Scalable Bottom-Up Hierarchical Clustering

Nicholas Monath[2], **Avinava Dubey**[1], **Guru Guruganesh**[1], **Manzil Zaheer**[1], **Amr Ahmed**[1], **Andrew McCallum**[2], **Gokhan Mergen**[1], **Marc Najork**[1], **Mert Terzihan**[1], **Bryon Tjanaka**[3], **Yuan Wang**[1], **Yuchen Wu**[1]

1. Google LLC
2. University of Massachusetts Amherst
3. University of Southern California

Abstract

Bottom-up algorithms such as the classic hierarchical agglomerative clustering, are highly effective for hierarchical as well as flat clustering. However, the large number of rounds and their sequential nature limit the scalability of agglomerative clustering. In this paper, we present an alternative round-based bottom-up hierarchical clustering, the Sub-Cluster Component Algorithm (SCC), that scales gracefully to massive datasets. Our method builds many sub-clusters in parallel in a given round and requires many fewer rounds —usually an order of magnitude smaller than classic agglomerative clustering. Our theoretical analysis shows that, under a modest separability assumption, SCC will contain the optimal flat clustering. SCC also provides a 2-approx solution to the DP-means objective, thereby introducing a novel application of hierarchical clustering methods. Empirically, SCC finds better hierarchies and flat clusterings even when the data does not satisfy the separability assumption. We demonstrate the scalability of our method by applying it to a dataset of 30 billion points and showing that SCC produces higher quality clusterings than the state-of-the-art.

1 Introduction

Hierarchical clustering is widely used for tasks such as data analysis and visualization, (Schwartz et al., 2020, inter alia) and entity resolution (Levin et al., 2012, inter alia). Hierarchical agglomerative clustering (HAC), the classic round-based bottom-up greedy algorithm is one of the most widely used and consistently top performing hierarchical clustering algorithm (Green et al., 2012; Kobren et al., 2017, inter alia). It is used as the basis for inference in many statistical models (Heller and Ghahramani, 2005a, inter-alia), as an approximation algorithm for hierarchical clustering (Dasgupta, 2016; Moseley and Wang, 2017, inter-alia) as well as for supervised clustering (Kenyon-Dean et al., 2018; Yadav et al., 2019). Interestingly, the hierarchical clustering algorithm has also been shown to be effective for flat clustering both theoretically in terms of K-means costs (Großwendt et al., 2019) as well as empirically (Green et al., 2012; Kobren et al., 2017).

The key limitation of hierarchical clustering is its 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 (Monath et al., 2019a), or by using randomized algorithms (Heller and Ghahramani, 2005a), or using parallel/distributed methods (Bateni et al., 2017). Scalability is often achieved at the cost of accuracy. Affinity clustering (Bateni et al., 2017), overcomes the main computational expense in HAC algorithm by leveraging distributed connected component algorithms. While efficient, 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)*. Our method uses a sequence of round-based thresholds to determine in which round certain mergers of sub-clusters should take place, thereby preventing over-merging and improving the accuracy of the predicted clustering.

Theoretical Contributions (§3): We analyze the algorithm theoretically showing that SCC:

- Produces hierarchies containing the optimal flat partition for data that satisfies δ-separability

Corresponding authors: NM (nmonath@cs.umass.edu), AD (avinavadubey@google.com). Work done while NM and BT were interns at Google. Preprint.

(Kushagra et al., 2016) and achieves a constant factor approximation of *DP-means objective* (Broderick et al., 2013), a flexible objective for flat clustering which adapts to different numbers of clusters.
- Recovers trees with perfect dendrogram purity on such data and generalizes HAC (HAC is a special case for any dataset when a reducible linkage function is used).

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 than competing state-of-the-art methods (Pan et al., 2013).
- 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 scalabiltiy 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 (Krishnamurthy et al., 2012)**]. *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* (Heller and Ghahramani, 2005b). A tree consistent partition is a set of internal nodes $\{C_1, \ldots C_K\} \subset \mathcal{T}$ that form a flat clustering.

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 clusters from the previous round that satisfy a certain "condition" determined by the threshold. 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 subset defined in the partition i.e. $C \in \mathbb{S}^{(i)}$.

The algorithm uses a linkage function: $d : 2^{\boldsymbol{X}} \times 2^{\boldsymbol{X}} \to \mathbb{R}^+$. In practice, any linkage and distance / similarity measure may be used. For consistency with theoretical results, we use the average pairwise dissimilarity between the points in the sub-clusters for our presentation:

$$d(C_j, C_k) = \frac{1}{|C_j||C_k|} \sum_{x_j \in C_j} \sum_{x_k \in C_k} ||x_j - x_k||^2 \qquad (1)$$

Let $\tau_1, \ldots \tau_L$ be a series of L predefined increasing thresholds. 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 dissimilarity $d : 2^{\boldsymbol{X}} \times 2^{\boldsymbol{X}} \to \mathbb{R}^+$, denoted $\text{CH}_d(C_j, C_k, \tau, \mathbb{S}) = 1$, 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* $0 \leq r \leq R$,
2. $C_{s_{r-1}} = \text{argmin}_{C \in \mathbb{S}} d(C_{s_r}, C)$ *and/or* $C_{s_r} = \text{argmin}_{C \in \mathbb{S}} d(C_{s_{r-1}}, C)$.

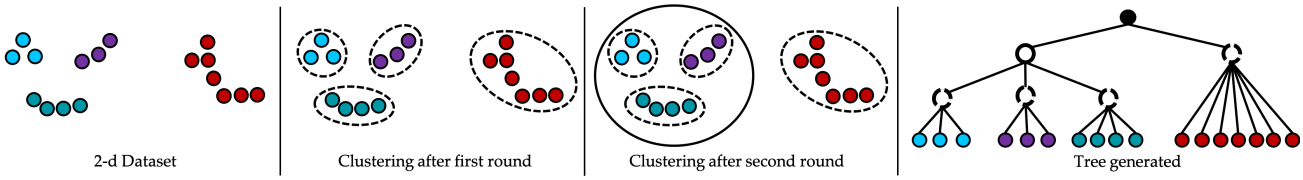

Figure 1: From left to right: toy 2-dimensional dataset, in dotted lines clusters obtained after running first round of SCC, clusters obtained using second round of SCC, and the final tree obtained from SCC.

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

$$\text{SC}_d(C_j, \mathbb{S}, \tau) := \bigcup_{\substack{C \in \mathbb{S}, \\ \text{CH}_d(C_j, C, \tau, \mathbb{S})=1}} C \qquad (2)$$

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:

$$\mathbb{S}^{(i)} := \{\text{SC}_d(C, \mathbb{S}^{(i-1)}, \tau^i) \,|\, C \in \mathbb{S}^{(i-1)}\} \qquad (3)$$

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. $\mathbb{S}^{(i-1)} = \mathbb{S}^{(i)}$. Sub-cluster component in a particular round can be found efficiently using a connected components algorithm (Borůvka, 1926; Kruskal, 1956) as mentioned above. 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 shows the flat clustering and hierarchical clusters formed by running Alg. 1 on toy two dimensional dataset.

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
$\mathbb{S}^{(0)} \leftarrow \{\{x\} \mid x \in \boldsymbol{X}\}$
$\mathsf{idx} \leftarrow 1$, $i \leftarrow 1$
while $\mathsf{idx} < L$ **do**
    Set $\text{SC}_d(C_i, \mathbb{S}^{(i-1)}, \tau^{(i)})$, $\forall C_i \in \mathbb{S}^{(i-1)}$, (Eq. 2)
    Set $\mathbb{S}^{(i)}$ (Eq. 3)
    $\mathsf{idx} \leftarrow \mathsf{idx} + \mathbb{I}[\mathbb{S}^{(i)} = \mathbb{S}^{(i-1)}]$
    $i \leftarrow i + \mathbb{I}[\mathbb{S}^{(i)} \neq \mathbb{S}^{(i-1)}]$
    $\tau^{(i)} \leftarrow \tau_{\mathsf{idx}}$
return $(\mathbb{S}^{(0)}, \ldots, \mathbb{S}^{(i-1)})$

3 Analysis

In this section, 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 relate our method to agglomerative clustering and show that in the limit of number of rounds our method will produce the same tree structure as agglomerative clustering.

3.1 Separation Assumptions

Separability assumptions in clustering provide a mechanism to understand whether or not an algorithm effectively and efficiently recovers cluster structure under "reasonable" conditions. Following much previous work (Jaiswal and Garg, 2012; Kushagra et al., 2016; Bachem et al., 2016; Kobren et al., 2017), our analysis of SCC will consider the case where data satisfies δ-Separability (Kushagra et al., 2016):

Assumption 1. (δ-Separability (Kushagra et al., 2016)) *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 $R := \max_{l \in [k]} \max_{x \in C_l} ||x - c_l^\star||$.*

3.2 Recovering the Target Clustering

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 the average distance*

between points 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 Section A.1. Intuitively, we prove that, for the aforementioned bounds on within/across cluster distances, a geometric series will necessarily 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 predicted flat clustering, $\mathbb{S}^{(i)}$, equal to the target clustering $\mathbb{S}^\star$, $\mathbb{S}^{(i)} = \mathbb{S}^\star$.

3.3 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 (Andrews et al., 2014; Steorts et al., 2017). DP-means (Jiang et al., 2012) 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 (Jiang et al., 2012)]** *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:*

$$DP(\boldsymbol{X}, \lambda, \mathbb{S}) = \sum_{C_l \in \mathbb{S}} \sum_{x \in C_l} ||x - c_l||^2 + \lambda |\mathbb{S}|. \quad (4)$$

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

We show that our proposed algorithm yields a constant factor approximation to DP-Means solution under the data separation Assumption 1. We do so by first showing that SCC contains constant factor approximation solution to Facility Location and then use the relationship between Facility Location and DP-Means (Pan et al., 2013). 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:*

$$cost(H, F, I, \phi, \mathbf{f}) = \sum_{i \in H} e(i, \phi(i)) + \sum_{i \in I} f_i \quad (5)$$

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. (2013), 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 (Pan et al., 2013)]** *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:

Theorem 2. *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* $R := \max_{l \in [k]} \max_{x \in C_l^\star} ||x - c_l^*||$.

We refer the reader to Section A.2 in the supplemental material for a proof of Theorem 2. To prove this, we use linear programming duality.

We've shown that SCC finds the target clustering for δ-separated data in Theorem 1. We now consider the DP-Means cost of this partition. The next proposition formally relates the DP-Facility problem to an approximate solution to the DP-Means objective.

Proposition 1. *(Pan et al., 2013) 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,*

$$DP(X, \lambda, Z^\dagger, \boldsymbol{\mu}^\dagger, K^\dagger) \leq 2 \cdot DP(X, \lambda, Z^*, \boldsymbol{\mu}^*, K^*) \quad (6)$$

Finally, we can analyze the quality of the solutions found by SCC on δ-separated data, showing that it is a constant factor approximation. Please refer to the supplementary section A.3 for the proof of the below statement:

Corollary 3. *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 the optimal solution to DP-Facility problem with* $\lambda = (\delta - 2) \cdot R$ *where* R *is defined in Theorem 2 and finds a solution that is a* 2*-approximation to the DP-Means objective with the same* λ.

3.4 Hierarchical Clustering Analysis

The sequence of partitions produced by SCC forms a hierarchical clustering of the dataset $\boldsymbol{X}$. More precisely, the union of the partitions returned by SCC is a hierarchical clustering, $\bigcup \text{SCC}(\boldsymbol{X}, \{\tau_1, \ldots, \tau_L\}, d)$.

Theorem 1 shows that SCC will recover the target clustering for data satisfying Assumption 1. We relate this to the metric used to evaluate hierarchical clusterings given a labeled flat partition, dendrogram purity (Heller and Ghahramani, 2005b; Kobren et al., 2017). We show our proposed algorithm will have perfect dendrogram purity (equal to 1) on well separated data. This follows naturally from the above theorem.

Dendrogram purity measures the quality of the flat partitions that are stored in the tree structure without requiring a single partition be selected from the tree structure. Dendrogram purity takes on values between 0 and 1 and achieves a value of 1 if and only if the tree contains the ground truth partition as a tree consistent partition (Kobren et al., 2017). Given a ground truth flat clustering $\mathbb{S}^\star$ of a dataset X, dendrogram purity of a tree structure $\mathcal{T}$, $\mathsf{DendrogramPurity}(\mathcal{T})$, as:

$$\frac{1}{|\mathcal{P}^\star|} \sum_{C^\star \in \mathbb{S}^\star} \sum_{(x_i, x_j) \in C^\star \times C^\star} \texttt{pur}(\texttt{lca}(x_i, x_j, \mathcal{T}), C^\star) \quad (7)$$

where $\mathcal{P}^\star = \{(x_i, x_j) \mid \mathbb{S}^\star(x_i) = \mathbb{S}^\star(x_j)\}$, $\mathbb{S}^\star(x)$ denotes the ground truth cluster membership of x, `lca` gives the least common ancestor of the nodes in the tree, and $\texttt{pur}(\hat{C}, C^\star) = \frac{|\hat{C} \cap C^\star|}{|\hat{C}|}$ is the (flat cluster) purity of the cluster represented by $\hat{C}$ with respect to the cluster $C^\star$. In words, dendrogram purity is the average over all pairs of points from the same ground truth cluster of the purity of the pair's least common ancestor.

Corollary 4. *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.* $\tau_i = 2^i \cdot \tau_0$. *Perfect dendrogram purity is achieved by* SCC, $\mathsf{DendrogramPurity}(\bigcup \text{SCC}(\boldsymbol{X}, d, \{\tau_1, \ldots, \tau_L\})) = 1$, *where* $\gamma = 6$ *for all metrics and* $\gamma = 30$ *for the* ℓ_2^2 *distance and* $\tau_0 \leq \min_{x, x' \in \boldsymbol{X}^2} ||x - x'||$.

Proof. There exists a round i equal to the target clustering. Each descendant of the target clusters will therefore have purity 1. Each pair of points will have a least common ancestor as one of the descendants of the nodes in round i or a node in round i. And so the purity of the least common ancestor for every pair of same-cluster points will have purity 1, leading to dendrogram purity to be 1. □

3.5 Relationship to Agglomerative Clustering

The bottom-up nature of our algorithm is reminiscent of hierarchical agglomerative clustering, another round-based algorithm in which each round merges the two subtrees with minimal distance according to a linkage function, $g : 2^{\boldsymbol{X}} \times 2^{\boldsymbol{X}} \to \mathbb{R}^+$. We make the following statement about SCC:

Proposition 2. *Let* $f : 2^{\boldsymbol{X}} \times 2^{\boldsymbol{X}} \to \mathbb{R}^+$ *be a linkage function that is symmetric and bijective,* $C_1, C_2, C_3, C_4 \subset \boldsymbol{X}$, $f(C_1, C_2) = f(C_3, C_4) \iff (C_1 = C_3 \wedge C_2 = C_4) \vee (C_1 = C_4 \wedge C_2 = C_3)$ *(i.e., the linkage between each pair of nodes is unique). Let* $\mathcal{T}$ *be the tree formed by Algorithm 2 and let* f *also satisfy reducibility, (Bruynooghe, 1978),* $\forall C, C', C'' \in \mathcal{T}$, $f(C, C') \leq \min\{f(C, C''), f(C', C'')\} \implies \min\{f(C, C''), f(C', C'')\} \leq f(C \cup C', C'')$ *then there exists a sequence of threshold* $t_1, \ldots, t_r$ *such that the tree formed by* $\bigcup \text{SCC}(\boldsymbol{X}, f, \{\tau_1, \ldots, \tau_r\})$ *is the same as* $\mathcal{T}$.

Proof. Each node in the tree, $\mathcal{T}$, produced by HAC has an associated linkage function score. For node $C \in \mathcal{T}$, we abuse notation and call this $f(C)$. We define the threshold-based rounds for SCC such that $t_1, \ldots, t_r$ to be the values, $\{f(C) + \epsilon | C \in \mathcal{T}\}$ sorted in ascending order. Because f is reducible and bijective, 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.

We can analyze HAC using the same round notation used in SCC. HAC proceeds by finding the closest two clusters according to g in the previous round's clustering $\mathbb{S}^{(i-1)}$, and joining them to form a new clustering for the next round $\mathbb{S}^{(i+1)} := \{\mathbb{S}^{(i)} \backslash \{C_a, C_b\}\} \cup (C_a \cup C_b)$ *s.t.* $C_a, C_b = \arg\min_{C_1, C_2 \in \mathbb{S}^{(i)}} d(C_1, C_2)$. The final hierarchical clustering is given as $\{\cup\ \mathbb{S}^{(i)}\}_{i=1}^{N-1}$. The round-based threshold is set to the linkage cost in the corresponding round of HAC to ensure that only a single merger is made by SCC since f is bijective. □

3.6 Worst-Case Time Analysis

For a given round, let $O(T)$ be the time required to build a 1-nearest neighbor graph over the sub-clusters

	CovType	ILSVRC (Sm.)	ALOI	Speaker	ImageNet	ILSVRC (Lg.)
$\lvert\mathbb{S}^\star\rvert$	7	1000	1000	4958	17K	1000
$\lvert\boldsymbol{X}\rvert$	500K	50K	108K	36.5K	100K	1.3M
dim.	54	2048	128	6388	2048	2048
gHHC	0.444	0.381	0.462	-	0.020	0.367
Grinch	0.430	0.557	0.504	0.48	0.065	-
Perch	**0.448**	0.531	0.445	0.372	0.065	0.207
Affinity	0.433	0.587	0.478	0.424	0.055	0.601
SCC	0.433	**0.622**	**0.575**	**0.510**	**0.072**	**0.606**

Table 1: **Dendrogram Purity** results on benchmark datasets. gHHC did not produce meaningful results on Speaker and Grinch did not scale to ILSVRC (Lg.). We find that Affinity and SCC outperform the baselines with SCC giving best performance on all datasets except one.

of a round. Cover Trees (Beygelzimer et al., 2006) allow this to be done in $O(c^{12}N \log N)$ for N elements where c is the expansion constant. 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). This leads to a running time $O(N * T)$. In practice, our algorithm can easily parallelize the computation of sub-cluster components. Further, we find that using a fixed number of rounds and simply advancing the threshold in each round experimentally works well.

4 Experiments on Clustering Benchmarks

Our analysis showed that SCC can not only recover optimal partitions of the data, but also higher quality hierarchies. In this section, we want to experimentally validate these claims on a variety of publicly available clustering benchmarks. Specifically we want to demonstrate that:

- SCC recovers more accurate hierarchical clustering than state-of-the-art methods (Section 4.1).
- SCC produces high quality flat partitions of the data (Section 4.2).
- SCC produces high quality solution to the DP-means objective (Section 4.3).

Lastly, we hope to demonstrate the scalability of SCC and so evaluate on a 30B point web-scale dataset (§5).

Datasets: We evaluate SCC on the following publicly available clustering benchmark datasets as in (Kobren et al., 2017) (Table 1): **CovType** - forest cover types; **Speaker** - i-vector speaker recordings, ground truth clusters refer each unique speaker (Greenberg et al.); **ALOI** - 3D rendering of objects, ground truth clusters refer to each object type (Geusebroek et al., 2005); **ILSVRC (Sm.) (50K subset)** and **ILSVRC (Lg.) (1.2M Images)** images from the ImageNet ILSVRC 2012 dataset (Russakovsky et al., 2015) with vector representations of images from the last layer of the Inception neural network; **ImageNet** features of the same kinds of images extended to all 17K classes present in ImageNet. In all experimental settings, we use the average linkage function with either dot products as similarity or ℓ_2^2 as distance (Appendix §B.3). To speed up computation of sub-cluster components, we use a k-NN graph. (Appendix §B.2).

4.1 Hierarchical Clustering

We first analyze the performance of SCC compared to state-of-the-art hierarchical clustering algorithms: **Affinity** (Bateni et al., 2017) - a distributed minimum spanning tree approach based on Borůvka's algorithm (Borůvka, 1926); **Perch** (Kobren et al., 2017) - 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** (Monath et al., 2019a) - 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** (Monath et al., 2019b) - a gradient-based hierarchical clustering method that uses a continuous tree representation in the unit ball. Algorithmic details and hyperparameters used are described in Appendix §B.3. Here we present results using dot product similarities and geometrically decreasing thresholds (for comparison to ℓ_2^2 see Appendix §B.3).

Since each dataset has ground truth flat clusters, we evaluate the quality of hierarchy using *dendrogram purity* (§3.4) as in previous work. 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 on this dataset from the 50K subset to the larger 1.2M point dataset.

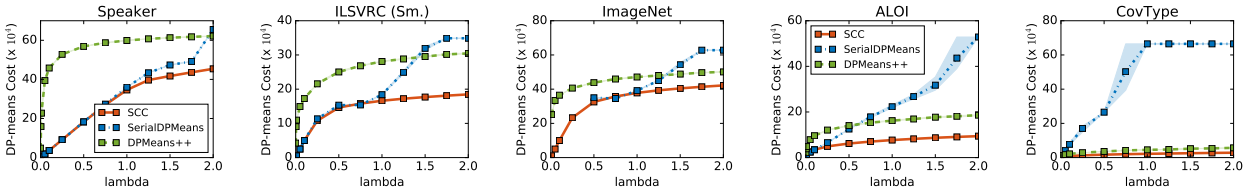

Figure 2: **DP-Means Cost** for the solutions found with a variety of methods for a range of λ values from close to 0 to 2. SCC produces lower cost solutions for different values of λ as compared to the other methods.

Dataset	SCC	Affinity	K-Means	Perch
CovType	**0.536**	**0.536**	0.245	0.230
ILSVRC (Sm.)	0.609	**0.632**	0.605	0.543
ALOI	**0.567**	0.439	0.408	0.442
Speaker	**0.493**	0.299	0.322	0.318
Imagenet	**0.076**	0.055	0.056	0.062
ILSVRC (Lg.)	0.602	**0.641**	0.562	0.257

Table 2: F1 when selecting a flat clustering with ground truth # of clusters.

We compared SCC's performance to HAC and found that SCC matches the performance of HAC while being much more scalable (Appendix §B.4). We analyse different threshold schedules in Appendix §B.5.

4.2 Flat Clustering

In this section, we empirically evaluate SCC against state-of-the-art approaches in terms of pairwise F1 and the DP-Means objective. We use the same experimental setting as in previous works (Kobren et al., 2017). 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 (Kobren et al., 2017; Manning et al., 2008).

Table 2 gives the F1 performance for each method. We observe that both SCC and Affinity outperform the previous state-of-the-art results and that the best performing approach varies from dataset to dataset. We also report the best F1 achieved in any round of our algorithm and Affinity, which is the next best performing method (see Appendix §B.6). We observe that the best F1 is better from our approach than from Affinity, highlighting that our tree structures potentially contain more high quality alternative clusterings.

4.3 Approximation of DP-Means Objective

Since our analysis section (Corollary 3) pointed to a new application of SCC, in this section we empirically evaluate these claims. In particular, 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 (Broderick et al., 2013; Jiang et al., 2012; Kulis and Jordan, 2012; Pan et al., 2013), in which data point is added to a cluster if it is within λ of that cluster center and otherwise starts a new cluster (for large datasets we use its distributed variant **OCC** (Pan et al., 2013)); **DPMeans++** (Bachem et al., 2015) an initialization-only method which performs a K-Means++ (Arthur and Vassilvitskii, 2007) 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.

Figure 2 shows for each dataset 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. Each method uses normalized ℓ_2^2 distance as the dissimilarity measure. We report the min/max/average performance over multiple runs of the SerialDPMeans and DPMeans++ algorithms with different random seeds. A more detailed explanation is provided in Appendix §C.

We observe that for each value of λ, SCC achieves the lowest DP-Means cost. We hypothesize that our improvement is due foremost to having an algorithm that discovers optimal clustering independently of λ and our algorithm's ability to explore multiple alternative partitions in the tree. For each method, if we consider the best F1 value achieved by the algorithm, SCC is usually one of the best performing methods. Since this is a completely new application of hierarchical clustering, we report a detailed analysis of SCC for DP-means objective in Appendix §C. Specifically we analyse SCC performance with different number of rounds, provide running times comparison and large scale experiments results.

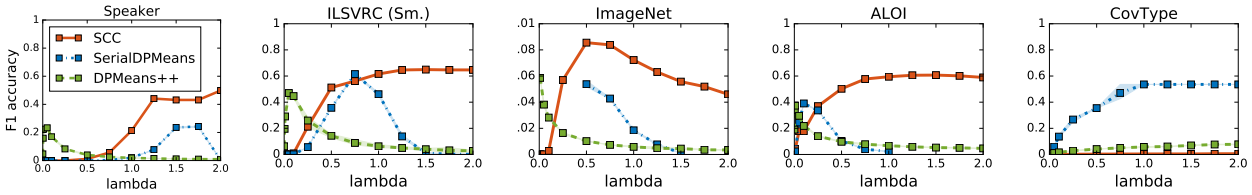

Figure 3: **Pairwise F1 Evaluation**. As each algorithm depends on λ in a different way, the settings of λ resulting in the best performance might differ between methods. We plot the performance of each method for each value of λ. When considering the best F1 achieved by each method for some value of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

5 Application: Large Scale Clustering of Web Queries

In this section, we investigate the use of SCC for clustering web queries. We run our proposed clustering approach and Affinity clustering on a 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. We do this 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. We evaluate the quality flat clusters via human annotation. We conducted an empirical evaluation with human annotators. We asked them to rate $\sim$ 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`. We found that the clusters are meaningful and better than other algorithms that work at this scale. In particular, 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.

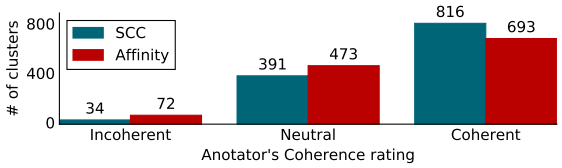

Figure 4: Human evaluation of clusters generated by SCC and Affinity

6 Related Work

A more detailed description of related work is provided in the Appendix §D. Parallel and distributed approaches for hierarchical clustering have been considered by previous work (Olson, 1995; Garg et al., 2006; Jin et al.; Dubey et al., 2014; Hu et al., 2015; Jin et al., 2015; Bateni et al., 2017; Yaroslavtsev and Vadapalli, 2018; Moseley et al., 2019; Santos et al., 2019; Dubey et al., 2020). Yaroslavtsev and Vadapalli (2018) 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 (Zhang et al., 1997; O'callaghan et al.; Widyantoro et al., 2002; Kranen et al., 2011; Nguyen et al., 2014; Kobren et al., 2017; Monath et al., 2019a,b).

Our work is closely related to the tree-based clustering methods proposed by Balcan et al. (2008) and Balcan et al. (2014). 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.

Density-based clustering algorithms such as DBSCAN (Ester et al., 1996) can also be viewed as using linkage functions as in hierarchical clustering methods. There have been several hierarchical variants of these approaches proposed including, most recently HDBSCAN* (Campello et al., 2013). Other hierarchical clustering work has used techniques to reduce the number of distance computations required to perform hierarchical clustering (Eriksson et al., 2011; Krishnamurthy et al., 2012; Shamir and Tishby, 2011; Balcan et al., 2014). Dasgupta's cost function for hierarchical clustering (Dasgupta, 2016) has widely studied (Wang and Wang, 2018; Moseley and Wang, 2017; Charikar and Chatziafratis, 2017; Cohen-Addad et al., 2017, 2018; Charikar et al., 2019b,a) including with

respect to its relationship to agglomerative clustering.

7 Conclusion

We introduce the Sub-Cluster Component algorithm (SCC) for scalable hierarchical as well as flat clustering. SCC uses a round-based, bottom-up 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.

Acknowledgements

We thank Avrim Blum and Nina Balcan for their insightful comments and suggestions.

References

N. Andrews, J. Eisner, and M. Dredze. Robust entity clustering via phylogenetic inference. *Association for Computational Linguistics (ACL)*, 2014.

D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. *Symposium on Discrete Algorithms (SODA)*, 2007.

O. Bachem, M. Lucic, and A. Krause. Coresets for nonparametric estimation-the case of dp-means. *International Conference on Machine Learning (ICML)*, 2015.

O. Bachem, M. Lucic, S. H. Hassani, and A. Krause. Approximate K-Means++ in sublinear time. *AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

M.-F. Balcan, A. Blum, and S. Vempala. A discriminative framework for clustering via similarity functions. *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 671–680, 2008.

M.-F. Balcan, Y. Liang, and P. Gupta. Robust hierarchical clustering. *The Journal of Machine Learning Research (JMLR)*, 2014.

M. Bateni, S. Behnezhad, M. Derakhshan, M. Hajiaghayi, R. Kiveris, S. Lattanzi, and V. Mirrokni. Affinity clustering: Hierarchical clustering at scale. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. *International Conference on Machine learning (ICML)*, 2006.

O. Borůvka. O jistém problému minimálním. 1926.

T. Broderick, B. Kulis, and M. Jordan. Mad-bayes: Map-based asymptotic derivations from bayes. *International Conference on Machine Learning (ICML)*, 2013.

M. Bruynooghe. Classification ascendante hiérarchique des grands ensembles de données: un algorithme rapide fondé sur la construction des voisinages réductibles. *Cahiers de l'analyse des données*, 1978.

R. J. G. B. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. *Advances in Knowledge Discovery and Data Mining*, 2013.

M. Charikar and V. Chatziafratis. Approximate hierarchical clustering via sparsest cut and spreading metrics. *Symposium on Discrete Algorithms (SODA)*, 2017.

M. Charikar, V. Chatziafratis, and R. Niazadeh. Hierarchical clustering better than average-linkage. *Symposium on Discrete Algorithms (SODA)*, 2019a.

M. Charikar, V. Chatziafratis, R. Niazadeh, and G. Yaroslavtsev. Hierarchical clustering for euclidean data. *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019b.

V. Cohen-Addad, V. Kanade, and F. Mallmann-Trenn. Hierarchical clustering beyond the worst-case. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

V. Cohen-Addad, V. Kanade, F. Mallmann-Trenn, and C. Mathieu. Hierarchical clustering: Objective functions and algorithms. *SODA*, 2018.

S. Dasgupta. A cost function for similarity-based hierarchical clustering. *Symposium on Theory of Computing (STOC)*, 2016.

A. Dubey, M. M. Zhang, E. P. Xing, and S. A. Williamson. Distributed, partially collapsed mcmc for bayesian nonparametrics. *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

K. A. Dubey, Q. Ho, S. A. Williamson, and E. P. Xing. Dependent nonparametric trees for dynamic hierarchical clustering. 2014.

B. Eriksson, G. Dasarathy, A. Singh, and R. Nowak. Active clustering: Robust and efficient hierarchical clustering using adaptively selected similarities. *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. *Conference on Knowledge Discovery and Data Mining (KDD)*, 1996.

A. Garg, A. Mangla, N. Gupta, and V. Bhatnagar. Pbirch: A scalable parallel clustering algorithm for incremental data. *International Database Engineering and Applications Symposium (IDEAS)*, 2006.

J.-M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. The amsterdam library of object images. *International Journal of Computer Vision (IJCV)*, 2005.

S. Gilpin, S. Nijssen, and I. Davidson. Formalizing hierarchical clustering as integer linear programming. *AAAI Conference on Artificial Intelligence (AAAI)*, 2013.

S. Green, N. Andrews, M. R. Gormley, M. Dredze, and C. D. Manning. Entity clustering across languages. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2012.

C. S. Greenberg, D. Bansé, G. R. Doddington, D. Garcia-Romero, J. J. Godfrey, T. Kinnunen, A. F. Martin, A. McCree, M. Przybocki, and D. A. Reynolds. The nist 2014 speaker recognition i-vector machine learning challenge. *Odyssey: The Speaker and Language Recognition Workshop*.

A. Großwendt, H. Röglin, and M. Schmidt. Analysis of ward's method. *Symposium on Discrete Algorithms (SODA)*, 2019.

K. Heller and Z. Ghahramani. Randomized algorithms for fast bayesian hierarchical clustering. *EUPASCAL Statistics and Optimization of Clustering Workshop*, 2005a.

K. A. Heller and Z. Ghahramani. Bayesian hierarchical clustering. *International Conference on Machine Learning (ICML)*, 2005b.

Z. Hu, H. Qirong, A. Dubey, and E. Xing. Large-scale distributed dependent nonparametric trees. *International Conference on Machine Learning (ICML)*, 2015.

R. Jaiswal and N. Garg. Analysis of k-means++ for separable data. *Approximation, Randomization, and Combinatorial Optimization.Algorithms and Techniques*, 2012.

K. Jiang, B. Kulis, and M. I. Jordan. Small-variance asymptotics for exponential family dirichlet process mixture models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

C. Jin, M. M. A. Patwary, A. Agrawal, W. Hendrix, W.-k. Liao, and A. Choudhary. Disc: A distributed single-linkage hierarchical clustering algorithm using mapreduce. *Data Intensive Computing in the Clouds (DataCloud)*.

C. Jin, R. Liu, Z. Chen, W. Hendrix, A. Agrawal, and A. Choudhary. A scalable hierarchical clustering algorithm using spark. *2015 IEEE First International Conference on Big Data Computing Service and Applications*, pages 418–426, 2015.

K. Kenyon-Dean, J. C. K. Cheung, and D. Precup. Resolving event coreference with supervised representation learning and clustering-oriented regularization. *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, 2018.

A. Kobren, N. Monath, A. Krishnamurthy, and A. McCallum. A hierarchical algorithm for extreme clustering. *Knowledge Discovery and Data Mining*, 2017.

P. Kranen, I. Assent, C. Baldauf, and T. Seidl. The clustree: indexing micro-clusters for anytime stream mining. *Knowledge and information systems*, 2011.

A. Krishnamurthy, S. Balakrishnan, M. Xu, and A. Singh. Efficient active algorithms for hierarchical clustering. *ICML*, 2012.

J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 1956.

B. Kulis and M. I. Jordan. Revisiting k-means: New algorithms via bayesian nonparametrics. *International Conference on Machine Learning (ICML)*, 2012.

S. Kushagra, S. Samadi, and S. Ben-David. Finding meaningful cluster structure amidst background noise. *International Conference on Algorithmic Learning Theory (ALT)*, 2016.

M. Levin, S. Krawczyk, S. Bethard, and D. Jurafsky. Citation-based bootstrapping for large-scale author disambiguation. *Journal of the American Society for Information Science and Technology*, 63(5): 1030–1047, 2012.

C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. 2008.

A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. *Conference on Knowledge discovery and data mining (KDD)*, 2000.

D. Menestrina, S. E. Whang, and H. Garcia-Molina. Evaluating entity resolution results. *VLDB*, 2010.

N. Monath, A. Kobren, A. Krishnamurthy, M. R. Glass, and A. McCallum. Scalable hierarchical clustering with tree grafting. *Knowledge Discovery & Data Mining (KDD)*, 2019a.

N. Monath, M. Zaheer, D. Silva, A. McCallum, and A. Ahmed. Gradient-based hierarchical clustering using continuous representations of trees in hyperbolic space. *Conference on Knowledge Discovery & Data Mining (KDD)*, 2019b.

B. Moseley and J. Wang. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

B. Moseley, K. Lu, S. Lattanzi, and T. Lavastida. A framework for parallelizing hierarchical clustering methods. *ECML PKDD 2019*, 2019.

T.-D. Nguyen, B. Schmidt, and C.-K. Kwoh. Sparsehc: a memory-efficient online hierarchical clustering algorithm. 2014.

L. O'callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-data algorithms for high-quality clustering. *International Conference on Data Engineering (ICDE)*.

C. F. Olson. Parallel algorithms for hierarchical clustering. *Parallel computing*, 1995.

X. Pan, J. E. Gonzalez, S. Jegelka, T. Broderick, and M. I. Jordan. Optimistic concurrency control for distributed unsupervised learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 2015.

J. Santos, T. Syed, M. C. Naldi, R. J. Campello, and J. Sander. Hierarchical density-based clustering using mapreduce. *IEEE Transactions on Big Data*, 2019.

G. W. Schwartz, Y. Zhou, J. Petrovic, M. Fasolino, L. Xu, S. M. Shaffer, W. S. Pear, G. Vahedi, and R. B. Faryabi. TooManyCells identifies and visualizes relationships of single-cell clades. *Nat. Methods*, 2020.

O. Shamir and N. Tishby. Spectral clustering on a budget. *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Y. Song, S. Liu, X. Liu, and H. Wang. Automatic taxonomy construction from keywords via scalable bayesian rose trees. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2015.

R. C. Steorts, M. Barnes, and W. Neiswanger. Performance bounds for graphical record linkage. *Artificial Intelligence and Statistics (AISTATS)*, 2017.

V. V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

D. Wang and Y. Wang. A new cost function for hierarchical cluster trees. *arXiv preprint arXiv:1812.02715*, 2018.

D. H. Widyantoro, T. R. Ioerger, and J. Yen. An incremental approach to building a cluster hierarchy. *International Conference on Data Mining (ICDM)*, 2002.

N. Yadav, A. Kobren, N. Monath, and A. Mccallum. Supervised hierarchical clustering with exponential linkage. *International Conference on Machine Learning (ICML)*, 2019.

G. Yaroslavtsev and A. Vadapalli. Massively parallel algorithms and hardness for single-linkage clustering under ℓ_p distances. *International Conference on Machine Learning (ICML)*, 2018.

I. E.-H. Yen, D. Malioutov, and A. Kumar. Scalable exemplar clustering and facility location via augmented block coordinate descent with column generation. *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

M. Zaheer, S. Kottur, A. Ahmed, J. Moura, and A. Smola. Canopy—fast sampling with cover trees. *International Conference on Machine Learning (ICML)*, 2017.

T. Zhang, R. Ramakrishnan, and M. Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1997.

Scalable Bottom-Up Hierarchical Clustering: Supplementary

A Proofs

A.1 Proof for Theorem 1

Theorem 1. *Suppose the dataset $\boldsymbol{X}$ satisfies the δ-separability assumption with respect to the clustering $C_1^*, \ldots, C_k^*$ for $\delta \geq \gamma$ then the set of partitions produced by* SCC*-algorithm with geometrically increasing thresholds i.e. $\tau_i = 2^i \cdot \tau_0$ contains the optimal partition $C_1^*, \ldots, C_k^*$ where $\gamma = 6$ for all metrics and $\gamma = 30$ for the ℓ_2^2 distance.*

Proof Sketch. Recall the assumption of δ-separability (Assumption 1) in which the maximum distance from any point to its true center is defined as $R := \max_{i \in [k]} \max_{x \in C_i^*} ||x - c_i^*||$. We make the additional assumption that the threshold of the first round, τ_0, is less than R, i.e., $\tau_0 < R$.

First, we give a proof for general metrics and then consider ℓ_2^2. 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 to $r' \leq r^\star$ 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 that the round $r^\star$ with $\mathbb{S}^{(r^\star)} = \mathbb{S}^*$ must exist.

Assume that for round $r^\star - 1$, we have *pure* sub-clusters in $\mathbb{S}^{(r^\star - 1)}$ 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.

Recall that we use c to refer to the center of cluster C. Using the triangle inequality we have that:

$$||c_i^* - c_j^*|| \leq \frac{1}{|X||Y|} \sum_{x \in X} \sum_{y \in Y} ||c_i^* - x|| + ||x - y|| + ||y - c_j^*|| \tag{8}$$

Re-arranging terms we have:

$$||c_i^* - c_j^*|| \leq \frac{1}{|X|} \sum_{x \in X} ||c_i^* - x|| + \frac{1}{|X||Y|} \sum_{x \in X} \sum_{y \in Y} ||x - y|| + \frac{1}{|Y|} \sum_{y \in Y} ||y - c_j^*|| \tag{9}$$

With further re-arrangements,

$$||c_i^* - c_j^*|| - \frac{1}{|X|} \sum_{x \in X} ||c_i^* - x|| - \frac{1}{|Y|} \sum_{y \in Y} ||y - c_j^*|| \leq \frac{1}{|X||Y|} \sum_{x \in X} \sum_{y \in Y} ||x - y|| \tag{10}$$

Since $||c_i^* - c_j^*|| \geq \delta \cdot R$, where (see Assumption 1), $R := \max_{i \in [k]} \max_{x \in C_i^*} ||x - c_i^*||$:

$$(\delta - 2) \cdot R \leq ||c_i^* - c_j^*|| - \frac{1}{|X|} \sum_{x \in X} ||c_i^* - x|| - \frac{1}{|Y|} \sum_{y \in Y} ||y - c_j^*|| \leq \frac{1}{|X||Y|} \sum_{x \in X} \sum_{y \in Y} ||x - y|| \tag{11}$$

However, we have that for $X \subset C_i^*$ and $X' \subset C_i^*$:

$$\frac{1}{|X||X'|} \sum_{x \in X} \sum_{x \in X'} ||x - x'|| \leq \frac{1}{|X||X'|} \sum_{x \in X} \sum_{x' \in X'} ||c_i^* - x|| + ||c_i^* - x'|| \leq 2R. \tag{12}$$

For $\delta \geq 6$, this indicates that there exists a τ_r, such that $2R \leq \tau_r \leq 4R$ 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 $2R$ and the doubling of $2R$. 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.

Now let's consider the case for ℓ_2^2. We update our definition of R for ℓ_2^2, $R := \max_{i\in[k]} \max_{x\in C_i^*} \|x - C_i^*\|_2^2$. Again, the algorithm begins with $\mathbb{S}^{(0)}$ set to be the shattered partition, with each data point in its own cluster, $\mathbb{S}^{(0)} = \{\{x\} | x \in \boldsymbol{X}\}$. Again, we want to show that for some round, $r^\star$ with threshold τ_r produces the ground truth partition, $\mathbb{S}^{(r^\star)} = \mathbb{S}^* = \{C_1^*, \ldots, C_k^*\}$. As before, we will show by induction that for each round prior to $r' \leq r^\star$ 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 that the round r with $\mathbb{S}^{(r^\star)} = \mathbb{S}^*$ must exist.

As before, 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 (Großwendt et al., 2019) for ℓ_2^2, we have:

$$\|c_i^* - c_j^*\|_2^2 \leq 3\left(\frac{1}{|X||Y|}\sum_{x\in X}\sum_{y\in Y}\|c_i^* - x\|_2^2 + \|x-y\|_2^2 + \|y - c_j^*\|_2^2\right) \tag{13}$$

$$\|c_i^* - c_j^*\|_2^2 \leq 3\left(\frac{1}{|X|}\sum_{x\in X}\|c_i^* - x\|_2^2 + \frac{1}{|X||Y|}\sum_{x\in X}\sum_{y\in Y}\|x-y\|_2^2 + \frac{1}{|Y|}\sum_{y\in Y}\|y - c_j^*\|_2^2\right) \tag{14}$$

Re-arranging the above:

$$\frac{1}{3}\|c_i^* - c_j^*\|_2^2 - \frac{1}{|X|}\sum_{x\in X}\|c_i^* - x\|_2^2 - \frac{1}{|Y|}\sum_{y\in Y}\|y - c_j^*\|_2^2 \leq \frac{1}{|X||Y|}\sum_{x\in X}\sum_{y\in Y}\|x-y\|_2^2 \tag{15}$$

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

$$(\frac{1}{3}\delta - 2)\cdot R \leq \frac{1}{3}\|c_i^* - c_j^*\|_2^2 - \frac{1}{|X|}\sum_{x\in X}\|c_i^* - x\|_2^2 - \frac{1}{|Y|}\sum_{y\in Y}\|y - c_j^*\|_2^2 \leq \frac{1}{|X||Y|}\sum_{x\in X}\sum_{y\in Y}\|x-y\|_2^2 \tag{16}$$

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

$$\frac{1}{|X||X'|}\sum_{x\in X}\sum_{x\in X'}\|x - x'\|_2^2 \leq 2\left(\frac{1}{|X|}\sum_{x\in X}\|c_i^* - x\|_2^2 + \leq \frac{1}{|X'|}\sum_{x\in X'}\|c_i^* - x'\|_2^2\right) \leq 4R. \tag{17}$$

For $\delta \geq 30$, we know that there exists a $4R \leq \tau_r \leq 8R$ 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^*. For the same reasons as the metric case, the result above indicates that at the round $r^\star - 1$ we will only have pure sub-clusters as no sub-clusters with points belonging to different ground truth clusters will be merged. This result indicates that the partition given by sub-cluster component from round $r^\star$, $\mathbb{S}^{(r^\star)}$ will contain every ground truth cluster in $\mathbb{S}^*$.

□

A.2 Proof for Theorem 2

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

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 : $\textsc{cost}(\alpha) \leq \textsc{Opt}(\textsc{Dual}) = \textsc{OPT}(\textsc{Primal}) \leq \textsc{cost}(C^*)$. Combined with the fact that $\textsc{cost}(\alpha) = \textsc{cost}(C^*)$, 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 Vazirani (2013)[Ch. 17].

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

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:

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

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. □

A.3 Proof of Corollary 3

Corollary 3 *Suppose a dataset $\boldsymbol{X}$, satisfies the $\delta-$separability assumption with respect to the clustering $C_1^*, \ldots, C_k^*$, for $\delta \geq \gamma$, then the set of partitions produced by $\text{SCC}(\boldsymbol{X}, \tau, d)$, with geometrically increasing thresholds τ, $\tau_i = 2^i \tau_0$, contains the optimal solution to DP-Facility problem with $\lambda = (\delta - 2) \cdot R$ where R is defined in Theorem 2 and finds a solution that is a 2-approximation to the DP-Means objective with the same λ.*

Proof Sketch. Using theorem 1 and 2, when the data satisfies the $\delta-$separation assumption, then SCC contains the optimal solution to DP-facility problem. Finally using Proposition 1, we see that this solution is within 2 factor of the DP-means solution. □

A.4 Relationship to Hierarchical Agglomerative Clustering

Hierarchical agglomerative clustering (HAC) builds a series of flat clusters like SCC in a round-based, greedy, bottom-up manner. In each round i, two clusters from the previous clustering $C, C' \in \mathbb{S}^{(i-1)}$ is merged such that the two clusters have minimum dissimilarity according to d i.e. $C, C' := \arg\min_{C_i, C_j \in \mathbb{S}^{(i-1)}} d(C_i, C_j)$. The partition at the next round is given by removing C and C' from $\mathbb{S}^{(i-1)}$ and then adding $\{C \cup C'\}$ to it i.e. $\mathbb{S}^{(i)} := \mathbb{S}^{(i-1)} - \{C, C'\} + \{C \cup C'\}$. Here $+$ and $-$ represent set addition and subtraction respectively. This process continues until a desired number of clusters is discovered, a threshold on similarity is met, or a full tree is built. Assuming d to be bijective, and non-decreasing, leads to another way of viewing this algorithm as shown in Algorithm 2.

Algorithm 2 Hierarchical Agglomerative Clustering

function HAC($\boldsymbol{X}$: Dataset , f: Linkage)
  $\mathcal{T} \leftarrow \{\{x_1\}, \ldots, \{x_N\}\}$
  while $|\mathcal{T}| > 1$ **do**
    $C_1, C_2 \leftarrow \text{argmin}_{C \neq C' \in \mathcal{V}} f(C, C')$
    $C \leftarrow C_1 \cup C_2$
    $\mathcal{T} \leftarrow \mathcal{T} \setminus \{C_1\}$
    $\mathcal{T} \leftarrow \mathcal{T} \setminus \{C_2\}$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{C\}$
  return $\mathcal{T}$

It is easy to see the relationship between HAC and SCC by looking at algorithm 2. We can set the threshold used in SCC to allow for the same sequence of mergers present in HAC to take place in SCC. This was formalized in Proposition 2.

A.5 Proof of Proposition 1

Proposition 1 (Pan et al., 2013) *Let $\boldsymbol{\mu}^*, Z^*, K^*$ 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,*

$$DP(X, \lambda, Z^\dagger, \boldsymbol{\mu}^\dagger, K^\dagger) \leq 2 \cdot DP(X, \lambda, Z^*, \boldsymbol{\mu}^*, K^*) \tag{18}$$

Proof. Pan et al. (2013) It is folklore that this holds for the K-means objective:

$$\min_{Z, \boldsymbol{\mu} \subseteq X} \sum_{i=0}^{N} \sum_{j=0}^{K} z_{i,j} ||\mathbf{x}_i - \boldsymbol{\mu}_j||^2 \leq 2 \min_{Z, \boldsymbol{\mu} \in \mathbb{R}^{N \times d}} \sum_{i=0}^{N} \sum_{j=0}^{K} z_{i,j} ||\mathbf{x}_i - \boldsymbol{\mu}_j||^2 \tag{19}$$

And so the K-means result for the optimal K^*.

$$\begin{aligned} DP(X, \lambda, Z^\dagger, \boldsymbol{\mu}^\dagger, K^\dagger) &\leq \min_{Z, \boldsymbol{\mu} \subseteq X} \left\{ \sum_{i=0}^{N} \sum_{j=0}^{K^*} z_{i,j} ||\mathbf{x}_i - \boldsymbol{\mu}_j||^2 \right\} + K^* \lambda \\ &\leq 2 \min_{Z, \boldsymbol{\mu} \in \mathbb{R}^{N \times d}} \sum_{i=0}^{N} \sum_{j=0}^{K^*} z_{i,j} ||\mathbf{x}_i - \boldsymbol{\mu}_j||^2 + K^* \lambda \leq 2DP(X, \lambda, Z^*, \boldsymbol{\mu}^*, K^*) \end{aligned} \tag{20}$$

Recall that $Z^\dagger, \boldsymbol{\mu}^\dagger, K^\dagger$ is an optimal solution and so is at most as expensive as any other solution to the DP facility problem (the first line above). □

B Experiments

B.1 Evaluation Metrics

B.1.1 Pairwise F1

Given a dataset $\boldsymbol{X}$ with ground truth clustering $\mathbb{S}^*$, we measure the pairwise F1 of a predicted clustering $\hat{\mathbb{S}}$. We compute the pairs of points that are in the same cluster in the ground truth:

$$\mathcal{P}^* = \{(x_i, x_j) \mid \ x_i, x_j \in \boldsymbol{X}, \ \exists \ C^* \in \mathbb{S}^* \text{ s.t. } \{x_i, x_j\} \subseteq C^*\} \tag{21}$$

and the pairs of points that are in the same cluster in the predicted clustering:

$$\hat{\mathcal{P}} = \{(x_i, x_j) \mid x_i, x_j \in \boldsymbol{X}, \ \exists \ C \in \hat{\mathbb{S}} \text{ s.t. } \{x_i, x_j\} \subseteq C\} \tag{22}$$

We then measure precision, recall, and F1:

$$\mathsf{Prec} = \frac{|\mathcal{P}^* \cap \hat{\mathcal{P}}|}{|\hat{\mathcal{P}}|} \quad \mathsf{Rec} = \frac{|\mathcal{P}^* \cap \hat{\mathcal{P}}|}{|\mathcal{P}^*|} \quad \mathsf{F1} = 2 \cdot \frac{\mathsf{Prec} \cdot \mathsf{Rec}}{\mathsf{Prec} + \mathsf{Rec}} \tag{23}$$

B.1.2 Dendrogram Purity

Given a dataset $\boldsymbol{X}$ with ground truth clustering $\mathbb{S}^*$, we measure the dendrogram purity of a hierarchical clustering $\mathcal{T}$ of $\boldsymbol{X}$ with respect to $\mathbb{S}^*$. 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.

Using the definition of $\mathcal{P}^*$ from Eq. 21:

$$\mathsf{DendrogramPurity}(\boldsymbol{X}, \mathcal{T}, \mathbb{S}^*) = \frac{1}{|\mathcal{P}^*|} \sum_{x_i, x_j \in \mathcal{P}^*} \mathsf{purity}(\mathsf{leaves}(\mathsf{LCA}(x_i, x_j, \mathcal{T})), \mathbb{S}^*(x_i)) \tag{24}$$

where the abuse of notation $\mathbb{S}^*(x_i)$ gives the ground truth cluster of x_i, LCA gives the least common ancestor of the pair of points in the tree, leaves gives the leaves of the node in the tree, and purity measures the cluster purity with respect to the given cluster label.

B.2 K-NN Graph Approximation

To speed up computation, we pre-compute a nearest-neighbor graph over the dataset and only use the distances defined as edges in this graph when computing distances between clusters. Referring to this set of edges in the nearest-neighbor graph as $W = \{(x, x', d) \mid x, x' \in \boldsymbol{X},\ d = \|x - x'\|_2^2\}$, we use the following approximation to Equation 1:

$$d(C_j, C_k) = \begin{cases} \frac{1}{|CR|} \sum_{x,x',d \in CR} d & \text{if } |CR| > 0 \\ \infty & \text{otherwise} \end{cases} \tag{25}$$

where $CR = \{(x, x', d) \mid (x, x', d) \in W,\ \ x \in C_j,\ x' \in C_k\}$.

B.3 Experiment Hyperparameters

Affinity clustering does not require any hyper-parameters. For Grinch, the results / settings from the original paper are used. For gHHC, we performed a grid search over the number of internal nodes used, learning rate, and child/parent regularization terms.

SCC Thresholds We experiment with two kinds of thresholds, one based on ℓ_2^2 as shown in our theoretical results and one based on using similarities (dot products/cosine similarities) between the points. We find that the two perform comparably on 3 out of the 5 datasets and that dot products work better on ALOI and Speaker (Table 4). Dot products give state-of-the-art results on all datasets and ℓ_2^2 gives state-of-the-art results on all except Speaker. We use the normalized ℓ_2^2 distance which is bounded in the range [0,4] and a range of [0,1] for dot products. For distances, we use a geometric progression with base $(\frac{M}{m})^{\frac{1}{L}}$ and $\tau_0 = m$, so our thresholds are $m \cdot (\frac{M}{m})^{\frac{i}{L}}$ for $i = 1$ to L where m is the minimum allowed pairwise distance and M is the maximum allowable pairwise distance. As $M/m > 1$, this meets the criteria of the theorems we proved. For similarities, we use the comparable geometrically increasing progression. We compare this to use a linear progression of the thresholds in Table 3. We find both schedules work quite well with linear working slightly better on the two ILSVRC datasets. Further, we compare using a fixed number of rounds, incrementing the threshold index after each round as an approximation of our method. We find that the results are nearly identical regardless of whether the threshold is incremented or not (Table 4).

B.4 Comparison to HAC

Next, we compare the efficiency and accuracy of SCC compared to HAC. We created a synthetic graph by generating 100 cluster centers and randomly selecting 30 points from a multivariate Gaussian distribution around

	CovType	ILSVRC (Sm.)	ALOI	Speaker	ImageNet	ILSVRC (Lg.)
Exponential	0.433	0.622	0.575	0.510	0.0722	0.606
Linear	0.433	0.641	0.572	0.491	0.0798	0.591

Table 3: **Comparison of Round Threshold Schedules**. We run SCC with two settings of round thresholds, exponentially decaying and linear schedules. We use 30 rounds in each case. We report the dendrogram purity of each and observe that the performance of the exponential is typically better.

Metric	Fixed # Rounds	CovType	ILSVRC (Sm.)	ALOI	Speaker	ImageNet
ℓ_2^2	Y	0.437	0.617	0.537	0.446	0.076
ℓ_2^2	N	0.443	0.626	0.554	0.455	0.077
$x_i^T x_j$	Y	0.438	0.631	0.586	0.524	0.074
$x_i^T x_j$	N	0.438	0.632	0.588	0.524	0.075

Table 4: **Distance/Similarity Metric Comparison & Fixed Number of Rounds**. We observe that ℓ_2^2 and $x_i^T x_j$ give comparable results on 3 out of 5 datasets and $x_i^T x_j$ improves results on two datasets. We observe that using a fixed number of rounds with one round per threshold does not impact performance.

each center. For each point in the graph, we selected k nearest neighbors and these nearest neighbours have edges between them. In our experiments, we varied the number of nearest neighbour in the sparsified nearest neighbor graph.

For the ground truth, we assigned the 30 points generated around each center in the original graph to the same cluster. Thus, our ground truth consisted of 100 clusters of 30 points each. The first metric we computed was cluster purity. For each cluster generated by the clustering algorithm, we counted the ground truth class that had the most members present in that cluster. We then added up all these counts and divided by the total number of points (3000).

In Figure 5, we show the flat cluster purity, running time, and pairwise F1 performance. We observe that the running time of SCC is orders of magnitude less than HAC (especially when a large number of nearest neighbors is used) while both methods achieve the same, nearly perfect solution. We observe that SCC achieves its benefit in speed in that it can merge multiple nodes in a single round requiring only a handful of rounds while HAC is limited to only merging a pair per round.

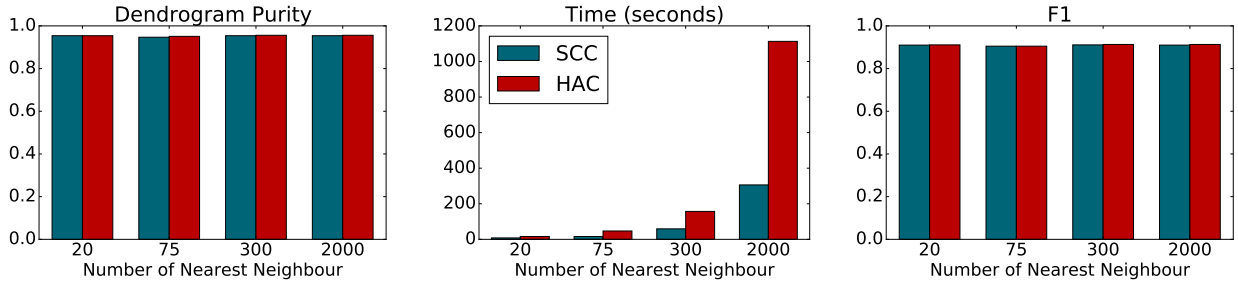

Figure 5: **Comparison to HAC**: We compare the speed and performance of our method compared to HAC, on synthetic dataset. From left to right we report, dendrogram purity, time till completion and F1 accuracy.

B.5 Analysing round threshold schedule

In Table 3, we compare the performance of our approach using exponentially decaying round parameters to using the linear schedule of rounds. We find that on most datasets the performance is typically quite similar, but find some improvements using the exponential scheme.

	CovType	ILSVRC (Sm.)	ALOI	Speaker	ImageNet	ILSVRC (Lg.)
Affinity	0.536	0.632	0.465	0.3141	0.055	0.641
SCC	*0.536*	*0.654*	*0.605*	*0.526*	*0.081*	*0.664*

Table 5: **Best F1** The best F1 achieved by the methods for any number of clusters. We observe that the best F1 achieved by SCC is consistently best.

Tea Recipes	Tennis Strategy	Electric Piano	Camping in Cupertino
tea drinks	playing strategies in tennis	digital piano price	campground cupertino
tea recipes	understanding tennis tactics	electric piano sale	cupertino camping
tea drinks to make at home	tennis strategy names	electric piano prices	campsites near cupertino
fancy tea recipes	baseline tactics	yamaha electric piano small	cupertino area campgrounds
black tea flavors and recipes	tennis strategy angles	best digital piano ebay	camping cupertino ca

Table 6: Example Fine-grained Query Clusters Discovered by SCC

B.6 Flat clustering Comparison to Affinity

In the hierarchical clustering and flat clustering experiments, we observe that SCC and affinity clustering were typically the best performing methods. We observe that the flat clustering produced when using the ground truth number of clusters is not always the best in terms of F1. We report in Table 5 the results of the best F1 of any flat partition produced in a round of Affinity and SCC.

B.7 Large Scale Clustering of Web Queries

Example clusters and hierarchies found on the 30 billion user query dataset are shown in Table 6 and Figure 6.

C DP-means

In this section we start with the description of the DP-means objective and then give details of all experiments related to evaluating SCC for DP-means.

Definition 7. (DP-Means) (Jiang et al., 2012) *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:*

$$DP(\boldsymbol{X}, \lambda, \mathbb{S}) = \sum_{C_l \in \mathbb{S}} \sum_{x \in C_l} ||x - c_l||^2 + \lambda |\mathbb{S}|. \tag{26}$$

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 $\mathbb{S}$.*

C.1 Explaining results shown in Section 4.3

Before giving details of other experiments we provide a detailed explanation of experiment results shown in Section 4.3. For each method, we recorded the assignment of points to clusters given by inference. We then use this assignment of points to flat clusters to produce a DP-Means cost. Note that this means that while methods like exemplar based clustering method would produce a data point as the cluster center, we instead replace this representative with the empirical mean of the cluster points. This strictly improves the DP-Means objective score (see Proposition 1). Each method uses normalized ℓ_2^2 as the dissimilarity measure.

Figure 2 shows for each dataset the DP-Means objective as a function of the value of the parameter λ. Each method uses normalized ℓ_2^2 distance as the dissimilarity measure. We use the following range of lambda values 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.

Noticeably, each algorithm depends on the value of λ in a different way. SerialDPMeans relies directly on λ to determine membership of clusters. This means that for values of λ greater than the maximum distance between

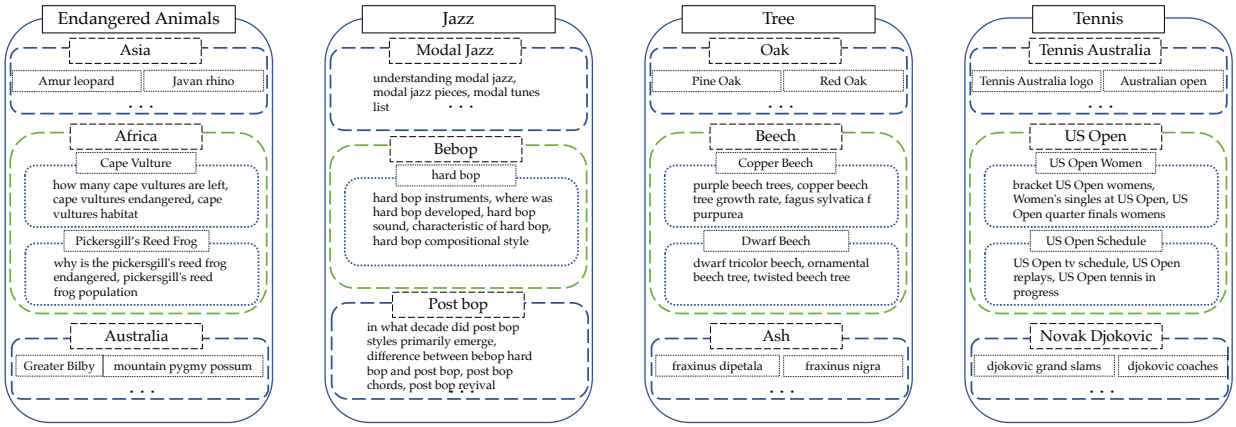

Figure 6: Hierarchy inferred on 30 billion user queries using SCC. We represent the hierarchy using rectangular boxes. The root with header is represented by the outer rectangular box (solid line). The second level of the hierarchy, with header, is shown in dashed ($- -$) rectangle withing the outer 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.

points the algorithms will put all points into the same cluster and return the clustering with the entire dataset with a single cluster. DPMeans++ has similar behavior, but behaves more globally with respect to the λ as its stopping condition depends on the sum of the distance of every point to cluster assignment. Our method, SCC, on the other hand constructs a series of candidate solutions to DP-Means independent of λ and then selects amongst these clusterings, the one which optimizes the DP-Means objective for a given value of λ.

F1 results Each dataset has a ground truth flat partition, defined for purposes of using the datasets for classification. We evaluate the ability of clustering algorithms to recover this ground truth flat partition. We measure the quality of a predicted partition by the pairwise precision/recall/F1 clustering metric (for detail see Manning et al. (2008); Menestrina et al. (2010) or our §B.1).

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. We run each method with the same values of λ as in the DP-Means objective evaluation and record the F1 score of the clustering for each. We show this in Figure 3. For each method, if we consider the best F1 value achieved by the algorithm, SCC is usually one of the best performing methods. We do observe that both DPMeans++ and SCC perform poorly on the CovType dataset. This is because both predict many more clusters than are needed for the dataset (See §C.6).

C.2 Comparison to LowrankALBCD (Yen et al., 2016)

In this experiment, we compare the performance of our method and LowrankALBCD (Yen et al., 2016). The parameter settings described by the authors in their paper as well as the code provided by the authors use values of lambda at the scale of $\lambda = 0.01 \cdot N$. Our experiments (following those in (Yen et al., 2016)) use normalized ℓ_2^2 distance. Any value of λ greater than the maximum pairwise distance which in the case of ℓ_2^2 is 4, will result in methods such as SerialDPMeans and OCC-DP-Means necessarily giving solutions of every data point being placed in the same cluster. We evaluated LowrankALBCD on CovType, ALOI, Speaker, and ILSVRC (Small). We tried to 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. And so, we instead 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. For instance, ALOI and ILSVRC (Sm.) have 1000 ground truth clusters and these values of λ are producing less than 30 clusters. This

Dataset	Method	Running Time (Seconds)	Best Pairwise F1 Achieved
Speaker	SCC (NN Graph Const. + SCC Alg.)	139.66 + 4.38	0.498
	OCC (50 Iterations)	403.36	0.243
	DPMeans++ (Center picking)	166.86	0.240
ILSVRC (Sm.)	SCC (NN Graph Const. + SCC Alg.)	670.01 + 3.25	0.646
	OCC (50 Iterations)	487.370	0.614
	DPMeans++	373.90	0.472
Imagenet	SCC (NN Graph Const. + SCC Alg.)	2242.31 + 7.62	0.0855
	OCC (50 Iterations)	2904.04	0.056
	DPMeans++ (Center picking)	645.17	0.059
ALOI	SCC (NN Graph Const. + SCC Alg.)	35.21 + 7.93	0.607
	OCC (50 Iterations)	171.33	0.392
	DPMeans++ (Center picking)	72.69	0.536
CovType	SCC (NN Graph Const. + SCC Alg.)	25.40 + 66.75	0.536
	OCC (50 Iterations)	60.67	0.536
	DPMeans++ (Center picking)	3.37	0.536
ILSVRC (Lg.)	SCC (NN Graph Const. + SCC Alg.)	26119.56 + 49.54	0.526
	OCC (2 Iterations)	7704.47	0.439
	DPMeans++ (Center picking)	36227.65	0.476

Table 7: **Running Times**. Each method is executed with a variety of different λ values. SCC needs to be executed only one time to build a series of partitions from which flat clusterings can be extracted. Other methods are re-executed for each value of λ. We report the *slowest* running time for any value of λ for each method and dataset. We also report the best pairwise F1 achieved by the method on the dataset for some value of λ. We observe that SCC is efficient and achieves the highest F1 clusterings.

leads to extremely low pairwise F1 scores. Similarly, the Speaker dataset has around 5000 ground clusters, yet these values of λ produce less than 10 clusters. As reported in the main body of the paper, SCC is able to achieve high F1 scores for lower values of λ when there are fine-grained clusters.

C.3 Timing Experiments

We give timing results comparing the SCC to DPMeans++ and the parallelized version of SerialDPMeans, Optimistic Concurrency Control DPMeans (OCC) (Pan et al., 2013). As seen in Figure 3, a search over multiple values of λ is necessary to achieve high quality clustering performance. We compare methods based on their *slowest* clustering performance across values of λ and report this in Table 7. We report for SCC the time it takes to compute the nearest neighbor graph and the time it takes to run SCC on the dataset restricted by the nearest neighbor graph. We do not report the time it takes to evaluate the cost at each level of the tree structure since this is trivially parallelizable for the number of rounds needed for SCC (< 200). We use Cover Tree for nearest neighbor computation. We perform exact K nearest neighbors and note that time could be improved if approximate nearest neighbors were used instead. We remind the reader that SCC needs to be only run one time to produce clusterings for any value of λ. For Optimistic Concurrency Control (OCC), we report the time it takes to perform 50 iterations (we observed improvements in DPMeans cost to this number of iterations) except in the case of the largest (1.2M) point ILSVRC (Lg.) dataset for which we find running 50 iterations with any value of λ less than 0.75 prohibitively expensive (more than 10 hours for 2 iterations). We use 12 threads for OCC and DPMeans++. We observe that SCC is typically quite competitive in terms of running time. On some datasets, DPMeans++ is fastest, particularly when it discovers a smaller number of clusters. However we see that given the nearest neighbor graph SCC is the fastest method. Furthermore, we observe that even in cases where SCC is not the fastest in total running time, SCC finds the best clustering in terms of pairwise F1.

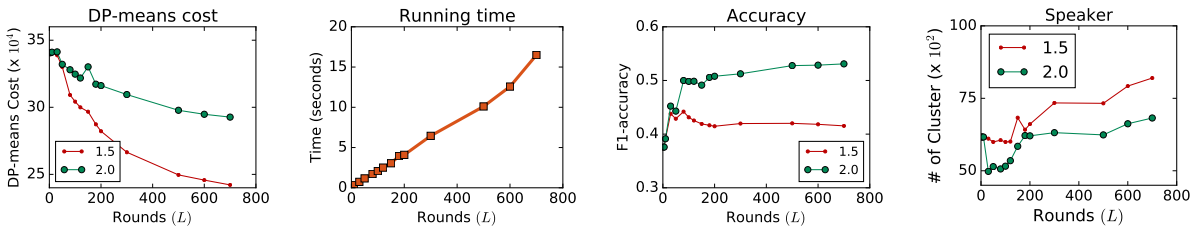

Figure 8: **Number of Rounds** The impact on number of rounds (L) in SCC algorithm, on DP-mean cost, running time, F1, and number of clusters on Speaker dataset. For DP-means and F1, we report for $\lambda = \{1.5, 2\}$.

C.4 Large scale experiments

In this experiment, we scale each of the methods for DP-Means to the largest dataset, a dataset of 1.2 million imagenet images (ILSVRC (Lg.)). We run SerialDPMeans using its parallel implementation Optimistic Concurrency Control (Pan et al., 2013). We run SCC, SerialDPMeans/OCC, DPMeans++ for the same values of λ used in the other experiments. We additionally run SCC and DPMeans++ for larger values of λ as one might expect is necessary for a larger dataset.

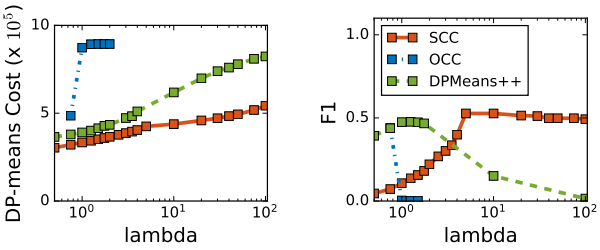

Figure 7: DP-means and F1 accuracy for ILSVRC (Lg.) dataset

SerialDPMeans/OCC can only be run with $\lambda < 4.0$ since $\lambda > 4.0$ will place all points in the same cluster (the maximum ℓ_2^2 distance between points is 4). We find that running SerialDPMeans/OCC for 50 iterations as in the other datasets is prohibitively time consuming. We find that it takes longer than 10 hours to run more than 2 iterations of OCC, for lambda values less than 0.75. We report the results after the method had finished 2 iterations in the time limit. We observe that the lambda value of 0.75 gives a reasonable F1 score for SerialDPMeans/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.

C.5 Number of Rounds - DP-Means

Rounds (L) After fixing the maximum value of the threshold and the mimimum, the number of rounds (L), determine how τ_i changes. To analyse the performance of SCC algorithm with number of rounds we increased L from 2 to 700. In Figure 9, we plot DP-means cost, the total distance of points from assigned clusters, the number of clusters discovered, the time taken and the F-1 accuracy vs number of rounds, for two values of $\lambda = \{1.5, 2\}$.

As seen in Figure 9, across datasets, the DP-means cost decreases with number of rounds but then the decrease tapers off around 100 - 200 rounds. Moreover, the number of clusters mostly increases with the number of rounds, decreasing the distance of points to the cluster centers. As expected the number of clusters found using higher value of λ (2.0), is always lower than the number of clusters found using lower value of λ (1.5). The time taken has a linear dependence on the number of rounds. The F1-accuracy numbers also increase with the number of rounds and are somewhat stable beyond 100-200 rounds. All this suggests that using 100-200 rounds is ideal for real world scenario.

C.6 CovType Performance

CovType is a dataset with 500K points belonging to just 7 different clusters. Because of the large number of points, we observe that for small values of λ, SCC produces far too many clusters. SerialDPMeans does not have the same problem as λ directly acts as a distance threshold giving cluster membership. We observe that

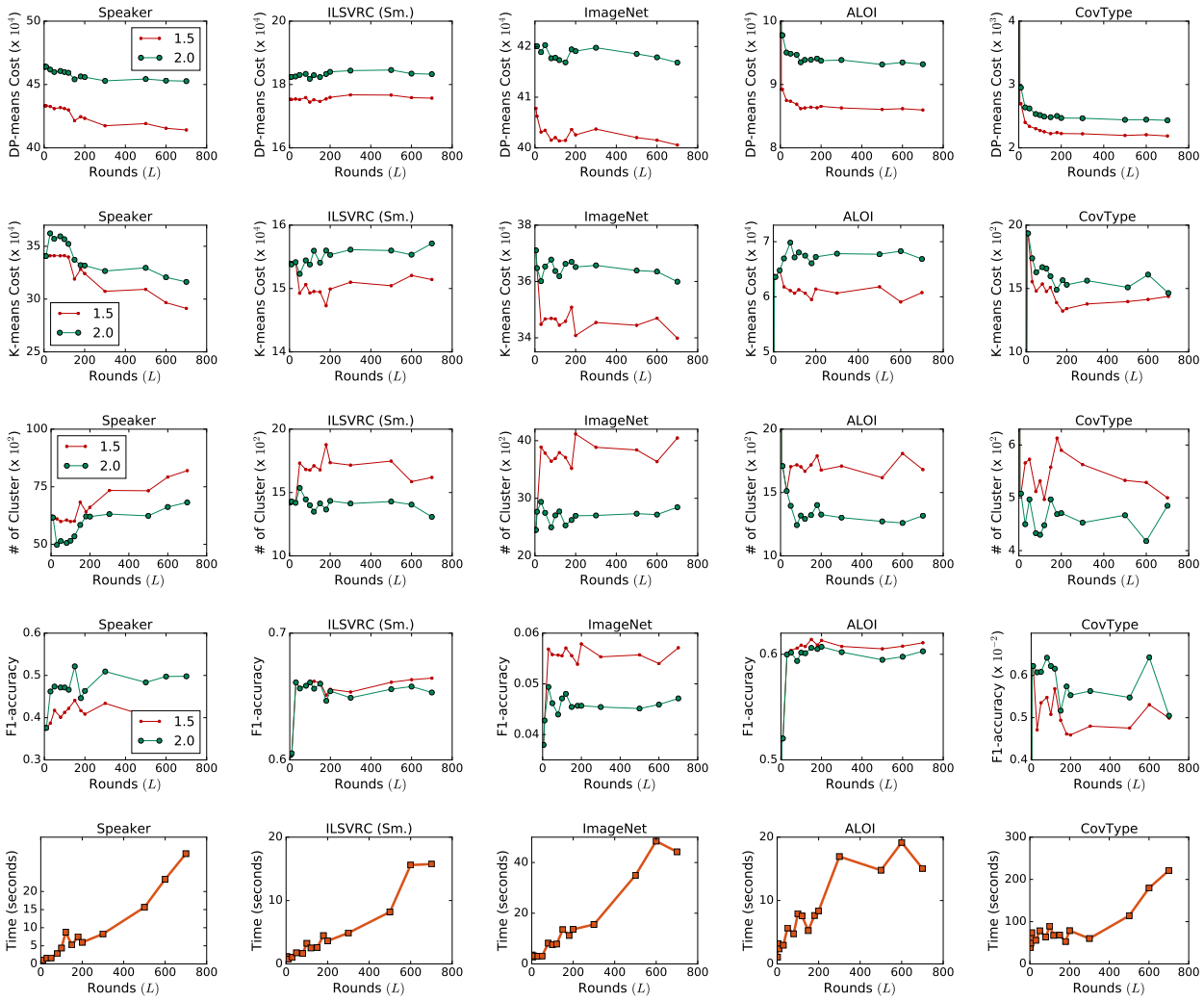

Figure 9: **Ablation Study on Number of Rounds Used**. The impact on the number of rounds (L) in SCC algorithm, on DP-mean cost, distance of points from cluster center (K-means cost), number of clusters, F1 and running time for the different datasets. We report numbers for $\lambda = \{1.5, 2\}$. Note that the running time for all λ is the same for SCC, as shown in the last row.

for larger values of λ the F1 performance of SCC can be improved. However, we do not find a comparison of F1 score particularly meaningful as we observed the highest F1 value (0.536) when all points are placed in a single cluster.

D Related Work

Parallel and distributed approaches for hierarchical clustering have been considered by previous work (Olson, 1995; Garg et al., 2006; Jin et al., 2015; Bateni et al., 2017; Yaroslavtsev and Vadapalli, 2018; Moseley et al., 2019; Santos et al., 2019). Yaroslavtsev and Vadapalli (2018) 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 (Zhang et al., 1997; O'callaghan et al.; Widyantoro et al., 2002; Kranen et al., 2011; Nguyen et al., 2014; Kobren et al., 2017; Monath et al., 2019a,b). 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 (Kobren et al., 2017) and GRINCH (Monath et al., 2019a) add points next to the nearest neighbor node

in the tree structure and perform tree re-arrangements in the form of rotations and subtree grafts. Widyantoro et al. (2002) use a bottom up incremental clustering approach. Kranen et al. (2011) build an adaptive index structure for streaming data that allows for weighting points based on their recently of arrival.

Our work is closely related to the tree-based clustering methods proposed by Balcan et al. (2008) and Balcan et al. (2014). 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.

While this paper has focused on linkage-based hierarchical clustering in general, density-based clustering algorithms like DBSCAN (Ester et al., 1996) correspond to specific linkage functions. There have been several hierarchical variants of these approaches proposed including, most recently HDBSCAN* (Campello et al., 2013).

Other work has use techniques to reduce the number of distance computations required to perform hierarchical clustering (Eriksson et al., 2011; Krishnamurthy et al., 2012; Shamir and Tishby, 2011; Balcan et al., 2014). Krishnamurthy et al. (2012) 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 McCallum et al. (2000); Zaheer et al. (2017); Song et al. (2015).

A variety of objective functions have been proposed for hierarchical clustering. Some work has use integer linear programming to perform hierarchical agglomerative clustering (Gilpin et al., 2013). Notably, Dasgupta's cost function (Dasgupta, 2016) has widely studied (Wang and Wang, 2018; Moseley and Wang, 2017; Charikar and Chatziafratis, 2017; Cohen-Addad et al., 2017, 2018; Charikar et al., 2019b,a). The cost, which is defined as the sum over all pairs of datapoints of the similarity of the pair multiplied by the number of leaves of the least common ancestor of the pair, prefers trees which place similar points together near the leaves of the structure.