (2012).
[8] A. Beygelzimer, S. Kakade, and J. Langford. 2006. Cover trees for nearest neighbor. *ICML* (2006).
[9] C. Blundell, Y. W. Teh, and K. A. Heller. 2010. Bayesian rose trees. *UAI* (2010).
[10] O. Borůvka. 1926. O jistém problému minimálním. (1926).
[11] T. Broderick, B. Kulis, and M. Jordan. 2013. MAD-Bayes: MAP-based asymptotic derivations from Bayes. *ICML* (2013).
[12] R. J. G. B. Campello, D. Moulavi, and J. Sander. 2013. Density-Based Clustering Based on Hierarchical Density Estimates. *Advances in Knowledge Discovery and Data Mining* (2013).
[13] F. Cao, M. Ester, W. Qian, and A. Zhou. 2006. Density-based clustering over an evolving data stream with noise. *ICDM* (2006).
[14] M. Charikar, V. Chatziafratis, and R. Niazadeh. 2019. Hierarchical Clustering better than Average-Linkage. *SODA* (2019).
[15] B. Chen, A. Shrivastava, R. C Steorts, et al. 2018. Unique entity estimation with application to the Syrian conflict. *The Annals of Applied Statistics* (2018).
[16] K. Cranmer, S. Macaluso, and D. Pappadopulo. 2019. Toy Generative Model for Jets. (2019).
[17] R. Das, A. Godbole, N. Monath, M. Zaheer, and A. McCallum. 2020. Probabilistic Case-based Reasoning for Open-World Knowledge Graph Completion. *Findings of EMNLP* (2020).
[18] S. Dasgupta. 2016. A cost function for similarity-based hierarchical clustering. *Symposium on Theory of Computing (STOC)* (2016).
[19] L. Ein Dor, Y. Mass, A. Halfon, E. Venezian, I. Shnayderman, R. Aharonov, and N. Slonim. 2018. Learning Thematic Similarity Metric from Article Sections Using Triplet Networks. *ACL* (2018).
[20] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. 1998. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences* (1998).
[21] B. Eriksson, G. Dasarathy, A. Singh, and R. Nowak. 2011. Active clustering: Robust and efficient hierarchical clustering using adaptively selected similarities. *AISTATS* (2011).
[22] M. Ester, H-P Kriegel, J Sander, X Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD* (1996).
[23] A. Gavryushkin and A. J Drummond. 2016. The space of ultrametric phylogenetic trees. *Journal of theoretical biology* (2016).
[24] J.M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. 2005. The Amsterdam Library of Object Images. *IJCV* (2005).
[25] S. Gilpin, S. Nijssen, and I. Davidson. 2013. Formalizing hierarchical clustering as integer linear programming. *AAAI* (2013).
[26] S. Green, N. Andrews, M. R. Gormley, M. Dredze, and C. D. Manning. 2012. Entity Clustering Across Languages. *NAACL-HLT* (2012).
[27] C. S et al Greenberg. 2014. The NIST 2014 speaker recognition i-vector machine learning challenge. *Odyssey: The Speaker and Language Recognition Workshop* (2014).
[28] A. Großwendt, H. Röglin, and M. Schmidt. 2019. Analysis of ward's method. *SODA* (2019).
[29] R. et al Guo. 2020. Accelerating Large-Scale Inference with Anisotropic Vector Quantization. In *ICML*.
[30] K Heller and Zoubin Ghahramani. 2005. Randomized algorithms for fast Bayesian hierarchical clustering. *EU-PASCAL Statistics and Optimization of Clustering Workshop* (2005).
[31] K. A. Heller and Z. Ghahramani. 2005. Bayesian hierarchical clustering. *ICML* (2005).
[32] K. Jiang, B. Kulis, and M. I Jordan. 2012. Small-variance asymptotics for exponential family Dirichlet process mixture models. *NeurIPS* (2012).
[33] C. et al Jin. 2015. A scalable hierarchical clustering algorithm using spark. *BigDataService* (2015).
[34] K. Kenyon-Dean, J. C. K. Cheung, and D. Precup. 2018. Resolving Event Coreference with Supervised Representation Learning and Clustering-Oriented Regularization. **SEM* (2018).
[35] J. Kleinberg. 2002. An impossibility theorem for clustering. *NeurIPS* (2002).
[36] A. Kobren, N. Monath, and A. McCallum. 2019. Integrating User Feedback under Identity Uncertainty in Knowledge Base Construction. *AKBC* (2019).
[37] N. Kobren, A.and Monath, A. Krishnamurthy, and A. McCallum. 2017. A hierarchical algorithm for extreme clustering. *KDD* (2017).
[38] P. Kranen, I. Assent, C. Baldauf, and T. Seidl. 2011. The ClusTree: indexing micro-clusters for anytime stream mining. *KIS* (2011).
[39] A. Krishnamurthy, S. Balakrishnan, M. Xu, and A. Singh. 2012. Efficient active algorithms for hierarchical clustering. *ICML* (2012).
[40] B. Kulis and M. I. Jordan. 2012. Revisiting K-Means: New Algorithms via Bayesian Nonparametrics. *ICML* (2012).
[41] S. Kushagra, S. Samadi, and S. Ben-David. 2016. Finding meaningful cluster structure amidst background noise. *ALT* (2016).
[42] H. Lee, M. Recasens, A. Chang, M. Surdeanu, and D. Jurafsky. 2012. Joint entity and event coreference resolution across documents. *EMNLP-CoNLL* (2012).
[43] M. Mahajan, P. Nimbhorkar, and K. Varadarajan. 2009. The planar k-means problem is NP-hard. *WALCOM* (2009).
[44] C. D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval.*
[45] N. Monath, Akshay Kobren, A.and Krishnamurthy, Michael R Glass, and A. McCallum. 2019. Scalable Hierarchical Clustering with Tree Grafting. *KDD* (2019).
[46] N. Monath, M. Zaheer, D. Silva, A. McCallum, and A. Ahmed. 2019. Gradient-Based Hierarchical Clustering Using Continuous Representations of Trees in Hyperbolic Space. *KDD* (2019).
[47] B. Moseley and J. Wang. 2017. Approximation Bounds for Hierarchical Clustering: Average Linkage, Bisecting K-means, and Local Search. *NeurIPS* (2017).
[48] C. F Olson. 1995. Parallel algorithms for hierarchical clustering. *Parallel computing* (1995).
[49] X. Pan, J. E Gonzalez, S. Jegelka, T. Broderick, and M. I Jordan. 2013. Optimistic concurrency control for distributed unsupervised learning. *NeurIPS* (2013).
[50] O. et al Russakovsky. 2015. Imagenet large scale visual recognition challenge. *IJCV* (2015).
[51] J. Santos, T. Syed, M. C. Naldi, R. JGB Campello, and J. Sander. 2019. Hierarchical Density-Based Clustering using MapReduce. *TBD* (2019).
[52] G. W et al Schwartz. 2020. TooManyCells identifies and visualizes relationships of single-cell clades. *Nat. Methods* (2020).
[53] V. V Vazirani. 2013. *Approximation algorithms.*
[54] F. Vitale, A. Rajagopalan, and C. Gentile. 2019. Flattening a Hierarchical Clustering through Active Learning. *NeurIPS* (2019).
[55] N. Yadav, A. Kobren, N. Monath, and A. Mccallum. 2019. Supervised Hierarchical Clustering with Exponential Linkage. *ICML* (2019).
[56] G. Yaroslavtsev and A. Vadapalli. 2018. Massively Parallel Algorithms and Hardness for Single-Linkage Clustering under ℓ_p Distances. *ICML* (2018).
[57] I. E.H. Yen, D. Malioutov, and A. Kumar. 2016. Scalable exemplar clustering and facility location via augmented block coordinate descent with column generation. *AISTATS* (2016).
[58] M. Zaheer, S. Kottur, A. Ahmed, J. Moura, and Alex Smola. 2017. Canopy—fast sampling with cover trees. *ICML* (2017).
[59] T. Zhang, R. Ramakrishnan, and M. Livny. 1996. BIRCH: an efficient data clustering method for very large databases. *SIGMOD* (1996).
[60] Y. Zhang, A. Ahmed, V. Josifovski, and A. Smola. 2014. Taxonomy discovery for personalized recommendation. *WSDM* (2014).
[61] Y. Zhang, F. Zhang, P. Yao, and J. Tang. 2018. Name Disambiguation in AMiner: Clustering, Maintenance, and Human in the Loop. *KDD* (2018).
[62] H. Zhu and W. Stuetzle. 2019. A Simple and Efficient Method to Compute a Single Linkage Dendrogram. *arXiv* (2019).

SCALABLE HIERARCHICAL AGGLOMERATIVE CLUSTERING – APPENDIX

A PROOFS

A.1 Proof of Theorem 1

Here we show a proof for the ℓ_2^2 case, the ℓ_2 case follows the same proof but uses the triangle inequality instead of the relaxed triangle inequality. Recall the assumption of δ-separability (Assumption 1) in which the maximum distance from any point to its true center is defined as $R := \max_{i\in[k]} \max_{x\in C_i^*} \|x - C_i^*\|_2^2$. We make the additional assumption that the threshold of the first round, τ_0, is less than R, i.e., $\tau_0 < R$.

The algorithm begins with $\mathbb{S}^{(0)}$ set to be the shattered partition, with each data point in its own cluster, $\mathbb{S}^{(0)} = \{\{x\}|x \in \boldsymbol{X}\}$.

We want to show that some round, $r^\star$ with threshold τ_r produces the ground truth partition, $\mathbb{S}^{(r^\star)} = \mathbb{S}^* = \{C_1^*, \ldots, C_k^*\}$. We will show by induction that for each round prior $r' \le r^\star$ that the clustering $\mathbb{S}^{(r')}$ is *pure*, i.e. $\forall C \in \mathbb{S}^{(r')}$, $\exists C^* \in \mathbb{S}^*$ $C \subseteq C^*$ (equality will be for round $r^\star$). We will show the round $r^\star$ with $\mathbb{S}^{(r^\star)} = \mathbb{S}^*$ must exist.

In our inductive hypothesis, we assume that for rounds before and including τ_r, we have *pure* sub-clusters such that $X, X' \subseteq C_i^*$, which are disjoint, $X \cap X' = \emptyset$, and $Y \subseteq C_j^*$ for $i \neq j$ ($\mathbb{S}^{(0)}$ by definition has pure sub-clusters). We want to show: every such X and X' must form a sub-cluster component without any such Y. In this way, we ensure that C_i^* exists as a pure cluster in some round. Using the relaxed triangle inequality [28] for ℓ_2^2, we have:

$$\|c_i^* - c_j^*\|_2^2 \le 3\left(\frac{1}{|X||Y|}\sum_{x\in X}\sum_{y\in Y}\|c_i^* - x\|_2^2 + \|x - y\|_2^2 + \|y - c_j^*\|_2^2\right)$$

$$\|c_i^* - c_j^*\|_2^2 \le 3\left(\frac{1}{|X|}\sum_{x\in X}\|c_i^* - x\|_2^2 + \frac{1}{|X||Y|}\sum_{x\in X}\sum_{y\in Y}\|x - y\|_2^2 + \frac{1}{|Y|}\sum_{y\in Y}\|y - c_j^*\|_2^2\right)$$

$$\frac{1}{3}\|c_i^* - c_j^*\|_2^2 - \frac{1}{|X|}\sum_{x\in X}\|c_i^* - x\|_2^2 - \frac{1}{|Y|}\sum_{y\in Y}\|y - c_j^*\|_2^2 \le \frac{1}{|X||Y|}\sum_{x\in X}\sum_{y\in Y}\|x - y\|_2^2$$

Since $\|c_i^* - c_j^*\|_2^2 \ge \delta \cdot R$ and by the definition of R,

$$(\tfrac{1}{3}\delta - 2)\cdot R \le \frac{1}{3}\|c_i^* - c_j^*\|_2^2 - \frac{1}{|X|}\sum_{x\in X}\|c_i^* - x\|_2^2 - \frac{1}{|Y|}\sum_{y\in Y}\|y - c_j^*\|_2^2 \le \frac{1}{|X||Y|}\sum_{x\in X}\sum_{y\in Y}\|x - y\|_2^2$$

However, for any two subclusters $X, X' \subset C_i^*$ we know that:

$$\frac{1}{|X||X'|}\sum_{x\in X}\sum_{x\in X'}\|x - x'\|_2^2 \le 2\left(\frac{1}{|X|}\sum_{x\in X}\|c_i^* - x\|_2^2 + \le \frac{1}{|X'|}\sum_{x\in X'}\|c_i^* - x'\|_2^2\right) \le 4R.$$

For $\delta \ge 30$, we know that there exists a $4R \le \tau_r \le 8R$ for which X and X' would form a sub-cluster component without Y. Since we use a geometric sequence of $\tau_1, \tau_2, \ldots$, we know that τ_r will exist since it is between $4R$ and the doubling of $4R$. X and X' are any sub-clusters of any ground truth cluster C_i^*. The result above indicates that at any round before the one using τ_r that takes as input pure sub-clusters will produce pure sub-clusters as no sub-clusters with points belonging to different ground truth clusters will be merged. Moreover, the existence of τ_r indicates that the partition given by sub-cluster component from a round using τ_r , will contain every ground truth cluster in $\mathbb{S}^*$. In particular, the last round that uses τ_r will be the $r^\star$ to do this, i.e., $\mathbb{S}^{(r^\star)} = \mathbb{S}^\star$. Observe that the separation condition requires that within cluster distances for any two subsets will be less than τ_r and so sub-clusters will continue to be merged together until each ground truth cluster is formed by the last round using τ_r.

A.2 Proof of Proposition 1

Previous work shows that hierarchical agglomerative clustering (HAC) can recover model-based separated data [45]. Proposition 2 shows that there exists a sequence of thresholds for which SCC produces the same tree structure as HAC and thus this same sequence of thresholds could be used to recover model-based separate data. It would also be possible to compress this sequence of thresholds to contain only those thresholds necessary to prevent the overmerging of clusters with points from different ground truth classes (which must exist by the previous result).

A.3 Proof of Theorem 2

We showed that under δ-separation SCC will recover the target partition (Theorem 1). We first relate this target partition to the facility location using the DP-Facility Problem and then use the relationship between Facility Location and DP-Means [49]. Facility Location problem is defined as:

Definition 5. **[Facility Location]** *Given a set of clients H as well as facilities F, and a set $\mathbf{f} = \{f_1, \ldots, f_K\}$ of facility opening costs, that is let f_j be the cost of opening facility j and let $e(i, j)$ be the cost of connecting client i to the open facility j. Let $I \subseteq F$ be the set of opened facilities and let $\phi : H \to I$ be the mapping from clients to facilities. The total cost of opening a set of facilities is:*

$$cost(H, F, I, \phi, \mathbf{f}) = \sum_{i\in H} e(i, \phi(i)) + \sum_{i\in I} f_i \tag{4}$$

The facility location problem is to solve: $\text{argmin}_{I,\phi}\, cost(H, F, I, \phi, \mathbf{f})$ *given e, F and $\mathbf{f}$.*

As shown by Pan et al. [49], Facility Location is closely related to DP-Means. In particular, the solution of Facility Location gives a solution to DP-Means:

Definition 6. **[DP-Facility [49]]** *We define the DP-Facility problem to be the facility location problem where $f_j = \lambda$ for all facilities $j \in F$, $H = F = \boldsymbol{X}$ and e be* squared euclidean distance. *Given a solution $I, \phi = \text{argmin}_{I,\phi}\, cost(\boldsymbol{X}, \boldsymbol{X}, I, \phi, \lambda)$, we define that $c_\cdot := I$ and $C_k = \{i|\phi(i) = x_k\}$, $K = |I|$ and say that $\mathbb{S} = (C_1, \ldots, C_K)$ is a solution to DP-Means given by the solution of DP-Facility.*

First, we consider δ-separated data in DP-Facility problem and show that the target separated partition gives an optimal solution to DP-Facility:

Proposition 3. *Suppose the dataset $\boldsymbol{X}$ satisfies the δ-separability assumption with respect to clustering $C_1^\star, \ldots, C_k^\star$, then this clustering is an optimal solution to the DP-Facility problem with $\lambda = (\delta - 2)\cdot R$. where $R := \max_{l\in[k]} \max_{x\in C_l^\star} ||x - c_l^*||$.*

Proof. To show that this clustering is an optimal solution to the DP-Facility problem, we will use linear programming duality. In particular, we will exhibit a feasible dual, α, to the linear programming relaxation of the DP-Facility Problem, whose cost is the same as the clustering $\{C_1^*, \ldots, C_k^*\}$. From linear programming duality, we know that the following set of relations are true : $\text{cost}(\alpha) \le \text{Opt(Dual)} = \text{OPT(Primal)} \le \text{cost}(C^*)$. Combined with the fact that $\text{cost}(\alpha) = \text{cost}(C^*)$, this will show that clustering is an optimal solution to the DP-Facility problem.

Consider the linear programming relaxation to DP-Facility problem. This LP is an adaption of the classical LP used for the facility location problem considered in [53][Ch. 17].

$$\min \sum_{i \in F} \sum_{j \in C} e(i,j) \cdot z_{i,j} + \lambda \sum_{i \in F} y_i$$
$$\sum_{i \in F} z_{ij} \geq 1 \qquad \text{for all } j \in C$$
$$y_i - z_{ij} \geq 0 \qquad \text{for all } j \in C$$
$$z, y \geq 0$$

The above program contains two variables z_{ij} indicating if client j is connected to facility i and variables y_i indicates if facility i is open. In particular, every feasible solution to the DP-Facility problem is a candidate solution to the above LP. The dual to the above program is given below:

$$\max \sum_{j \in C} \alpha_j$$
$$\sum_{j} \beta_{ij} \leq \lambda \qquad \text{for all } i \in F$$
$$(\alpha_j - e(i,j)) \leq \beta_{ij} \qquad \text{for all } i \in F, j \in C$$
$$\alpha, \beta \geq 0$$

For each cluster C_i, and each point in the cluster $x \in C_i$, $\alpha_x = ((\delta - 2)R + e(c_i^*, x)/r$ where r is the size of cluster $r := |C_j|$. By δ-separability assumption, we can deduce that $\beta_{\tilde{i}x} = 0$ for all other clusters $C_{\tilde{i}}^* \neq C_i^*$. However, for all $x \in C_i$, we will have $\sum_{x \in C_i} \beta_{ix} = r \cdot \frac{\lambda}{r} = \lambda$. This shows that α is a valid dual to the LP. □

The next proposition formally relates the DP-Facility problem to an approximate solution to the DP-Means objective.

Proposition 4. *[49] Let $\boldsymbol{\mu}^\star, Z^\star, K^\star$ be an optimal solution to the DP-Means problem and let $\boldsymbol{\mu}^\dagger, Z^\dagger, K^\dagger$ be the DP-Means solution given by an optimal solution, $I^\dagger, \phi^\dagger$ to the DP-facility location problem. Then, $DP(X, \lambda, Z^\dagger, \boldsymbol{\mu}^\dagger, K^\dagger) \leq 2 \cdot DP(X, \lambda, Z^*, \boldsymbol{\mu}^*, K^*)$*

Finally, we can analyze the quality of the solutions found by SCC on δ-separated data, showing that it is a constant factor approximation. Using theorem 1 and Proposition 3, when the data satisfies the δ−separation assumption, then SCC contains the optimal solution to DP-facility problem. Finally using Proposition 4, we see that this solution is within 2 factor of the DP-means solution.

B EXPERIMENTS

ILSVRC (Lg.) DP-Means. We replace the SerialDPMeans baseline with its parallel/distributed variant **OCC** [49]. OCC can only be run with $\lambda < 4.0$ (max normalized ℓ_2^2 distance). We find that it takes longer than 10 hours to run more than 2 iterations of OCC, for lambda values less than 0.75. After these iterations, we observe that the lambda value of 0.75 gives a reasonable F1 score for OCC. We observe that SCC achieves higher F1 scores when the value of λ is larger, as having too small a λ value creates too many clusters. We observe that SCC produces lower costs than DPMeans++ and achieves a higher F1 value for particular values of λ.

LowrankALBCD [57] uses values of lambda at the scale of $\lambda = 0.01 \cdot N$ and normalized ℓ_2^2. Any value of λ greater than the maximum pairwise distance (4) will result in SerialDPMeans and OCC necessarily giving solutions of every data point in the same cluster. We evaluated LowrankALBCD on CovType, ALOI, Speaker, and ILSVRC (Small). We use the code provided by the authors of LowrankALBCD to solutions with small values of λ in the range 0 to 4, for large numbers of iterations we found the code either required more than 100GB of RAM or took longer than 10 hours. Instead, we compare our method and LowrankALBCD for larger values of lambda, specifically the authors suggestion of $0.01N$. We observe that on SCC and LowrankALBCD perform similarly on several of the datasets and LowrankALBCD performs better on CovType. However, we notice that for all datasets except CovType, these values of λ produce unreasonably few clusters.

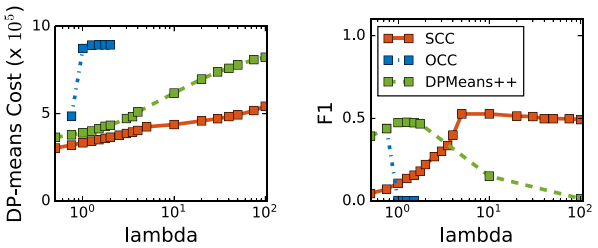

Figure 8: DP-Means and F1 accuracy for ILSVRC (Lg.) dataset

	ALOI	ILSVRC (Sm.)	ILSVRC (Lg.)
HDBSCAN	1635	7902.90	DNF
Grinch	385.113	836.748	DNF
Affinity	29.01 + 0.834	261.831 + 0.298	849.846 + 9.886
SCC	29.01 + 2.79	261.831 + 4.29	849.846 + 65.621

Table 5: Running Time (seconds) of Top Performing Methods. We show the running times in seconds. Each run on machine using 24 2.40GHz CPUs. Grinch and HDBSCAN did not finish on largest dataset in 10 hours. Affinity and SCC report Sparse Graph Construction Time + Algorithm Execution Time for given graph. DNF = Did not finish in 10hours.

Running Times. We report running times in Table 5. The time required by SCC and Affinity is dominated by the construction of the sparse nearest neighbor graph. We use the publicly available implementations of HDBSCAN and Grinch.

Robust Hierarchical Clustering (RHC) [5] RHC has stronger theoretical guarantees than SCC. In Table 6, we compare the best dendrogram purity achieved by SCC and RHC on the Iris and Wine datasets using a grid search over each method's hyperparameters ($\alpha + \nu$ for RHC, number of nearest neighbors and rounds for SCC). We use the publicly available MATLAB implementation of RHC.

	RHC	SCC
Iris	**0.955**	0.926
Wine	0.944	**0.975**

Table 6: Comparison to RHC We report the best dendrogram purity achieved across various hyperparameter settings of each method.