# Hierarchical Clustering of Data Streams: Scalable Algorithms and Approximation Guarantees

Anand Rajagopalan [1] Fabio Vitale [2] Danny Vainstein [3] Gui Citovsky [1] Cecilia M. Procopiuc [1] Claudio Gentile [1]

## Abstract

We investigate the problem of hierarchically clustering data streams containing metric data in $\mathbb{R}^d$. We introduce a desirable invariance property for such algorithms, describe a general family of hyperplane-based methods enjoying this property, and analyze two scalable instances of this general family against recently popularized similarity/dissimilarity-based metrics for hierarchical clustering. We prove a number of new results related to the approximation ratios of these algorithms, improving in various ways over the literature on this subject. Finally, since our algorithms are principled but also very practical, we carry out an experimental comparison on both synthetic and real-world datasets showing competitive results against known baselines.

## 1. Introduction

Hierarchical clustering (HC) is a fundamental tool of data analysis by which data items are grouped together based on some notion of (semantic) similarity working at different levels of granularity. The goal of a HC algorithm operating on a dataset $X$ is then to construct a tree whose leaves host the data items in $X$, and whose internal nodes encode subsets of $X$ (that is, the clusters) at increasing levels of resolution from root to leaves. Applications are ubiquitous, ranging from phylogeny (e.g., (Eisen et al., 1998)) to data mining and information retrieval (e.g., (Manning et al., 2008)), to social network analysis (e.g., (Gilbert et al., 2011)), and beyond.

In practice, HC methods are often deployed in data-intensive applications, where massive datasets have to be hierarchically organized and connected to data-acquisition pipelines within highly dynamic (and typically non-stationary) environments. In these contexts, it is crucial to devise adaptive HC solutions that enable the handling of massive data streams in a robust and efficient manner. In HC for data streams, it is a common desideratum to have a fast way of updating the hierarchy with the newly acquired data without recomputing everything from scratch. Yet, at the same time, we would like to do so without exposing ourselves to unexpected temporal behaviors of the data stream that skew the hierarchy towards undesirable configurations.

**Contributions.** In this paper, we present the general algorithmic framework of hyperplane-based HC for data streams containing metric data. The (randomized) algorithms originating from this framework are purely geometric algorithms that can interchangeably be described as *batch* HC solutions (the dataset $X$ is given up front in its entirety) or *dynamic* (aka sequential) HC solutions (the data points in $X$ are disclosed one by one or in small batches). Crucially, within our framework, the two solutions turn out to be *statistically equivalent*, in that the statistical properties of the trees computed in the batch mode are the same as those for trees computed in the sequential mode. This means that the specific ordering of data by which the tree structure is grown does not affect the properties of the final tree, thereby giving our HC solutions a desirable robustness. We call this the *sequential property* of HC algorithms (see Section 2 for a formal definition). Moreover, the computed hierarchy is fully online in the sense that points are inserted as siblings of existing nodes, without changing the tree topology.

*Quality measures:* In order to evaluate the quality of our HC solutions, we follow the recent trend initiated by Dasgupta (2016), and further developed by a number of more recent works (e.g., (Charikar & Chatziafratis, 2017; Cohen-addad et al., 2019; Charikar et al., 2019b; Cohen-addad et al., 2019; Naumov et al., 2020; Alon et al., 2020; Vainstein et al., 2021)), who framed the HC problem as a combinatorial optimization problem over hierarchical structures against exogenous pairwise similarity/dissimilarity information on individual data points. In practice, as emphasized, e.g., by Charikar et al. (2019b); Naumov et al. (2020), data are often described by feature vectors, so that this pairwise information can be naturally delivered by the underlying

[1]Google Research, NY, USA [2]Lille University and INRIA Lille, France [3]Tel-Aviv University, Israel. Correspondence to: Anand Rajagopalan <anandbr@google.com>, Fabio Vitale <fabio.vitale@inria.fr>.

*Proceedings of the 38th International Conference on Machine Learning*, PMLR 139, 2021. Copyright 2021 by the author(s).

metric structure (e.g., $\ell_1$ or $\ell_2$) where data lies. In this paper we consider several quality measures: CKMM Revenue (Cohen-addad et al., 2019), MW Revenue (Moseley & Wang, 2017), Dasgupta Cost (Dasgupta, 2016) and MW Cost (a natural dissimilarity metric that, to our knowledge, has not been investigated before).

From the general hyperplane-based HC framework, we focus on two scalable algorithms: the **Random Cut Tree (RCT)** algorithm, originally proposed by Guha et al. (2016), and the **Uniform Radial Random Hyperplane** (**URRH**) algorithm, which is novel. We prove a number of new approximation guarantees for these two algorithms, including the following:

(i) For the CKMM Revenue, we prove that RCT (resp. URRH) has a *0.9-approximation ratio* when using the $\ell_1$ (resp. $\ell_2$) distances as dissimilarities, which improves on the 0.74 approximation ratio recently shown by Naumov et al. (2020) for a computationally more demanding algorithm;

(ii) For MW Revenue, when the similarity weights are defined through inverse $\ell_1$-distances $1/||x-y||_1$ (resp. $\ell_2$-distances) , we provide a *0.8-approximation ratio* for RCT (resp. URRH), while for the $\ell_2$ Gaussian kernel similarity, URRH improves (Figure 2) on the approximation guarantee contained in (Charikar et al., 2019b);

(iii) When similarity weights are defined in terms of $\ell_2$-distances in $\mathbb{R}^d$, we show that URRH achieves an approximation of $\frac{1}{3} + O(1/d^3)$ for the MW Revenue, yielding the first $> \frac{1}{3}$ approximation for non-constant $d$ (in contrast to Charikar et al. (2019b) and Vainstein et al. (2021)).

(iv) For the MW Cost, we provide a *2-approximation ratio* for RCT (resp. URRH) in the case when dissimilarities are defined as $\ell_1$-distances (resp. $\ell_2$-distances).

We refer the reader to Table 2 in Section 4 for a summary of results on RCT as well as to Theorems 5.2 and 5.3 in Section 5 for the approximation guarantee of URRH.

Finally, we perform preliminary experiments on both synthetic and real-world datasets, where we compare RCT and URRH to known dynamic HC baselines. These experiments show that, in terms of approximation quality, our algorithms are on par with these baselines when the cluster separation in the data is moderate, tend to outperform the baselines in the presence of high level of noise (harder clustering instances), and vice versa for low noise levels.

**Related work.** Most of the existing hierarchical clustering solutions for streaming data are heuristics, e.g., Rodrigues et al. (2006); Loewenstein et al. (2008); Nguyen et al. (2014). The approach in Kobren et al. (2017) optimizes for a different quality measure, the so-called dendogram purity. In our experimental evaluation, we include three of the most popular previous approaches: BIRCH, PERCH and GRINCH.

Introduced by Zhang et al. (1996), BIRCH is a dynamic algorithm that maintains a hierarchical clustering tree such that every internal node contains the metadata corresponding to its subcluster (Clustering-Feature). PERCH (Kobren et al., 2017) is a dynamic clustering algorithm that performs rotations to enhance subtree purity and balance. GRINCH (Monath et al., 2019) is a dynamic clustering algorithm that employs two key operations, rotate and graft, which respectively handle local and global rearrangements.

Orthogonally, many theoretical results exist for the batch case, wherein the dataset is given up front in its entirety. This line of work may be divided into *general* instances and *metric-based* instances.

*General weights.* Paving the way, Dasgupta (2016) first framed the HC problem as an optimization problem. Currently the best known approximation to the Dasgupta Cost is achieved through iterative sparsest cut, yielding an approximation factor of $O(\sqrt{\log n})$ (Charikar & Chatziafratis, 2017; Cohen-addad et al., 2019). Furthermore, a constant approximation does not exist assuming the Small Set Expansion (SSE) Hypothesis (Charikar & Chatziafratis, 2017). Moseley & Wang (2017) introduced a maximization variant of the problem. Under this objective (MW Revenue), state of the art results include a 0.585 approximation factor (Alon et al., 2020). Dissimilarity information is considered in Cohen-addad et al. (2019) (CKMM Revenue). In this case, the best approximation is known to be 0.74 (Naumov et al., 2020). We note that both objectives (MW and CKMM) are APX-hard assuming the SSE Hypothesis.

*Metric-based weights.* The MW objective has also been studied in connection to metric-based instances. Charikar et al. (2019b) considered the case where the similarity weights are defined through a non-increasing function $g : \mathcal{R} \to [0, 1]$ applied to pairwise distances defined via a metric. Vainstein et al. (2021) showed that if $g$ admits certain "nice properties" and the metric has constant doubling dimension then there exists a $1 - \epsilon$ approximation for any constant $\epsilon > 0$. We note however, that this algorithm's running time is double exponentially dependent on $\frac{1}{\epsilon}$. Furthermore, in order to handle data streams the tree must be computed from scratch at each new insertion. Thus, the algorithm is impractical in many real-world applications, especially in dynamic settings.

We note that these objectives have been researched in many more flavours: Structural constraints (Chatziafratis et al., 2018), HC through hyperbolic embeddings (Chami et al., 2020), and many others (Wang & Moseley, 2020; Charikar et al., 2019a; Chatziafratis et al., 2020a).

Finally, it is worth mentioning that there has been work on

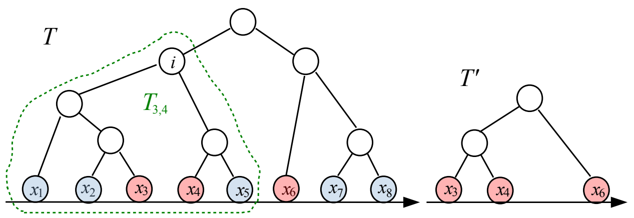

**Figure 1:** Left: A hierarchical clustering $T$ on the set $X = \{x_1, \ldots, x_8\}$ made up of eight points laying on a line. Internal node $i$ encodes the cluster $\{x_1, \ldots, x_5\} \subseteq X$. Tree $T_{1,4}$ is the subtree rooted at $\mathrm{lca}_T(x_1, x_4) = i$, with $|T_{1,4}| = 5$. Notice that $i = \mathrm{lca}_T(x_j, x_k)$ for $j = 1, 2, 3$, and $k = 4, 5$. Right: $T'$ is the restriction of $T$ to triplet $\{x_3, x_4, x_6\}$.

clustering of data streams when one is instead interested in optimizing flat clusters of the data. Most relevant to us is the work by Schmidt & Sohler (2019) on *hierarchical diameter k-clustering*. Here, the data are first hierarchically clustered and then each pointwise flat cluster (corresponding to a cut in the hierarchical clustering tree) is considered. The goal is to simultaneously minimize the median of each resulting flat clustering. Though operating in a dynamic setting, the resulting algorithms are of different flavor than ours due to the significant difference in objectives. Further work on flat clustering of data streams includes, e.g., Lin et al. (2010); Chen (2009).

## 2. Preliminaries and basic notation

In its standard formulation, in the HC problem we are given a set[1] of $n$ items $X = \{x_1, \ldots, x_n\}$, and the goal is to construct a (binary) tree $T$ whose leaves are the $n$ items above so as to optimize some criterion. The tree encodes a clustering of $X$ at different levels of granularity. Each internal node $i$ of $T$ can be naturally viewed as the cluster (that is, the subset of $X$) made up of all the leaves in the subtree rooted at $i$. Given leaves $x_i$ and $x_j$ of $T$, we denote by $T_{i,j}$ the subtree rooted at the lowest common ancestor $\mathrm{lca}_T(x_i, x_j)$ of $x_i$ and $x_j$ in $T$, while $|T_{i,j}|$ denotes the number of leaves in $T_{i,j}$. See Figure 1 (left) for a simple illustration.

Following the recent trend in the HC literature, we cast the problem as an optimization problem (e.g., (Dasgupta, 2016; Moseley & Wang, 2017; Wang & Wang, 2018; Cohen-addad et al., 2019; Alon et al., 2020; Charikar et al., 2019a;b; Chatziafratis et al., 2020b; Wang & Moseley, 2020; Naumov et al., 2020)), where an objective function is constructed that only depends on information about the pairwise *similarity* or pairwise *dissimilarity* over the points in $X$. Moreover, we assume the data are described by suitable feature vectors, so that the items in $X$ lie within a suitably bounded subset of $\mathbb{R}^d$, for some input dimension $d \geq 1$, and the pairwise information is then a function of the feature vectors alone. This pairwise information may be encoded either through a *similarity* function $\mathrm{sim} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, for instance: $\mathrm{sim}(x_i, x_j) = x_i \cdot x_j$, the inner product between $x_i$ and $x_j$; or $\mathrm{sim}(x_i, x_j) = \exp\left(-||x_i - x_j||_2^2/2\sigma^2\right)$, the Gaussian kernel between $x_i$ and $x_j$ at scale $\sigma > 0$; or $\mathrm{sim}(x_i, x_j) = D - ||x_i - x_j||$, where $||\cdot||$ is some norm over $\mathbb{R}^d$, and $D$ is some notion of diameter of the set of points $X$; or through a *dissimilarity* function, $\mathrm{dissim} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, e.g., $\mathrm{dissim}(x_i, x_j) = ||x_i - x_j||$. A meaningful definition that connects $\mathrm{sim}(\cdot,\cdot)$ to $\mathrm{dissim}(\cdot,\cdot)$ is simply $\mathrm{sim}(x_i, x_j) = -\mathrm{dissim}(x_i, x_j)$.

Notation-wise, when the pairwise information represents similarity, we collectively denote it by a weight matrix $[w_{i,j}]_{i,j=1}^n$; when it represent dissimilarity we use instead matrix $[d_{i,j}]_{i,j=1}^n$. A number of objectives can then be defined, depending on whether we want to maximize similarity or minimize dissimilarity (see Table 1 for reference). The MW Revenue (Moseley & Wang, 2017) of tree $T$ on similarity matrix $[w_{i,j}]$, denoted here as[2] $\mathrm{Rev}_S(T)$, is defined as

$$\mathrm{Rev}_S(T) = \sum_{i,j=1}^{n} w_{i,j}(n - |T_{i,j}|) ,$$

and the goal is to find a tree $T$ such that the approximation ratio $\mathrm{Rev}_S(T)/\mathrm{Opt}_{\mathrm{Rev}_S}$ is as large as possible, where $\mathrm{Opt}_{\mathrm{Rev}_S} = \max_T \mathrm{Rev}_S(T)$ is the largest ("optimal") possible revenue any tree $T$ can achieve on the given $[w_{i,j}]$. The other objectives (CKMM Revenue (Cohen-addad et al., 2019), Dasgupta Cost (Dasgupta, 2016), and MW Cost[3]) are defined analogously – please refer to Table 1 – and so are the corresponding optima and approximation ratios. For instance, when dealing with dissimilarity information and costs, we have $\mathrm{Opt}_{\mathrm{Cost}_D} = \min_T \mathrm{Cost}_D(T)$ and the goal is to find $T$ so as to make the ratio $\mathrm{Cost}_D(T)/\mathrm{Opt}_{Cost_D}$ as small as possible.

Optimizing the above objectives exactly is know to be NP-hard (Dasgupta, 2016; Cohen-addad et al., 2019), hence the recent flurry of papers (e.g., (Dasgupta, 2016; Cohen-addad et al., 2019; Charikar et al., 2019a;b; Alon et al., 2020; Chatziafratis et al., 2020b; Naumov et al., 2020)) looking for fast approximation algorithms.

In this paper, we are specifically interested in HC algorithms having the *sequential property*, as defined next.

**Definition 2.1.** *Given a set of $n$ items $X = \{x_1, \ldots, x_n\}$, denote by $T_i = A(\langle x_1, \ldots, x_i \rangle)$ the (random) output of a HC algorithm $A$ having input $\{x_1, \ldots, x_i\}$, and let $T' = \mathrm{Ins}(T, x)$ denote a (randomized) insertion operation*

[1] This set may actually contain repeated items.

[2] In these notations, we leave the dependence on $[w_{i,j}]$ or $[d_{i,j}]$ implicit, no ambiguity will arise.

[3] This is a natural dissimilarity metric, although it does not seem to have been investigated previously.

| Objective | Type | Name | Symbol |
|---|---|---|---|
| $\max \sum_{i,j} w_{ij}(n - \lvert T_{ij}\rvert)$ | Sim. Rev. | MW Rev. | $\mathrm{Rev}_S$ |
| $\max \sum_{i,j} d_{ij}\lvert T_{ij}\rvert$ | Diss. Rev. | CKMM Rev. | $\mathrm{Rev}_D$ |
| $\min \sum_{i,j} w_{ij}\lvert T_{ij}\rvert$ | Sim. Cost | Dasgupta Cost | $\mathrm{Cost}_S$ |
| $\min \sum_{i,j} d_{ij}(n - \lvert T_{ij}\rvert)$ | Diss. Cost | MW Cost | $\mathrm{Cost}_D$ |

**Table 1:** HC objectives. In the above, we abbreviated "similarity" by "Sim.", "Dissimilarity" by "Diss., and "Revenue" by "Rev". The type of objective refers to whether it deals with similarity or dissimilarity information and that the goal is to maximize (revenue) or minimize (cost).

*that adds a new leaf $x$ to tree $T$, producing in output the augmented tree $T'$. We say that $A$ has the* sequential property *w.r.t.* Ins *if, for all item sets $X$, and all $i = 1, \ldots, n$, the random variable $T_i$ has the same distribution over trees as the random variable* $\mathrm{Ins}(T_{i-1}, x_i)$*, where $T_0$ is the empty tree.*

In other words, $A$ has the sequential property if it admits an exogenous insertion procedure Ins such that building the tree incrementally by inserting one leaf after the other through Ins yields the same statistical properties as if the tree were constructed by $A$ looking at all data in batch. Hence, if the tree constructed by $A$ has approximation ratio $r$ (in expectation over the internal randomization of $A$) then so does the tree incrementally constructed by Ins (in expectation over the internal randomization of Ins). Also, observe that this equivalence holds *independent* of the order in which Ins processes the $n$ items.

In the next section (Section 3), we develop a general framework for *hyperplane-based* hierarchical clustering which encompasses a family of dynamic algorithms for that task. Then, in the two subsequent sections, we shall describe and analyze two members of this family. The first one (Random Cut Tree Algorithm, Section 4) operates with axis-aligned hyperplanes, and is suited to the $\ell_1$-based objectives contained in Table 1, on a variety of definitions for $w_{i,j}$ and $d_{i,j}$. The second one (Uniform Radial Random Hyperplane Algorithm, Section 5) operates with general hyperplanes, and is suited to the analogous $\ell_2$-based objectives (Theorem 5.2). In addition, it enjoys an unconditional approximation guarantee (Theorem 5.3) for the weights defined by case 1 of Table 1. For both algorithms, we show that they can (i) be implemented efficiently, (ii) cater to $\ell_1$-based and $\ell_2$-based geometry respectively, and (iii) enjoy good approximation guarantees.

## 3. Hyperplane-based hierarchical clustering

Let $\mathrm{Graff}_{d-1}(\mathbb{R}^d)$ be the manifold of all $(d-1)$-dimensional affine subspaces (that is, hyperplanes) of $\mathbb{R}^d$. Let $\mu$ be a nonnegative measure on $\mathrm{Graff}_{d-1}(\mathbb{R}^d)$ that is finite on compact sets. With such a $\mu$, we associate a HC algorithm $A_\mu$, as described next.

| Objective | Metric (L1) | Approx. | Random |
|---|---|---|---|
| 1. MW Rev* | L1-similarity | 0.73 | 5/9 |
| 2. Dasgupta Cost* | L1-similarity | 2 | $\infty$ |
| 3. CKMM Rev | L1-distance | 0.90 | 2/3 |
| 4. MW Cost | L1-distance | 2 | $\infty$ |
| 5. MW Rev | Inverse distance | 0.80 | 1/3 |
| 6. Dasgupta Cost | Inverse distance | 1.5 | $\infty$ |
| 7. MW Rev | Gauss. Kernel | Figure 2 | $\frac{1+2\delta}{3}$ |
| 8. MW Rev | Abs. Exp. Kernel | Figure 2 | $\frac{1+2\delta}{3}$ |

**Table 2:** RCT approximation guarantees for different objectives and metrics. All metrics are $L1$-based. Only the first two cases (*) require Assumption 4.4. The last two cases assume that the weights are in $[\delta, 1]$, for some $\delta \in (0, 1]$. The last column is the approximation achieved by the baseline RANDOM that returns a binary tree on the leaves, which is chosen uniformly at random.

Given finite $X \subset \mathbb{R}^d$ as input, denote by $\mathrm{Conv}(X)$ the convex hull of $X$. Then the set $\mathcal{H}_X$ of hyperplanes of $\mathrm{Graff}_{d-1}(\mathbb{R}^d)$ that intersect $\mathrm{Conv}(X)$ is compact, and hence $\mu(\mathcal{H}_X) < \infty$. Let $\mu_X = \mu/\mu(\mathcal{H}_X)$ be the probability measure induced on $\mathcal{H}_X$ by restricting to $\mathcal{H}_X$ and normalizing to 1. On input $X$, algorithm $A_\mu$:

1. Samples a random hyperplane $H_X \sim \mu_X$;
2. Partitions $X$ into $Y$ and $Z$ according to $H_X$ (points lying on $H_X$ can be arbitrarily assigned to either $Y$ or $Z$).
3. Recurses on $Y$ and $Z$ using the probability measures $\mu_Y$ and $\mu_Z$, respectively.

Applying the above until we arrive at singleton sets, we construct a (random) binary tree $T$ with leaves the points in $X$, based on the partitions induced by the sampled hyperplanes. Such $T$ is the output of $A_\mu$ on input $X$. The key observation now is that for any set of points $Y'$ with $\mathrm{Conv}(Y) \subseteq \mathrm{Conv}(Y')$, we have $\mathcal{H}_Y \subseteq \mathcal{H}_{Y'}$, and thus $\mu_Y = \mu_{Y'}|_Y$, the probability measure of $\mu_{Y'}$ conditioned on $Y$. Thus we may rephrase Step 3 above as rejection-sampling from $\mu_{Y'}$ conditioned on the sampled hyperplane intersecting $\mathrm{Conv}(Y)$ (resp. $\mathrm{Conv}(Z)$).

If we have a way of sampling efficiently from the hyperplane probability measures, the main property of algorithm $A_\mu$ is that it leads to a natural algorithm for HC with the sequential property.[4]

**Theorem 3.1.** *Let $\mu$ be a nonnegative measure on* $\mathrm{Graff}_{d-1}(\mathbb{R}^d)$ *which is finite on compact sets, and suppose there is an efficient way to sample from $\mu_X$ for all finite sets $X$. Then, there is an efficient insertion operation* $\mathrm{Ins}_\mu$ *such that $A_\mu$ has the sequential property w.r.t.* $\mathrm{Ins}_\mu$.

In particular, recall that this means that the (random) tree generated by $\mathrm{Ins}_\mu$ is independent of the order in which Ins processes the inserted items. The general pseudocode for

[4] All proofs are given in the supplementary material.

$\text{Ins}_\mu$ is given in Appendix A. In the following sections, we specify particular measures $\mu$ from which hyperplanes can be efficiently sampled and which additionally give rise to HC algorithms having the sequential property, and exhibiting good approximation ratios for the metrics of Section 2. The associated insertion operations are presented in the corresponding sections of the appendix.

**Remark 3.2.** *It is important to stress that the above algorithm, as well as its by-products in later sections, do not take as input the pairwise information encoded by $[w_{i,j}]$ or $[d_{i,j}]$. These algorithms are purely geometric algorithms that will exhibit strong approximation properties, provided the pairwise information we use at evaluation time to compute the metrics in Table 1 is reasonably aligned with the geometry these algorithms rely upon. Further examples of this sort are the Projected Random Cut algorithm in (Charikar et al., 2019b), as well as the dynamic algorithms we compare to in our experimental investigation (Section 6).*

## 4. Random Cut Tree approximation

In this section we discuss a special case of hyperplane-based clustering known as the Random Cut Tree (RCT) which has been introduced by Guha et al. (2016) in the context of anomaly detection. We provide approximation results for related similarity and dissimilarity objectives (from Table 1). In the case of dissimilarity objectives, we use the distances themselves as the dissimilarity measure.

An RCT (batch algorithm) $T(X)$ on item set $X \subseteq \mathbb{R}^d$ is a tree-valued random variable generated as follows:

- Draw random index $I \in [d]$ with probability $\mathbb{P}[I = i] = \frac{l_i}{\sum_{i=1}^d l_i}$, where
$$l_i = \max_{x\in X}(x)_i - \min_{x\in X}(x)_i \ ,$$
with $(x)_i$ denoting the $i$-th component of vector $x$. Hence the above probability is proportional to the side lengths of the (axis-parallel minimum) bounding box of $X$;
- Draw threshold $\theta$ uniformly at random in the interval $[\min_{x\in X} x_I, \max_{x\in X} x_I]$;
- Let $X_1 = \{x \,|\, x \in X, (x)_I \le \theta\}$ and $X_2 = X \backslash X_1$ correspond to the left and right subtrees of the root of $T(X)$, and recurse on $X_1$ and $X_2$, until $T(X)$ is a (singleton) leaf.

We have the following characterization of RCT:

**Fact 4.1.** *Fix dimension d, and let $H_{i,v} = \{x \in \mathbb{R}^d \,|\, x_i = v\}$, where $x_i$ is the $i$-th component of vector $x$. Let then $\mathcal{H} = \{H_{i,v} \,|\, i \in [d], v \in \mathbb{R}\}$ be the set of axis-parallel hyperplanes. For $\mathcal{H}' \subset \mathcal{H}$, define $\mu_{\text{RCT}}$ by $\mu_{\text{RCT}}(\mathcal{H}') = \sum_{i=1}^d \mu_L(\{v \in \mathbb{R} \,|\, H_{i,v} \in \mathcal{H}'\})$, where $L$ is the standard Lebesgue measure on $\mathbb{R}$. Then $A_{\mu_{\text{RCT}}}$ (resp. $\text{Ins}_{\mu_{\text{RCT}}}$) is the offline (resp. dynamic) RCT algorithm.*

In (Guha et al., 2016), it is shown (Theorem 3 therein) that an RCT can be maintained over a set of points $X$ that is dynamically updated with streaming data in sub-linear update time and $O(dn)$ space. The pseudocode for the insertion operation (adapted from (Guha et al., 2016)) is given in Appendix B.

The analysis of RCT with respect to the HC objectives in Table 1 rests on an important restriction property that this algorithm enjoys.

**Definition 4.2.** *Given tree $T$ on the set of leaves $X$, and $R \subseteq X$, the* restriction *of $T$ to $R$ is the tree obtained by deleting the leaves of $T$ in $X \setminus R$ (along with their edges), and contracting edges to obtain a binary tree whose leaves are identified with $R$. In particular, if $R$ is a triplet $R = \{x_i, x_j, x_k\}$, the restriction of $T$ to $R$ when $\text{lca}_T(x_i, x_j)$ is a descendant of $\text{lca}_T(x_i, x_k)$ is the tree where $x_i, x_j$ are siblings, and $x_k$ is a sibling of their parent (and similarly for the other cases). See Figure 1 (right) for an illustration.*

**Lemma 4.3.** *Let $X \subseteq \mathbb{R}^d$ be a set of items. For any $R \subseteq X$, the restriction of the RCT $T(X)$ (that is, the output of RCT on input $X$) to subset $R$ has the same distribution as $T(R)$.*

In fact all algorithms from the family $A_\mu$ enjoy this property (see the supplementary material for a proof). We will use this result in the particular case of $R$ being a generic triplet $\{x_i, x_j, x_k\}$.

RCT as characterized in Fact 4.1 can be seen as naturally operating in an $\ell_1$ geometry. We now introduce a necessary assumption in order to obtain competitive approximation guarantees for RCT in the case of similarity-based objectives (MW Revenue and Dasgupta Cost) for the $\ell_1$ similarity measure $w_{i,j} = D - d_{i,j}$, where $d_{i,j} = ||x_i - x_j||_1$ and $D = \max_{i,j} d_{i,j}$. As we shall see momentarily, this assumption will not be required by dissimilarity-based objectives (CKMM Revenue and MW Cost).

**Assumption 4.4.** *We assume $\binom{n}{3}^{-1} \sum_{i<j<k}(d_{i,j} + d_{i,k} + d_{j,k})/2 \le D = \max_{i,j} d_{i,j}$. Observe that we have $\max_{i,j,k}(d_{i,j} + d_{i,k} + d_{j,k})/2 \le \frac{3D}{2}$ always, so this also follows under the modified similarity $w_{i,j} = \frac{3}{2}D - d_{i,j}$. The weights are now in the range $[\frac{D}{2}, \frac{3D}{2}]$, and in this case,* RANDOM *gives a baseline revenue approximation of $\frac{3D/2+D/2+D/2}{3(3D/2)} = \frac{5}{9}$.*

The reason for Assumption 4.4 is the following. RCT is a geometric algorithm whose cuts of triplets $\{x_i, x_j, x_k\}$ depend on the distances $d_{i,j}$, $d_{i,k}$, and $d_{j,k}$. Allowing the similarity weights $w_{i,j}$ to have ratios substantially different from the corresponding ratios of the $d_{i,j}$'s can lead to adversarial situations, as we illustrate next.

**Example 4.5.** *Let $V \subset \mathbb{R}^3$ consist of the points $x_1 = (1+\epsilon, 0, 0)$, $x_2 = (0,1,0)$, and $x_3 = (0,0,1)$. We have $D = d_{1,2} = d_{1,3} = 2+\epsilon$, $d_{2,3} = 2$, $w_{1,2} = w_{1,3} = 0$, and $w_{2,3} = \epsilon$. Thus $\mathrm{Rev}_{\mathrm{Opt}_S} = \epsilon$. On the other hand, being based on the $\ell_1$ geometry,* RCT *makes the cuts with approximately equal probabilities, which leads to an approximation ratio of $1/3 + O(\epsilon)$, that is, very close to the trivial approximation ratio of 1/3 achieved on $\mathrm{Rev}_S$ by a* random *binary tree (Moseley & Wang, 2017).*

**Theorem 4.6.** RCT *satisfies the approximation guarantees for the combination of objectives and metrics listed in Table 2. In detail, for each combination of revenue (resp. cost) objective $\mathrm{Obj}$, metric $m$, and approximation factor $\alpha$ in Table 2, we have the approximation guarantee that for all $X \subseteq \mathbb{R}^d$ endowed with metric $m$, $\mathbb{E}[\mathrm{Obj}(\mathrm{RCT}(X))] \geq \alpha \mathrm{Opt}_{\mathrm{Obj}}(X)$ (resp. $\leq$), where the expectation is over the internal randomization of* RCT*, and $X$ satisfies Assumption 4.4 in the first two cases.*

While Theorem 4.6 covers a diverse range of objectives and metrics, the proof technique is similar. We sketch the main idea in the case of MW Revenue with $\ell_1$-similarity.

The following length-proportional cut property of the RCT algorithm is a main ingredient of our approximation results.

**Lemma 4.7.** *Given input $X$ and a cut $H_X$ sampled from $\mu_X$, the probability $p_{i,j}$ that $x_i$ and $x_j$ are split by $H$ is proportional to their $\ell_1$ distance $d_{i,j}$.*

A consequence of Lemma 4.7 which we need for the proof of Theorem 4.6 is the following lemma.

**Lemma 4.8.** *Fix a triplet $\{x_i, x_j, x_k\}$ of $X$. Then the probability, $p_{i,j|k}$, that* RCT *$T(X)$ separates $x_i$ and $x_j$ from $x_k$ is given by*

$$p_{i,j|k} = \frac{d_{i,k} + d_{j,k} - d_{i,j}}{d_{i,j} + d_{i,k} + d_{j,k}} ,$$

*and similarly for $p_{i,k|j}$ and $p_{j,k|i}$.*

We use below the cyclic sum notation $\sum_{\mathrm{cyc}} f(i,j,k) = f(i,j,k) + f(j,k,i) + f(k,i,j)$. For a tree $T$, and a triplet of leaves $i,j,k$, we write $ij|k$ to mean that $\mathrm{lca}_T(i,j)$ is a descendant of $\mathrm{lca}_T(i,k)$.

*Proof of Theorem 4.6– sketch.* Fix input $X = \{x_1, \dots, x_n\}$. Given a tree $T$ on $X$, note that we can rewrite the MW Revenue $\sum_{i,j} w_{i,j}(n - |T_{i,j}|)$ as the triplet-wise sum $\sum_{i<j<k} \mathrm{Rev}_{i,j,k}(T)$, where

$$\mathrm{Rev}_{i,j,k}(T) = \begin{cases} w_{i,j} & \text{if } ij|k \text{ in } T \\ w_{i,k} & \text{if } ik|j \text{ in } T \\ w_{j,k} & \text{if } jk|i \text{ in } T \,. \end{cases} \quad (1)$$

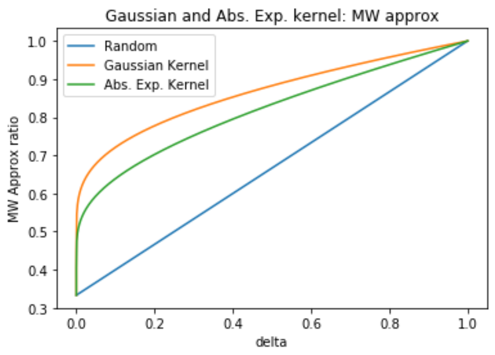

**Figure 2:** Approximation ratio for absolute exponential kernel and Gaussian kernel with weights $w_{i,j} \in [\delta, 1]$. RCT (resp. URRH) satisfy these guarantees with the $\ell_1$ (resp. $\ell_2$) distances in these kernels.

Now, from Lemma 4.8, for tree $T_{\mathrm{RCT}}(X)$ computed by RCT on $X$ we can write

$$\mathbb{E}[\mathrm{Rev}_{i,j,k}(T(X))] = \sum_{\mathrm{cyc}} \frac{d_{i,k} + d_{j,k} - d_{i,j}}{d_{i,j} + d_{i,k} + d_{j,k}} (D - d_{i,j}) \,.$$

On the other hand, we also have the upper bound $\mathrm{Rev}_{i,j,k}(T_{\mathrm{Opt}}(X)) \leq \max\{D - d_{i,j}, D - d_{i,k}, D - d_{j,k}\}$, where $T_{\mathrm{Opt}}(X)$ is the optimal tree on $X$. It now remains to use these expressions to prove the required approximation bound, which reduces to a tractable optimization problem. □

We have shown that RCT enjoys good approximation guarantees for $\ell_1$-based measures. In the following section, we introduce a new algorithm URRH that matches RCT's approximation guarantees for $\ell_2$-based similarity measures. To conclude this section, we introduce a simple extension of RCT that achieves weaker guarantees than URRH, but is of theoretical interest. Namely, we define the Projected Random Cut Tree (PRCT) algorithm as follows:

**Definition 4.9.** *Given input $X$ and projection dimension $k$,* PRCT$(k)$ *applies* RCT *to $(Px_i)_{i=1}^n$, where $P$ is a $k \times d$ Gaussian projection matrix with i.i.d. entries.*

We remark that when $k = 1$, PRCT reduces to the projected random cut algorithm from (Charikar et al., 2019b).

We have the following guarantees for PRCT:

**Theorem 4.10.** *Fix $\epsilon, \delta > 0$. Consider a revenue case from Table 2 (cases 1, 3, 5, 7, 8) but with the corresponding $\ell_2$ metric. Let $\alpha$ be the corresponding approximation guarantee of* RCT *under $\ell_1$ metric. Then there exists an absolute constant $c$ such that with probability $1-\delta$,* PRCT *with $k = c\log(n/\delta)/\epsilon^2$ achieves an expected approximation of $\alpha - \epsilon$ in the $\ell_2$ metric.*

**Theorem 4.11.** *Fix $\epsilon, \delta > 0$. Consider a cost case from Table 2 (cases 2, 4, 6) but with the corresponding $\ell_2$ metric. Let $\alpha$ be the corresponding approximation guarantee of* RCT *under $\ell_1$ metric and assume that the weights lie in the range $[\gamma, 1]$ for arbitrarily small but positive $\gamma$. Then there exists an absolute constant $c$ such that with probability $1-\delta$,* PRCT *with $k = c\log(n/\delta)/\epsilon^2$ achieves an expected approximation of $\alpha + \epsilon$ in the $\ell_2$ metric.*

## 5. Uniform radial random hyperplane approximation

The second hyperplane-based HC algorithm we present is the URRH (Uniform Radial Random Hyperplane) Algorithm.

At each recursive step, URRH takes as input a subset $C \subseteq X$ of the input items $X$, and *any* $(d-1)$-sphere $\mathcal{S}(C)$ containing all points of $C$. The algorithm randomly cuts $\mathcal{S}(C)$ to split $C$ into $C'$ and $C''$. Whenever the cut makes either $C'$ or $C''$ empty, the hyperplane is rejected, and a new cut is drawn until $C', C'' \neq \emptyset$. Finally, as in RCT, the URRH algorithm recurses on $C'$ and $C''$ until the input becomes a singleton.

The details (pseudocode) of URRH are given in Appendix C. Below we give an idea of the key steps.

Each random hyperplane is selected through a two-step process: (i) A direction in $\mathbb{R}^d$ is selected by choosing a unit vector $p$ uniformly at random, and (ii) a hyperplane orthogonal to $p$ is selected among those intersecting $\mathcal{S}(C)$. More precisely, at each recursive step, URRH operates as follows:

- Direction $p$ is selected uniformly at random from $\mathcal{S}^{d-1}$, the unit $(d-1)$-sphere;
- Let $\mathcal{S}(C)$ be *any* $(d-1)$-sphere containing all items in $C$ (e.g., $\mathcal{S}(C)$ is the *circumsphere* of $\mathrm{Conv}(C)$).[5] Let $r$ and $c$ be the radius and the center of $\mathcal{S}(C)$. Hyperplane $H_{p,b}(\mathcal{S}(C)) := \{x \in \mathbb{R}^d \,:\, x \cdot p = b\}$ is generated by drawing $b$ uniformly at random from the interval $[c \cdot p - r,\, c \cdot p + r]$.
- If $H_{p,b}(\mathcal{S}(C))$ cuts $C$, i.e., it splits $C$ into $C'$ and $C''$ such that $C', C'' \neq \emptyset$, then we recurse on $C'$ and $C''$, otherwise we *reject* $H_{p,b}(\mathcal{S}(C))$ and generate it again (by re-drawing $p$ and $b$)).

From the above, a clear computational trade-off emerges between calculating a $(d-1)$-sphere $\mathcal{S}(C)$ having small radius, and the number of hyperplanes that get rejected. As mentioned in Appendix C, there are several strategies to resolve this trade-off. For now, we just anticipate that, for any $d \in \mathbb{N}$ and any $C$, the probability that $H_{p,b}(\mathcal{S}(C))$ is *not* rejected is at least $\frac{c}{\sqrt{d}}$ for a constant $c$, and decreases linearly in the radius of the sphere $\mathcal{S}$ currently used by URRH (see the formal statement in Lemma C.5 in Appendix C).

[5] That is, the smallest sphere enclosing all the points of $\mathrm{Conv}(X)$.

We have the following characterization of URRH as a member of the general hyperplane-based family.

**Fact 5.1.** *Fix dimension $d$, let $H_{u,v} = \{x \in \mathbb{R}^d \,|\, x \cdot u = v\}$, and $\mathcal{H} = \{H_{u,v} \,|\, u \in \mathcal{S}^{d-1}, v \in \mathbb{R}\}$ be the set of all hyperplanes in $\mathbb{R}^d$. Define $\mu_{\text{URRH}}(\mathcal{H}') = \int_{u \in \mathcal{S}^{d-1}} \mu_L(\{v \in \mathbb{R} \,|\, H_{u,v} \in \mathcal{H}'\} d\nu$ for $\mathcal{H}' \subset \mathcal{H}$, where $\mu_L$ is the Lebesgue measure on $\mathbb{R}$ and $\nu$ is the uniform measure on $\mathcal{S}^{d-1}$. Then $A_{\mu_{\text{URRH}}}$ (resp. $\mathrm{Ins}_{\mu_{\text{URRH}}}$) is the offline (resp. dynamic)* URRH *algorithm.*

In fact, URRH satisfies Lemma 4.7 with $d_{i,j}$ now representing $\ell_2$ distances. The same proof machinery for Theorem 4.6 thus applies to URRH for the case that all measures are $\ell_2$-based, and we have the following result.

**Theorem 5.2.** URRH *satisfies the same approximation guarantees as* RCT *given in Theorem 4.6, with $\ell_2$-based measures under the equivalent $\ell_2$ analog of Assumption 4.4 for the first two cases.*

We also have the following unconditional approximation ratio guarantees, for the case of $\mathrm{Rev}_S$ and similarity weights defined as $w_{i,j} := D - d_{i,j}$, where $d_{i,j}$ is the Euclidean distance between $x_i$ and $x_j$, and $D = \max_{1 \le i < j \le n} d_{i,j}$ is the maximal distance over all pairs of points in $X$. The result states that the MW Revenue of URRH is strictly larger than the trivial $\frac{1}{3}$ approximation ratio[6] for any input dimension $d > 3$, whenever $n$ is not too small w.r.t. to $d$.

**Theorem 5.3.** *Given any input set $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$, with $d > 3$, the approximation ratio $\frac{\mathbb{E}[\mathrm{Rev}_S(\text{URRH}(X))]}{\mathrm{Opt}_{\mathrm{Rev}_S}}$ is lower bounded by $\frac{1}{3} + g(d,n)$, where $g(d,n)$ is a function of $d$ and $n$ such that $g(d,n) > 0$ for all $n > \frac{605}{116} d \approx 5.22d$. In particular, if $n \geq \left(9 + \frac{38}{d-3.98}\right) d$ and $d > 3$, we have*

$$\mathbb{E}[\mathrm{Rev}_S(\text{URRH}(X))] \geq \left(\frac{1}{3} + \frac{1}{31d^3}\right) \mathrm{Opt}_{\mathrm{Rev}_S} .$$

*In the above, the expectation is over the internal randomization of* URRH.

Notice that condition $d > 3$ does not really limit the scope of Theorem 5.3, since one can always pad the input vectors with dummy components that do not alter pairwise distances, so as to force $d \geq 4$. Moreover, it is also worth observing that for all $d < 8$, the requirement $n \geq \left(9 + \frac{38}{d-3.98}\right) d$

[6] Recall that approximation ratio 1/3 can be trivially achieved in expectation by a randomly generated tree (Moseley & Wang, 2017).

becomes less stringent if one pads the input so as to force $d = 8$. This implies that $148 = \left\lceil \left(9 + \frac{38}{8-3.98}\right) 8 \right\rceil$ input points are always sufficient when $d \leq 8$. Finally, the special case $d = 1$ can be treated separately (see Theorem D.1 in the appendix) obtaining an expected MW revenue larger than the one of Theorem 5.3.

As mentioned in Fact 5.1, URRH has the sequential property (Definition 2.1). The pseudocode of the insertion procedure is detailed in Algorithm 5 in Appendix C.

## 6. Experiments

In this section, we demonstrate experimentally that RCT and URRH perform competitively compared to other well-known dynamic HC algorithms. In particular, we compare to BIRCH (Zhang et al., 1996), PERCH (Kobren et al., 2017), and GRINCH (Monath et al., 2019). Additionally, we compare all algorithms to the RANDOM baseline that builds a tree at random, and to PROJECTED RANDOM CUT (Charikar et al., 2019b) on a line, which can be made dynamic by applying RCT to the projected dataset. The objectives considered are MW Revenue, MW Cost, CKMM Revenue and Dasgupta Cost.

We evaluate these algorithms on both synthetic and real-world datasets. For synthetic datasets, we evaluate the performance of these algorithms in both noisy and well-separated settings. In particular, we draw 10K examples from standard Gaussians in $\mathbb{R}^2$. In the noisy setting, we draw from one Gaussian, and in the well-separated setting, we draw from two Gaussians with centers separated by four standard deviations in one direction. We denote these datasets by `OneG`, resp. `TwoG`. For real-world datasets, we compare the algorithms on the following data of varying scale: `MNIST`, `ALOI` (Geusebroek et al. (2005)), and `ILSVRC12` (Deng et al. (2009)) trained with ResNet34 architecture. We note that when considering our (four) objectives, the resulting trees must be binary. All algorithms other than BIRCH output binary trees and thus do not need to be modified. In order to handle BIRCH, we follow the methodology of (Naumov et al., 2020) and simply assign the value of a random partitioning to all data point triplets that share the same lca in the tree. For an extended explanation see (Naumov et al., 2020), Appendix B.1 therein. For hardware, we used machines with a maximum of 125GB of RAM and 16 CPUs.

**Methodology.** For each of these experiments, we randomly permute the datasets and stream each one in the preprocessed order consistently across algorithms. We evaluate each of the aforementioned measures on the produced hierarchies, as follows: We sample 10K triplets $\mathcal{T}'$ from each dataset, then compute the measures restricted to these triplets. For the MW Revenue, this is $\sum_{(i,j,k)\in\mathcal{T}'} \mathrm{Rev}_{i,j,k}(T)$ (see Equation 1). We also report the measures for RANDOM: $\sum_{(i,j,k)\in\mathcal{T}'}(w_{i,j} + w_{j,k} + w_{i,k})/3$ and an upper (resp. lower) bound for the optimal revenue (resp. cost): $\sum_{(i,j,k)\in\mathcal{T}'} \max$ (resp. $\min)(w_{i,j}, w_{j,k}, w_{i,k})$ (see (Naumov et al., 2020)).[7] Finally, for RCT, URRH and PROJECTED RANDOM CUT, the output trees are non-deterministic, so we report the average over 10 different runs.

Table 3 compares MW Revenue using RBF kernel similarity $\mathrm{RBF}(x, y) = e^{-||x-y||_2^2/2\sigma^2}$ across all algorithms. Note that this is a function of the $\ell_2$ distance, which we have chosen for uniform comparison across all algorithms. We choose $\sigma$ as the mean $\ell_2$ distance between pairs of points. This is to ensure a reasonable distribution of similarity weights. We defer the results for MW Cost, CKMM Revenue and Dasgupta Cost to the appendix, but note that they show similar trends. The following conclusions can be drawn:

(1) RCT and URRH achieve the highest revenue for `OneG`, a noisy setting in which there is no obvious way to split the data into two clusters at the root level. By contrast, they are outperformed by BIRCH, PERCH and GRINCH on `TwoG`, where the two clusters are well separated. We believe this can be explained by the fact that the baseline approaches rely heavily on clusters and/or nearest neighbor information to build the trees. On the other hand, RCT and URRH split the data by random cuts that are less sensitive to local data densities. Thus, the baselines take advantage of well-separated datasets, while RCT and URRH are more robust on noisy data.

(2) BIRCH, PERCH and GRINCH perform reasonably well for these objectives even though they are not explicitly designed to do so. We offer two reasons for this. First, in many of our experiments, RANDOM turned out to perform reasonably (and sometimes surprisingly) well; a similar phenomenon has been experimentally observed in (Naumov et al., 2020). When this happens, there is not much room for improvement across the various algorithms. Second, in order to ensure a fair comparison, in our experiments each dataset was randomly shuffled. This does not affect RCT and URRH, but it might have potentially eliminated unfavorable orderings of data for BIRCH, PERCH and GRINCH, thereby giving these competitors some advantage.

(3) RCT and URRH perform competitively compared to all other algorithms on the real-world datasets we tested, where clusters are moderately well-separated. Unlike BIRCH, PERCH and GRINCH, the practical relevance of RCT and URRH is complemented by their approximation

[7] These metrics are computed by sampling triplets, since exact computation would be unwieldy. The extra variance generated in the results turns out to be negligible.

| | MNIST | ILSVRC12 | ALOI | OneG | TwoG |
|---|---|---|---|---|---|
| RCT | 0.93±0.01 | 0.94±0.0 | 0.91±0.01 | 0.9±0.01 | 0.9±0.06 |
| URRH | 0.93±0.0 | 0.94±0.0 | 0.9±0.01 | 0.9±0.01 | 0.9±0.03 |
| BIRCH | 0.93 | 0.94 | 0.91 | 0.87 | 0.98 |
| PERCH | 0.92 | 0.94 | 0.91 | 0.87 | 0.90 |
| GRINCH | 0.93 | 0.93 | 0.89 | 0.88 | 0.97 |
| PROJECTED RANDOM CUT | 0.92±0.0 | 0.94±0.0 | 0.88±0.01 | 0.87±0.0 | 0.86±0.07 |
| RANDOM | 0.92 | 0.93 | 0.85 | 0.74 | 0.71 |
| UPPER BOUND | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |

**Table 3:** MW Revenue approximation factors using RBF kernel similarity; ↑ is better. Each revenue is shown as a percentage of the corresponding upper bound for that dataset.

guarantees.

## 7. Conclusions and ongoing activity

We have introduced the general framework of hyperplane-based HC for data streams in metric spaces. We have given a general family of algorithms indexed by a sampling probability over hyperplanes. Each algorithm in this family admits two formulations, batch and sequential, whose (statistical) equivalence ensures a desirable robustness to data arrival order. We have studied two fast HC algorithms originating from this general family, and provided a number of approximation guarantees w.r.t. known objective functions, some of which improve on the available literature on HC. In addition, the algorithms are simple to implement, requiring only the selection of a splitting hyperplane in each node. New points are inserted as siblings of existing nodes, without the need to perform other changes in the tree structure.

We have run initial experiments on synthetic and real-world metric data, where the trend that seems to emerge is that our randomized algorithms are on par with celebrated dynamic HC baselines in the presence of moderate noise levels, tend to outperform these baselines with higher noise rate and be outperformed in the opposite case of clear cluster separation.

An interesting research direction is to generalize the family of hyperplane-based HC to richer separation classes, which would give us higher flexibility, while still retaining the crucial benefits of the dynamic solutions.

We conclude by mentioning a couple of additional results we obtained for the MW Revenue maximization problem on one-dimensional data, with weights $w_{i,j} := D - d_{i,j}$, and $n \to \infty$. We proved (see Appendix D) that the approximation ratio of RCT is at least $0.8303$. We also developed two *very* fast deterministic algorithms for the batch setting, achieving approximation ratios of $\frac{3}{4}$ and $\frac{1}{2}$, respectively. Interestingly enough, the latter is always obtained by simply building a caterpillar tree, and has also a $\frac{3}{4}$-approximation ratio for the CKMM Revenue.

## References

Ailon, N. and Chazelle, B. The fast johnson-lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1), 2009.

Alon, N., Azar, Y., and Vainstein, D. Hierarchical clustering: A 0.585 revenue approximation. In *Proceedings of Thirty Third Conference on Learning Theory*, volume 125, pp. 153–162. PKLR, 2020.

Chami, I., Gu, A., Chatziafratis, V., and Ré, C. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020.

Charikar, M. and Chatziafratis, V. Approximate hierarchical clustering via sparsest cut and spreading metrics. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, pp. 841–854, 2017.

Charikar, M., Chatziafratis, V., and Niazadeh, R. Hierarchical clustering better than average-linkage. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2291–2304. Society for Industrial and Applied Mathematics, 2019a.

Charikar, M., Chatziafratis, V., Niazadeh, R., and Yaroslavtsev, G. Hierarchical clustering for euclidean data. In *Proceedings of Machine Learning Research*, volume 89, pp. 2721–2730. PMLR, 2019b.

Chatziafratis, V., Niazadeh, R., and Charikar, M. Hierarchical clustering with structural constraints. In *Proceedings of the 35th International Conference on Machine Learning, (ICML 2018)*, pp. 773–782, 2018.

Chatziafratis, V., Gupta, N., and Lee, E. Inapproximability for local correlation clustering and dissimilarity hierarchical clustering. *CoRR*, abs/2010.01459, 2020a.

Chatziafratis, V., Yaroslavtsev, G., Lee, E., Makarychev, K., Ahmadian, S., Epasto, A., and Mahdian, M. Bisect and conquer: Hierarchical clustering via max-uncut bisection. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108, pp. 3121–3132. PMLR, 2020b.

Chen, K. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM J. Comput.*, 39(3):923–947, 2009. doi: 10.1137/070699007. URL https://doi.org/10.1137/070699007.

Cohen-addad, V., Kanade, V., Mallmann-trenn, F., and Mathieu, C. Hierarchical clustering: Objective functions and algorithms. *J. ACM*, 66:4, 2019.

Dasgupta, S. A cost function for similarity-based hierarchical clustering. In *Proceedings of the 48th annual ACM symposium on Theory of Computing*, pp. 118127, 2016.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Eisen, M. B., Spellman, P. T., Brown, P. O., and Botstein, D. Cluster analysis and display of genomewide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

Geusebroek, J.-M., Burghouts, G. J., and Smeulders, A. W. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

Gilbert, F., Simonetto, P., Zaidi, F., Jourdan, F., and Bourqui, R. Communities and hierarchical structures in dynamic social networks: Analysis and visualization. *Social Network Analysis and Mining*, 1(2), 2011.

Guha, S., Mishra, N., Roy, G., and Schrijvers, O. Robust random cut forest based anomaly detection on streams. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pp. 2712–2721. PMLR, 2016.

Kobren, A., Monath, N., Krishnamurthy, A., and McCallum, A. A hierarchical algorithm for extreme clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 255–264. Association for Computing Machinery, 2017.

Li, S. Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics and Statistics*, 4(1):66–70, 2011.

Lin, G., Nagarajan, C., Rajaraman, R., and Williamson, D. P. A general approach for incremental approximation and hierarchical clustering. *SIAM J. Comput.*, 39(8):3633–3669, 2010. doi: 10.1137/070698257. URL https://doi.org/10.1137/070698257.

Loewenstein, Y., Portugaly, E., Fromer, M., and Linial, M. Efficient algorithms for accurate hierarchical clustering of huge datasets: tackling the entire protein space. In *Proceedings 16th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pp. 41–49, 2008.

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

Monath, N., Kobren, A., Krishnamurthy, A., Glass, M. R., and McCallum, A. Scalable hierarchical clustering with tree grafting. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1438–1448. Association for Computing Machinery, 2019.

Moseley, B. and Wang, J. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. In *Advances in Neural Information Processing Systems*, volume 30, pp. 30943103. Curran Associates, Inc., 2017.

Naumov, S., Yaroslavtsev, G., and Avdiukhin, D. Objective-based hierarchical clustering of deep embedding vectors. In *Proc. AAAI*, 2020.

Nguyen, T., Schmidt, B., and Kwoh, C. K. Sparsehc: A memory-efficient online hierarchical clustering algorithm. In Abramson, D., Lees, M., Krzhizhanovskaya, V. V., Dongarra, J. J., and Sloot, P. M. A. (eds.), *Proceedings of the International Conference on Computational Science, ICCS 2014*, volume 29 of *Procedia Computer Science*, pp. 8–19. Elsevier, 2014.

Rodrigues, P. P., Gama, J., and Pedroso, J. P. ODAC: hierarchical clustering of time series data streams. In Ghosh, J., Lambert, D., Skillicorn, D. B., and Srivastava, J. (eds.), *Proceedings of the Sixth SIAM International Conference on Data Mining*, pp. 499–503. SIAM, 2006.

Schmidt, M. and Sohler, C. Fully dynamic hierarchical diameter k-clustering and k-center. *CoRR*, abs/1908.02645, 2019. URL http://arxiv.org/abs/1908.02645.

Vainstein, D., Chatziafratis, V., Citovsky, G., Rajagopalan, A., Mahdian, M., and Azar, Y. Hierarchical clustering via sketches and hierarchical correlation clustering. *CoRR*, abs/2101.10639, 2021.

Wang, D. and Wang, Y. An improved cost function for hierarchical cluster trees. In *ArXiv, abs/1812.02715*, 2018.

Wang, Y. and Moseley, B. An objective for hierarchical clustering in euclidean space and its connection to bisecting k-means. In *Proc. AAAI*, pp. 6307–6314, 2020.

Zhang, T., Ramakrishnan, R., and Livny, M. Birch: An efficient data clustering method for very large databases. In *SIGMOD*, volume 25(2), pp. 103–114, 1996.