

Fitting Tree Metrics and Ultrametrics in Data Streams

Amir Carmel 

Pennsylvania State University, State College, PA, USA¹

Debarati Das 

Pennsylvania State University, State College, PA, USA

Evangelos Kipouridis 

Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Evangelos Pipis 

National Technical University of Athens, Greece

Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Abstract

Fitting distances to tree metrics and ultrametrics are two widely used methods in hierarchical clustering, primarily explored within the context of numerical taxonomy. Formally, given a positive distance function $D : \binom{V}{2} \rightarrow \mathbb{R}_{>0}$, the goal is to find a tree (or an ultrametric) T including all elements of set V , such that the difference between the distances among vertices in T and those specified by D is minimized. Numerical taxonomy was first introduced by Sneath and Sokal [Nature 1962], and since then it has been studied extensively in both biology and computer science.

In this paper, we initiate the study of ultrametric and tree metric fitting problems in the semi-streaming model, where the distances between pairs of elements from V (with $|V| = n$), defined by the function D , can arrive in an arbitrary order. We study these problems under various distance norms; namely the ℓ_0 objective, which aims to minimize the number of modified entries in D to fit a tree-metric or an ultrametric; the ℓ_1 objective, which seeks to minimize the total sum of distance errors across all pairs of points in V ; and the ℓ_∞ objective, which focuses on minimizing the maximum error incurred by any entries in D .

- Our first result addresses the ℓ_0 objective. We provide a single-pass polynomial-time $\tilde{O}(n)$ -space $O(1)$ approximation algorithm for ultrametrics and prove that no single-pass exact algorithm exists, even with exponential time.
- Next, we show that the algorithm for ℓ_0 implies an $O(\Delta/\delta)$ approximation for the ℓ_1 objective, where Δ is the maximum, and δ is the minimum absolute difference between distances in the input. This bound matches the best-known approximation for the RAM model using a combinatorial algorithm when $\Delta/\delta = O(n)$.
- For the ℓ_∞ objective, we provide a complete characterization of the ultrametric fitting problem. First, we present a single-pass polynomial-time $\tilde{O}(n)$ -space 2-approximation algorithm and show that no better than 2-approximation is possible, even with exponential time. Furthermore, we show that with an additional pass, it is possible to achieve a polynomial-time exact algorithm for ultrametrics.
- Finally, we extend all these results to tree metrics by using only one additional pass through the stream and without asymptotically increasing the approximation factor.

2012 ACM Subject Classification Theory of computation $\rightarrow$ Streaming models; Theory of computation $\rightarrow$ Sketching and sampling; Theory of computation $\rightarrow$ Facility location and clustering

Keywords and phrases Streaming, Clustering, Ultrametrics, Tree metrics, Distance fitting

Digital Object Identifier 10.4230/LIPIcs.ICALP.2025.42

Category Track A: Algorithms, Complexity and Games

* Part of this work was done while the author was affiliated at Weizmann Institute of Science, Israel



Related Version *Full Version*: <https://arxiv.org/abs/2504.17776> [17]

Funding *Debarati Das*: Research supported by NSF grant 2337832.

1 Introduction

Hierarchical clustering is a method of cluster analysis that builds a hierarchy of clusters by starting with each data point as its own cluster and successively merging the two closest clusters until all points are merged into a single cluster or a stopping criterion is met. This method involves creating a *dendrogram*, a tree-like diagram, that records the sequence of merges or splits, allowing for easy visualization and interpretation of the hierarchical structure. Hierarchical clustering uses various distance metrics (e.g., Euclidean, Manhattan, cosine) and linkage criteria (e.g., single, complete, average, Ward’s method), providing flexibility to tailor the analysis to specific data characteristics and clustering goals. It is versatile across different types of data, including numerical, categorical, and binary data, and has become the preferred method for analyzing gene expression data [35] and constructing phylogenetic trees [3, 43]. Consequently, hierarchical clustering plays a significant role in both theory and practice across various domains, such as image processing to group similar regions within images [48], social network analysis to identify communities within a network [14], and business and marketing to segment customers based on behavior, preferences, or purchasing patterns [46].

Tree metrics and ultrametrics are fundamental measures used in hierarchical clustering, where the distance between any two points is defined by the cost of the unique path connecting them in a tree-like structure. Formally, given a distance function $D : \binom{V}{2} \rightarrow \mathbb{R}_{>0}$, the goal is to find a tree T with positive edge weights, encompassing all elements of set V as vertices. This tree T should best match the distances specified by D . In the case of ultrametrics, the tree must be rooted, and all elements of V must appear as leaf nodes at the same depth.

The task of fitting distances with tree metrics and ultrametrics, often referred to as the numerical taxonomy problem, has been a subject of interest since the 1960s [18, 53, 54]. One of the pioneering works in this area was presented by Cavalli-Sforza and Edwards in 1967 [18]. Different formulations of the optimal fit for a given distance function D lead to various objectives, such as minimizing the number of disagreements (using the ℓ_0 norm of the error vector), minimizing the sum of differences (using the ℓ_1 norm), and minimizing the maximum error (using the ℓ_∞ norm).

Deploying hierarchical clustering (HC) algorithms faces significant challenges due to scalability issues, particularly with the rise of data-intensive applications and evolving datasets. As data volumes continue to grow, there is an urgent need for efficient algorithms tailored for large-scale models such as streaming, local algorithms, MPC, and dynamic models, given the large input sizes relative to available resources. In this work we study hierarchical clustering, focusing on tree metrics and ultrametrics in the semi-streaming model. The model supports incremental updates, keeping the information about the clusters current without the need to reprocess the entire dataset. This adaptability makes hierarchical clustering highly valuable for applications such as network monitoring and social media analysis, where real-time insights are essential [47, 49, 52].

A recent result [5] studied hierarchical clustering (HC) in the graph streaming model, providing a polynomial-time, single-pass $\tilde{O}(n)$ space algorithm that achieves an $O(\sqrt{\log n})$ approximation for HC. When space is more limited, specifically $n^{1-o(1)}$, the authors show that no algorithm can estimate the value of the optimal hierarchical tree within an $o(\log n \log \log n)$ factor, even with *poly* $\log n$ passes over the input and exponential time.

A special case of the ultrametric fitting problem is where the tree depth is two, known as the *Correlation Clustering problem*. In this problem given a complete graph $G = (V, E)$ with edges labeled either similar (0) or dissimilar (1), the objective is to partition the vertices V into clusters to minimize the disagreements. After a decade of extensive research on correlation clustering in the semi-streaming setting [2, 6, 9, 10, 23, 28, 29], a recent breakthrough in [32] introduces a single-pass algorithm that achieves a 1.847 approximation using $\tilde{O}(n)$ space. This directly improves upon two independent works [15, 51], both presenting single-pass algorithms achieving a $(3 + \varepsilon)$ -approximation using $O(n/\varepsilon)$ space.

However, our understanding of streaming algorithms for larger depths, particularly within the context of ultrametrics and tree metrics, is very limited. The challenge arises from the fact that, unlike correlation clustering, which deals with only two distinct input distances, this problem may involve up to n^2 different distances, especially in a highly noisy input. Although the optimal output tree can be defined using at most n of these n^2 distances, identifying these n distances is non-trivial. As a result, in the worst case, it may be necessary to store all observed input distances, which would require quadratic space if done naively. Additionally, the hierarchical nature of clusters at various tree depths introduces inherent dependencies among clusters at different levels. This complexity makes it highly challenging to adapt the ideas used in streaming algorithms for correlation clustering to ultrametrics and tree metrics. In this paper, we offer the first theoretical guarantees by providing several algorithmic results for fitting distances using ultrametrics and tree metrics in the semi-streaming setting under various distance norms.

1.1 Other Related Works

1.1.1 Ultrametrics and Tree metrics

The numerical taxonomy problem, which involves fitting distances with tree metrics and ultrametrics, was first introduced by Cavalli-Sforza and Edwards in 1967 [18]. Day demonstrated that this problem is NP-hard for both ℓ_1 and ℓ_2 norms in the context of tree metrics and ultrametrics [34]. Moreover, these problems are APX-hard [20], as inferred from the APX-hardness of Correlation Clustering, which rules out the possibility of a polynomial-time approximation scheme. On the algorithmic side, Harp, Kannan, and McGregor [42] developed an $O(\min\{n, k \log n\}^{1/p})$ approximation for the ℓ_p objective in the ultrametric fitting problem, where k is the number of distinct distances in the input. Ailon and Charikar [3] improved this to an $O((\log n)(\log \log n)^{1/p})$ approximation, which they extended to the tree metric using a reduction from Agarwala [1]. A recent breakthrough in [24] presented the first constant-factor approximation for the ℓ_1 objective for both ultrametric and tree metric.

The ℓ_0 objective was first investigated in [26], which developed a novel constant-factor approximation algorithm. Charikar and Gao [19] improved the approximation guarantee to 5. For the weighted ultrametric violation distance, where the weights satisfy the triangle inequality, they provided an $O(\min\{L, \log n\})$ approximation, with L being the number of distinct values in the input. Kipouridis [44] further extended these results to tree metrics.

Research into the ℓ_∞ numerical taxonomy began in the early 1990s. It was discovered by several authors that the ℓ_∞ case of the ultrametric fitting problem is solvable in polynomial time (in fact linear time in the input) and it is the only case with that property, whereas the problem of ℓ_∞ tree fitting is APX-hard [1, 22, 39, 45]. Since then, the ℓ_∞ Best-Fit Ultrametrics/Tree-Metrics problems were extensively studied from both mathematical and computational perspectives [4, 12, 13, 22, 25, 27, 36, 50].

1.1.2 Correlation Clustering

The classic correlation clustering problem, introduced by Bansal, Blum, and Chawla [7], can be visualized as a special case of ultrametrics where the tree's depth is bounded by two. Correlation clustering serves as a fundamental building block for constructing ultrametrics and tree metrics. Despite being APX-hard [20], extensive research [7, 20, 21, 30, 31] has aimed at finding efficient approximation algorithms, with the latest being a 1.437-approximation [16]. Correlation clustering also boasts a rich body of literature and has been extensively studied across various models designed for large datasets, including streaming [2, 6, 9, 31], MPC [29], MapReduce [23], and dynamic models [8, 11, 33].

1.1.3 Metric Violation Distance

Another counterpart of the ultrametric violation distance problem is the metric violation distance problem, which requires embedding an arbitrary distance matrix into a metric space while minimizing the ℓ_0 objective. While only a hardness of approximation of 2 is known, [38, 40, 41] provided algorithms with an approximation ratio of $O(OPT^{1/3})$. An exponential improvement in the approximation guarantee to $O(\log n)$ was achieved in [26]. The maximization version of this problem is also well-motivated by its application in designing metric filtration schemes for analyzing chromosome structures, as studied in [37].

1.2 Our Contributions

In this work, we examine the problem of fitting tree metrics and ultrametrics in the semi-streaming model, focusing on the ℓ_0 and ℓ_∞ objectives. Note that storing the tree alone requires $\Omega(n)$ word space. Since we are working within the semi-streaming model, where $\tilde{O}(n)$ space is permitted, this is a natural consideration. Our results apply to the most general semi-streaming settings where the entries of the input distance matrix, of size n^2 , arrive one by one in some arbitrary order, possibly adversarially. Notably, all our algorithms require either one or two passes over the data while achieving constant factor approximations in polynomial time. Before discussing the key contributions of this work, we provide a formal definition of the problem.

Problem: ℓ_p Best-Fit Ultrametrics/Tree-Metrics

Input: A set V and a distance matrix $D : \binom{V}{2} \rightarrow \mathbb{R}_{>0}$.

Desired Output: An ultrametric (resp. tree metric) T that spans V and fits D in the sense of minimizing:

$$\|T - D\|_p = \sqrt[p]{\sum_{uv \in \binom{V}{2}} |T(uv) - D(uv)|^p}$$

For $p = 0$, the aim is to minimize the total number of errors. In other words, each pair comes with a request regarding their distance in the output tree, and our goal is to construct a tree that satisfies as many of these requests as possible, minimizing the total number of pairs whose distances are altered. This fundamental problem for ultrametrics, also known as the *Ultrametric violation distance* problem, was first investigated in [26], in which a novel constant-factor approximation algorithm in the RAM model was developed. Charikar and Gao [19] further improved the approximation guarantee to 5.

We present for this problem a single-pass algorithm, in the semi-streaming setting, that provides a constant approximation and succeeds with high probability. We remark that straightforwardly adapting this algorithm in the RAM model yields a near-linear time

algorithm ($\tilde{O}(n^2)$, while the input size is $\Theta(n^2)$), improving over the best known $\Omega(n^4)$ time from [26].² Due to space constraints, several proofs are deferred to the full version [17].

► **Theorem 1.** *There exists a single pass polynomial time semi-streaming algorithm that w.h.p. $O(1)$ -approximates the ℓ_0 Best-Fit Ultrametrics problem.*

Following, we show that this result also implies an approximation for the ℓ_1 objective.

► **Corollary 2.** *Let δ (resp. Δ) be the smallest (resp. largest) absolute difference between distinct distances in D , for an ℓ_1 Best-Fit Ultrametrics instance. There exists a single pass polynomial time semi-streaming algorithm that w.h.p $O(\Delta/\delta)$ -approximates the ℓ_1 Best-Fit Ultrametrics problem.*

Proof. Let T_0 (resp. T_1) be an ultrametric minimizing $\|T_0 - D\|_0$ (resp. $\|T_1 - D\|_1$), and T be the output from Theorem 1 (thus $\|T - D\|_0 = O(\|T_0 - D\|_0)$).

It holds that $\|T_1 - D\|_1 \geq \|T_1 - D\|_0 \delta$, as in the ℓ_1 objective we pay at least δ for each pair having a disagreement. Similarly $\|T - D\|_1 \leq \|T - D\|_0 \Delta = O(\|T_0 - D\|_0 \Delta) = O(\|T_1 - D\|_0 \Delta)$, by definition of T_0 . ◀

It is interesting to note that for the ℓ_1 objective, most recent approximation algorithms in the offline setting are not combinatorial, making it a significant challenge to adapt them to the semi-streaming model. The best known combinatorial approximation for ℓ_1 Best-Fit Ultrametrics and Tree-Metrics is $O(\Delta/\delta)$, when $\Delta/\delta = O(n)$ [3, 42], achieved through the so-called pivoting algorithm³. Unfortunately, this algorithm is very challenging to adapt to a single-pass semi-streaming setting as it generalizes the PIVOT-based algorithm of Correlation Clustering, which, despite extensive research, has not been adapted to semi-streaming settings with only a single pass without significant modifications [9, 10, 15, 51]. Surprisingly, our $O(\Delta/\delta)$ approximation for the ℓ_1 objective is derived directly from the algorithm for the ℓ_0 objective, eliminating the need to explicitly use this pivoting approach.

Further, we contrast Theorem 1 by ruling out the possibility of a single-round exact algorithm, even with sub-quadratic space and exponential time. For this, we provide a new lower bound result for the correlation clustering problem, showing that any single-pass streaming algorithm with sub-quadratic space cannot output the optimal clustering nor can maintain its cost.

► **Theorem 3.** *Any randomized single pass streaming algorithm that with probability greater than $\frac{2}{3}$ either solves the correlation clustering problem or maintains the cost of an optimal correlation clustering solution requires $\Omega(n^2)$ bits.*

We then extend this result to ℓ_p Best-Fit Ultrametrics problems for $p \in \{0, 1\}$, using the fact that correlation clustering is a special case of these problems (see e.g. [3]).

► **Corollary 4.** *For $p \in \{0, 1\}$, any randomized single pass streaming algorithm that with probability greater than $\frac{2}{3}$ either solves ℓ_p Best-Fit Ultrametrics or just outputs the error of an optimal ultrametric solution requires $\Omega(n^2)$ bits.*

² In [26] the exact running time of the algorithm has not been analyzed, but there exist inputs where it must perform $\Omega(n^2)$ repetitions of a flat-clustering algorithm that takes $\Omega(n^2)$ time per repetition.

³ The authors of [3] claim the approximation is proportional to the number of distinct distances. That is because of a simplification they make in the paper, that the distances are all consecutive positive integers starting from 1. For an example showing the $O(\Delta/\delta)$ analysis is tight, take $V = \{u_1, u_2, u_3\}$ and $D(uv) > \Delta, D(uw) = D(uv) - \delta, D(vw) = D(uv) - \Delta$. With probability $1/3$ we pick u as the pivot, and pay Δ , while the optimum solution only pays δ .

Next, we consider the ℓ_∞ objective, where the goal is to minimize the maximum error. We provide a complete characterization of ℓ_∞ Best-Fit Ultrametrics in the semi-streaming model, including a single-pass algorithm with a 2-approximation factor.

► **Theorem 5.** *There exists a single pass polynomial time semi-streaming algorithm that 2-approximates the ℓ_∞ Best-Fit Ultrametrics problem.*

We contrast Theorem 5 by showing that this is the best approximation factor achievable using a single pass, even with sub-quadratic space and exponential time.

► **Theorem 6.** *Any randomized one-pass streaming algorithm for ℓ_∞ Best-Fit Ultrametrics with an approximation factor strictly less than 2 and a success probability greater than $\frac{2}{3}$ requires $\Omega(n^2)$ bits of space.*

Moreover, we demonstrate that allowing two passes is sufficient for an exact solution. Therefore, we provide optimal tradeoffs between the number of passes and the approximation factor in all scenarios.

► **Theorem 7.** *There exists a two-pass polynomial time semi-streaming algorithm that computes an exact solution to the ℓ_∞ Best-Fit Ultrametrics problem.*

We show that all aforementioned algorithms can be extended to tree metrics. This is achieved by providing reductions to the corresponding ultrametrics problems, requiring only one additional pass over the stream. The reductions used for the ℓ_0 and ℓ_∞ objectives differ significantly from each other.

► **Theorem 8.** *There exists a two-pass polynomial time semi-streaming algorithm that w.h.p $O(1)$ -approximates the ℓ_0 Best-Fit Tree-Metrics problem.*

Using the same arguments as in Corollary 2, we obtain an analogous result for the ℓ_1 objective.

► **Corollary 9.** *Let δ (resp. Δ) be the smallest (resp. largest) absolute difference between distinct distances in D , for an ℓ_1 Best-Fit Tree-Metrics instance. There exists a two-pass polynomial time semi-streaming algorithm that w.h.p $O(\Delta/\delta)$ -approximates ℓ_1 Best-Fit Tree-Metrics problem.*

► **Theorem 10.** *There exists a two-pass polynomial time semi-streaming algorithm that 6-approximates the ℓ_∞ Best-Fit Tree-Metrics problem.*

1.3 Technique Overview.

We provide a technical overview of the most technically novel contribution of our work, namely the results regarding the ℓ_0 Best-Fit Ultrametrics algorithm in the semi-streaming model (more details in Section 3).

1.3.1 Why previous ℓ_0 approaches cannot be adapted

In general, it is difficult to ensure a hierarchical structure while providing non-trivial approximation guarantees. In Hierarchical Clustering research, such results usually rely on one of two standard approaches, namely the top-down (divisive) approach, and the bottom-up (agglomerative) approach. In fact, with the exception of [24], results for ℓ_p Best-Fit Ultrametrics ($p < \infty$) [3, 19, 24, 26, 42] all rely on the divisive approach.

1.3.1.1 Non-divisive approaches

The only relevant result applying a non-divisive approach is that of [24], which crucially relies on a large LP. Unfortunately, it is not known how to solve such an LP in streaming.

1.3.1.2 Divisive approaches

A divisive algorithm starts with the root node (containing the whole V), computes its children (subsets V' at height h) based on some division strategy, and recurses on its children. Different division strategies have been employed, with the most prominent ones using the solution to an LP, or attempting to satisfy a particular (usually randomly chosen) element called the pivot, or solving some flat clustering problem. In what follows, we discuss why existing division strategies do not work in our case.

Correlation Clustering. Perhaps the most straightforward approach is to solve a (flat) clustering problem; for each pair of vertices $u, v \in V'$, we ideally want them together if $h > D(uv)$, and apart otherwise. This corresponds to the Correlation Clustering problem, which returns a clustering violating as few of our preferences as possible. Unfortunately, this approach does not work for ℓ_0 Best-Fit Ultrametrics, as certain choices that appear good locally (on a particular height h) may be catastrophic globally.

The first result for ℓ_0 Best-Fit Ultrametrics. The authors of [26] overcame the shortcomings of Correlation Clustering as a division strategy by solving a particular flavor of it, called Agreement Correlation Clustering. This guaranteed further structural properties that could be leveraged to provide $O(1)$ approximation for ℓ_0 Best-Fit Ultrametrics. However this approach is too strong to guarantee in streaming, since one can recover adjacency information with black-box calls to an Agreement Correlation Clustering subroutine. This of course requires $\Theta(n^2)$ bits of memory, while in streaming we only have $\tilde{O}(n)$.

Other results for ℓ_0 Best-Fit Ultrametrics. The other results for ℓ_0 Best-Fit Ultrametrics are pivot-based and do not work in our case. Indeed, one of them [19] is based on a large LP for which no streaming solution is known, while the other one [26] is combinatorial but with approximation factor $\Omega(\log n)$.

1.3.2 Our techniques

ℓ_0 Best-Fit Ultrametrics

Our streaming algorithm is a divisive algorithm. In the divisive framework, each level of the tree is defined by a distinct distance from the input, which allows each level to be visualized as an instance of the correlation clustering problem. In this instance, two vertices are connected if their distance is at most the threshold associated with that level; otherwise, they are not connected. Following this, different layers of the ultrametric tree are built by repeatedly applying a division strategy in a top-down fashion. Here, we highlight the techniques we develop to design a semi-streaming algorithm that uses only a single pass and computes an $O(1)$ approximation for the ℓ_0 Best-Fit Ultrametrics problem.

Distances Summary. The first fundamental challenge is identifying which distances should be preserved in the constructed ultrametric, given that the input may contain $\Omega(n^2)$ distinct distances. A divisive algorithm may need to perform its division strategy on every level

defined by such a distance (of course, sometimes it may decide not to divide anything); however, in the semi-streaming setting, we cannot even afford to store all these distances. Instead, we work with a compressed set of distances that effectively captures all important information. More formally, we focus on distances d for which there exists at least one vertex u such that the number of vertices with distance less than d from u is significantly smaller than the number of vertices with distance at most d from u . Using this notion, we demonstrate that it is sufficient to consider only a near-linear number of distances to achieve a good approximation.

Agreement Sketches. A key component in many (flat) clustering algorithms (including the first algorithm for Correlation Clustering [7], which inspired many others, such as [6, 8, 26, 29]) involves comparing the set of neighbors of two vertices. While our division strategy also builds on such comparisons, both the hierarchical nature and streaming constraints of our setting present unique challenges. In Correlation Clustering each vertex has only a single set of neighbors, however, in our hierarchical setting, each layer of the tree is associated with a different distance threshold, producing different sets of neighbors for a vertex. In the worst case, there can be $O(n)$ such sets, and building a sketch for each can require up to quadratic space.

To address this, we build a new sketch for a node only when its set of neighbors changes significantly. The intuition here is that if the neighborhood of a node has not changed substantially, then a precomputed sketch for a nearby neighborhood will suffice. However, implementing this in a semi-streaming setting, where distances between pairs can arrive in any arbitrary order, is challenging. Since we cannot store the distances to all other nodes from a given node simultaneously, identifying significant changes in a node's neighborhood becomes difficult. To manage this, we develop a new technique that combines random sampling with a pruning strategy, ensuring that the overall space required to store all the sketches is $\tilde{O}(n)$.

In this approach, we build each sketch by randomly sampling nodes. Assuming the neighborhood size has dropped substantially, we expect the correct sketch to reach a certain size. Notably, the set of neighbors of a node only shrinks as the distance decreases. Thus, for a specific weight threshold, if the sample (or sketch) size grows considerably, this indicates that the neighborhood has not changed much, so we disregard that weight threshold and delete the corresponding sample from the sketch. Specifically, for each node, we build and store sketches when the neighborhood size shrinks by a constant factor. Following this, we consider at most $\log n$ different sizes, and storing the sketch for all sizes takes only polylogarithmic space for a node. Therefore, the total space required to store the sketches for all the nodes is bounded by $\tilde{O}(n)$. Moreover, the sketches ensure that for each node u and each weight w , there exists a weight w' such that the neighboring nodes of u at w and w' differ very little, and we have built a sketch corresponding to the neighboring set at weight w' .

Across-Levels Correlations. In divisive algorithms, while building a new level of the ultrametric tree, the recursions performed depend on the divisions computed at previous recursion levels. In this sense, the divisive framework can be viewed as an adaptive adversary for the division strategy we need to perform. This is not an issue when deterministic division strategies are used (e.g. as in [26]), but it becomes particularly problematic in our case, where we are forced to use random sketches because of the issue with the $\Omega(n^2)$ distinct distances.

The challenge here arises from the fact that for a given vertex we do not build a new sketch for each level of the tree. Instead, we only construct a sketch when the set of neighbors changes substantially. Consequently, multiple levels of the tree must reuse the same sketch, which increases the correlation among clusters at different levels. This makes it difficult to ensure concentration bounds when limiting the overall error.

To address this, our approach aims to “limit” the dependencies by ensuring our algorithm has only a logarithmic recursion depth (as opposed to the $\Omega(n^2)$ recursion depth in straightforward divisive approaches). This allows us to afford independent randomness by using a new sketch for each recursion depth. To reduce the recursion depth, we make the following observation: if the correlation clustering subroutine identifies a large cluster (e.g., containing a 0.99 fraction of the vertices), we can detect this cluster without explicitly applying the correlation clustering algorithm (thus omitting the requirement of using a sketch). This is because all vertices within this cluster have large degrees, while those outside have very small degrees. Therefore, it suffices to identify the vertices with small degrees and remove them to generate the new cluster. It is important to note that the degree calculation must consider the entire graph, not just the subgraph induced by the current cluster being considered. Otherwise, intra-recursion dependencies could be introduced, and thus the logarithmic recursion depth guarantee may not suffice.

Within-Level Correlations. Correlation issues do not only occur vertically (across levels), but also horizontally (within the same level), as most algorithms for correlation clustering compute a cluster, and then recurse on the rest of the elements. However in our case, such an adaptive construction may lead to the possibility of reusing sketches, making it difficult to ensure concentration.

We overcome these issues in several ways. First, we use these sketches to compute the agreement among vertices (i.e., computing the similarity between the set of neighbors of each pair of vertices) before we start constructing any clusters. Finally we propose an algorithm that is relying solely on our agreement sketches and is decomposed into independent events, thus only requiring us to consult the sketches of each layer only once. By using the agreements precomputation and our proposed clustering algorithm we ensure concentration while limiting the error for all clusters within a level.

S -Structural Clustering. Finally, as argued in Section 1.3.1, we need a division strategy that is different from the existing Agreement Correlation Clustering of [26]. That is because it can be proven that solving Agreement Correlation Clustering on arbitrary subgraphs requires $\Omega(n^2)$ bits of memory.

Instead, we introduce S -Structural Clustering, which is inspired by Agreement Correlation Clustering. The key distinction is that now we require a clustering of S to satisfy the structural properties, while also considering edges with only one endpoint in S . This distinction is exactly what allows us to generalize our proposed algorithm to solve S -Structural Clustering by relying solely on the global neighborhoods of its vertices. Interestingly, the resulting time complexity of our general algorithm only depends on the size of the subgraph, as we compress all the necessary global information through our sketches. Finally, we remark that both the construction of the sketches (Section 3.1) and the introduction of the S -Structural Clustering (Section 3.2.2) are two novel contributions of our work and could be of independent interest.

ℓ_0 Best-Fit Tree-Metrics

In [44] it is shown how to reduce ℓ_0 Best-Fit Tree-Metrics to ℓ_0 Best-Fit Ultrametrics. In this approach, however, one needs to create n different instances of ℓ_0 Best-Fit Ultrametrics, which is not feasible in the semi-streaming model. In this work, we show that randomly solving a logarithmic number of these n different instances suffices.

Our initial approach requires 3 passes over the stream. One for a preprocessing step implicitly constructing the ℓ_0 Best-Fit Ultrametrics instances, one to solve these instances (and post-process them to extract trees that solve the original problem), and a final one to figure which one of the logarithmically many trees we need to output (the one with the smallest cost is picked).

We further improve the number of passes to 2, by eliminating the need for the final pass. To do that, we note that there are many trees with “tiny” cost related to the input; let A be the set containing these trees. By triangle inequality, all trees in A have “small” cost related to each other. If we create a graph with the trees as nodes, and an edge between two nodes when the cost relative to each other is small, we then show that with high probability this graph contains a big clique. Finally, we show that any node (corresponding to a tree) from a big clique is a good enough approximation to the original input, even if it is not in A .

ℓ_∞ Results

Regarding ℓ_∞ Best-Fit Ultrametrics, we show that the existing exact algorithm [39] can be straightforwardly adapted to a 2-pass semi-streaming algorithm. Naturally, as the problem has been solved exactly, no research has focused on approximation algorithms. In this work we show that the solution to a related problem (ℓ_∞ Min-Decrement Ultrametrics) 2-approximates ℓ_∞ Best-Fit Ultrametrics. Then we adapt the exact solution for ℓ_∞ Min-Decrement Ultrametrics [39] to obtain a single pass semi-streaming algorithm.

We also show that no single-pass semi-streaming algorithm can give a better-than-2 approximation, for otherwise we could compress any graph in $\tilde{O}(n)$ space. Together, these results completely characterize ℓ_∞ Best-Fit Ultrametrics in the semi-streaming setting, regarding the optimal number of passes and the optimal approximation factor.

For ℓ_∞ Best-Fit Tree-Metrics, there exists a reduction to Ultrametrics [1], blowing up the approximation by a factor 3. Adapting it in the semi-streaming requires one additional pass through the stream. Using it with our 2-approximation for ℓ_∞ Best-Fit Ultrametrics (rather than with the exact algorithm, as done in [1]), we need 2 passes (instead of 3).

2 Preliminaries

We start by presenting useful notations we employ throughout the text. We use uv to denote an unordered pair $\{u, v\}$. We use the term distance matrix to refer to a function from $\binom{V}{2}$ to the non-negative reals. Let D be a distance matrix. For easiness of notation, we use $w_{\max} = \max_{uv} D(uv)$. We slightly abuse notation and say that for any $u \in V$, $D(uu) = 0$. For $p \geq 1$, $\|D\|_p = \sqrt[p]{\sum_{uv \in \binom{V}{2}} |D(uv)|^p}$ is the ℓ_p norm of D . We extend the notation for $p = 0$. In this case, $\|D\|_0$ denotes the number of pairs uv such that $D(uv) \neq 0$. We even say $\|D\|_0$ is the ℓ_0 norm of D , despite ℓ_0 not being a norm.

If T is a tree and u, v are two nodes in T , then we write $T(uv)$ to denote the distance between u and v in T . An ultrametric is a metric (V, D) with the property that $D(uv) \leq \max\{D(uw), D(vw)\}$ for all $u, v, w \in V$. It holds that (V, D) is an ultrametric iff there exists a rooted tree T spanning V such that all elements of V are in the leaves of T , the depth of all leaves is the same, and $D(uv) = T(uv)$ for all $u, v \in V$. We call trees with these properties *ultrametric trees*.

In the semi-streaming model, the input is again a distance matrix D on a vertex set V . Let $n = |V|$. Our algorithm has $\tilde{O}(n)$ available space, and the entries of D arrive one-by-one, in an arbitrary order, as pairs of the form $(uv, D(uv))$. For simplicity, we use the standard assumption that each distance $D(uv)$ fits in $O(\log n)$ bits of memory.

We let E_w be the set of pairs uv such that $D(uv) \leq w$. We define $N_w(u)$, the set of neighbors of u at level w , to be the vertices v such that $uv \in E_w$ (including u itself), and the degree of u at level w to be $d_w(u) = |N_w(u)|$. We even write $N(u)$ and $d(u)$ (instead of $N_w(u)$ and $d_w(u)$) when E_w is clear from the context. Given an ultrametric tree T , a cluster at level w is a maximal set of leaves such that every pairwise distance in T is at most w . It is straightforward that a cluster at level w corresponds to the set of leaves descending from a node of T at height $w/2$. Abusing notation, and only when it is clear from the context, we refer to this node as a cluster at level w as well.

Regarding ℓ_0 , it is sufficient to focus on ultrametrics where the distances between nodes are also entries in D . That is because if an ultrametric T does not have this property, we can create an ultrametric T' with this property such that $\|T' - D\|_0 \leq \|T - D\|_0$ (folklore). To do this, simply modify every distance d in T to the smallest entry in D that is at least as large as d (if no such entry in D exists, then we modify d to be the maximum entry in D).

3 ℓ_0 Ultrametrics

In this section, we show how to $O(1)$ -approximate ℓ_0 Best-Fit Ultrametrics with a single pass in the semi-streaming model. Formally we show the following.

► **Theorem 1.** *There exists a single pass polynomial time semi-streaming algorithm that w.h.p. $O(1)$ -approximates the ℓ_0 Best-Fit Ultrametrics problem.*

Our algorithm consists of two main phases. In the streaming phase, we construct efficient sketches that capture the essential information of the input matrix D . That is, we store a compressed representation of D , denoted as $\tilde{D}$, which, unlike D , has a reduced size of $\tilde{O}(n)$ rather than $O(n^2)$ values (hereafter called *weights*). Yet, we will show in Section 3.1 that for every weight $w \in D$ and every $u \in V$, a weight $\tilde{w} \in \tilde{D}$ is stored, such that $N_w(u)$ and $N_{\tilde{w}}(u)$ are roughly the same. This guarantee enables us to approximate both the size of a neighborhood and the size of the intersection for two different neighborhoods.

The second step is a post-stream process that carefully utilizes the precomputed sketches while addressing the adaptivity challenges discussed in Section 1.3.2. In Section 3.2 we show how to compute the S -Structural Clustering subroutine, which we will use as our division strategy. In Section 3.3 we present our main algorithm, which uses this subroutine and the distances summary as black-boxes to construct the ultrametric tree. In the full version, we establish the necessity of approximation in the streaming setting by proving that computing an optimal solution requires $\Omega(n^2)$ bits of memory.

3.1 Construction of Sketches

This section outlines the process for constructing sketches that enable our algorithm's implementation. For now we consider large neighborhoods of size $\Omega(\log^4 n)$. While a similar approach was used in [29]⁴, for the problem of correlation clustering, the challenge here is

⁴ In [29], the authors claim polylogarithmic size sketches for each vertex. However, we are unable to verify this. Specifically, the random set is constructed by selecting each vertex with probability $\min\left\{\frac{a \log n}{\beta j}, 1\right\}$,

different. Unlike correlation clustering, where each vertex has only a single set of neighbors, each layer of the tree in our context is associated with a different distance threshold. Thus each varying threshold can produce a different set of neighbors for a vertex. In the worst case, there can be n such sets, and building a sketch for each changing set of neighbors for each vertex can require up to quadratic space (or even cubic, if implemented naively).

We denote the weight of an edge $e = uv$ by $w(e) = D(uv)$. Each sketch is constructed for a specific vertex with a predetermined size chosen from the set $\mathbb{S} = \{n, \frac{n}{(1+\zeta)}, \frac{n}{(1+\zeta)^2}, \dots, \log^4 n\}$, where ζ is a small constant parameter to be adjusted. Each sketch will encapsulate a neighborhood of the vertex of size roughly s , and allow us to compare the common intersection of two different neighborhoods. Let w_s^v be the largest weight for which $\frac{s}{1+\zeta} < |N_{w_s^v}(v)| \leq s$. We call size s *relevant* for vertex v if such w_s^v exists.

To obtain the sketches, for each $s' \in \mathbb{S}$, we start by generating a random subset $R_{s'} \subseteq [n]$ by sampling each vertex from V independently with probability $\log^2 n/s'$, prior to processing the stream. For each vertex v , each relevant size s , and each s' satisfying $\frac{1}{2}s \leq s' \leq s$, we define a sketch $\mathcal{S}_{s,s'}^v$. Every sketch consists of (i) an estimate of the parameter w_s^v , denoted by $\tilde{w}_s^v$, and (ii) an (almost) random sample of $v \times N_{\tilde{w}_s^v}(v)$ of size $O(\log^2 n)$, along with the weight of each sampled edge. To achieve this, we store a collection of edges $C_1^v, \dots, C_\ell^v \subseteq v \times N_{\tilde{w}_s^v}(v)$, where all edges in C_i^v have the same weight w_i^v , which we also store alongside C_i^v , and let C_ℓ^v be the collection corresponding to the largest weight. Furthermore, we ensure that the overall size of all collections $\sum_{i \in [\ell]} |C_i^v|$ is $O(\log^3 n)$ bits.

The purpose of incorporating two size parameters, s and s' into the sketch is to enable comparisons of neighborhoods that have slightly different, but relatively close, sizes. Yet, for simplicity, the reader may assume $s = s'$ for the following construction and claims. We now describe the process of constructing a specific sketch $\mathcal{S}_{s,s'}^v$ given the input stream:

1. Initialize *counter* = 0, $w_m = 0$.
2. If e is not incident on v , continue to the next edge.
3. Else, if $w_m \neq 0$ and $w(e) \geq w_m$, continue to the next edge.
4. Else if $u \notin R_{s'}$, where $e = (u, v)$, continue to the next edge.
5. Otherwise proceed as follows:
 - a. If there is a collection of edges C_i^v with an associated weight of $w(e)$, add the edge e to C_i^v . Otherwise, create a new collection C_i^v containing the edge e alongside $w(e)$.
 - b. Increase *counter* by 1.
 - c. If *counter* $> (1 + \frac{\zeta}{2}) \frac{s}{s'} \log^2 n$, delete C_ℓ^v the collection with the largest weight associated with it, set $w_m = w_\ell^v$ and *counter* = *counter* - $|C_\ell^v|$.

After processing all the edges, output $\mathcal{S}_{s,s'}^v = \bigcup_{i \in [\ell]} (C_i^v \times \{w_i^v\})$, furthermore if $s = s'$ we let $\tilde{w}_s^v = \max_{i \in [\ell]} w_i^v$ and call it a *governing weight* of the sketches parametrized by v and s . Namely, $\tilde{w}_s^v$ is the weight associated with the neighborhood a sketch parametrized by v and s is encapsulating. The next claim shows that this sketches can be stored in semi-streaming settings.

▷ **Claim 11.** The sketches $\mathcal{S}_{s,s'}^v$, where $v \in V$ and $s \in \mathbb{S}$, can be constructed and stored in $O(n \log^4 n)$ bits.

The next claim demonstrates that for every w_s^v there is a sketch with governing weight $\tilde{w}_s^v$ such that $|N_{\tilde{w}_s^v}(v)|$ is a good approximation to $|N_{w_s^v}(v)|$.

where a is a constant. Since j is at most $O\left(\frac{\log n}{\beta}\right)$, the probability of selecting a vertex is at least $\min\{\Omega(a), 1\}$, which is a constant. Thus, each random set is of size $\Omega(n)$. Therefore, for a vertex v with $|N(v)| = \Omega(n)$, the sketch size will be of size $\Omega(n)$.

▷ **Claim 12.** With high probability, for each vertex v and each relevant size $s \in \mathbb{S}$, we have $(1 - \zeta) |N_{w_s^v}(v)| \leq |N_{\tilde{w}_s^v}(v)| \leq (1 + \zeta)^2 |N_{w_s^v}(v)|$.

We now extend this result for every weight w , and show how to obtain a sketch that is a good approximation to $N_w(v)$.

▷ **Claim 13.** For each vertex v and each weight w with $|N_w(v)| \geq \log^4 n$, we can report a sketch associated with size s and governing weight $\tilde{w}_s^v$, such that with high probability, $\frac{|N_{\tilde{w}_s^v}(v)|}{1+5\zeta} \leq |N_w(v)| \leq \frac{|N_{\tilde{w}_s^v}(v)|}{1-\zeta}$.

We conclude this section by providing another data structure for storing the nearest $2\log^4 n$ neighbors for each vertex v , denoted by $N_{\text{close}}(v)$. This will allow us to compare neighborhoods of small size. The implementation of $N_{\text{close}}(v)$ is done using a priority queue with predefined and fixed size $2\log^4 n$. We add every edge incident to v to the priority queue $N_{\text{close}}(v)$ together with the associated edge. This leads to the following claim.

▷ **Claim 14.** For each vertex v , $N_{\text{close}}(v)$ can be stored in $O(\log^5 n)$ bits, and it contains the nearest $2\log^4 n$ neighbors of v .

In this section, we have outlined the construction of sketches with a total space of $\tilde{O}(n)$, showing that for each vertex v and weight w , we can report a sketch associated with governing weight $\tilde{w}_s^v$ that with high probability is a random sample of a neighborhood $N_{\tilde{w}_s^v}(v)$ that is roughly of the same size as $N_w(v)$. In the next section, we will demonstrate how these sketches can be utilized to estimate the size of the symmetric difference in a way that supports the algorithm's requirements, justifying the need for maintaining several sketches for each choice of v and s .

3.2 Structural Clustering

In this section, we introduce an algorithm that requires a single pass over the input stream to solve S -Structural Clustering. This extends the notion of Agreement Correlation Clustering from [26], to which we refer as V -Structural Clustering. Our semi-streaming algorithm hinges on the key idea that clusters should be formed from vertices that share almost similar neighborhoods. We emphasize that our algorithm is also applicable in the standard RAM (non-streaming) setting and runs in near-linear $\tilde{O}(|S|^2)$ time, for $S \subseteq V$, improving the $\Omega(|V|^3)$ time algorithm previously known for V -Structural Clustering.

We begin our presentation in Section 3.2.1 with an algorithm solving V -Structural Clustering, designed to be adapted in the semi-streaming model. Then, in Section 3.2.2 we show how our proposed algorithm could also be extended to compute S -Structural Clustering, which is our division strategy for constructing ultrametrics in Section 3.3. Finally, in Section 3.2.3 we show how to actually implement these algorithms in the semi-streaming model, by utilizing our sketches outlined in Section 3.1.

3.2.1 Algorithm for V-Structural Clustering

We begin by solving the Structural Clustering problem for the entire vertex set V . Our graph is $(V, E = E_W)$ for some weight W . First we present the definitions of *agreement* and *heavy* vertices as in [29]. The parameters β and ϵ that appear in the following definitions are sufficiently small constants. Furthermore we denote by Δ the symmetric difference between two sets, that is, $A \Delta B = A \setminus B \cup B \setminus A$.

► **Definition 15** (agreement). *Two vertices u, v are in β -agreement iff $|N(u) \Delta N(v)| < \beta \max\{d(u), d(v)\}$, which means that u, v share most of their neighbors. $A(u)$ is the set of vertices in β -agreement with u .*

► **Definition 16** (heavy). *We say that a vertex u is ϵ -heavy if $|N(u) \setminus A(u)| < \epsilon d(u)$, which means that most of its neighbors are in agreement with u . Denote by $H(u)$ the ϵ -heaviness indicator of vertex u .*

Computing the β -agreement set $A(u)$ of a vertex and its ϵ -heaviness indicator $H(u)$ is a crucial part of the algorithm. Normally, both can be computed exactly by applying the definitions, even using a deterministic algorithm. However, in the semi-streaming model, we can only approximate $A(u)$ and $H(u)$ with high probability. In Section 3.2.3 we show that, using the sketches outlined in Section 3.1, we can achieve a sufficient approximation that allows us to solve the Structural Clustering for V with high probability.

We allow the following relaxations. Let $A(v)$ be a set containing all vertices that are in 0.8 agreement with v and no vertices that are not in β agreement with v . Similarly let $A^3(v)$ be a set containing all vertices that are in 2.4β -agreement with v and no vertices that are not in 3β -agreement with v . And finally let $H(v)$ be a method that returns true if v is ϵ -heavy and false if v is not 1.2ϵ -heavy.

With these tools and definitions at our disposal, we can introduce Algorithm 1. It is important to note that this algorithm is executed, using the sketches alone, post stream process. Given the respective sketches, the time complexity of the algorithm is $\tilde{O}(|V|^2)$.

■ **Algorithm 1** V -Structural-Clustering.

-
- 1: **for** $v \in V$ **do**
 - 2: **if** $H(v)$ **and** v is not already included in an existing cluster **then**
 - 3: Create a new cluster $A^3(v)$
 - 4: Create singleton clusters for all remaining vertices.
-

We now prove that Algorithm 1 is guaranteed to return a set of disjoint clusters that satisfy the required structural properties. We are referring to the special properties of V -Structural Clustering, which are expressed in terms of the definitions of *important* and *everywhere dense* groups as in [26].

► **Definition 17** (important group). *Given a correlation clustering instance, we say that a group of vertices C is important if for any vertex $u \in C$, u is adjacent to at least $(1 - \epsilon)$ fraction of the vertices in C and has at most ϵ fraction of its neighbors outside of C .*

► **Definition 18** (everywhere dense). *Given a correlation clustering instance, we say that a group of vertices C is everywhere dense if for any vertex $u \in C$, u is adjacent to at least $\frac{2}{3}|C|$ vertices of C .*

Lemma 19 outlines the supplementary properties required for a correlation clustering algorithm to qualify as structural, demonstrated in the context of Algorithm 1.

► **Lemma 19** (structural properties). *Suppose $\beta = 5\epsilon(1 + \epsilon)$ for a small enough parameter $\epsilon \leq 1/95$. Let $\mathcal{C}$ be the set of clusters returned by Algorithm 1. Then, for any important group of vertices $C' \subseteq V$, there is a cluster $C \in \mathcal{C}$ such that $C' \subseteq C$, and C does not intersect any other important groups of vertices disjoint from C' . Moreover, every cluster $C \in \mathcal{C}$ is everywhere dense.*

Proof. In order to prove Lemma 19, we require the following claims.

▷ **Claim 20.** Suppose $(1 - 3\beta - 1.2\epsilon)(1 - 3\beta) > \frac{1}{2}$. Assume u_1, u_2 are two vertices for which $H(u_1)$ and $H(u_2)$ both return true. If u_2 is not part of $A^3(u_1)$, then the sets $A^3(u_1)$ and $A^3(u_2)$ are disjoint.

▷ **Claim 21.** Suppose $1.2\epsilon \leq 1/3 - 6\beta$. Every cluster C returned by Algorithm 1 is everywhere dense.

▷ **Claim 22.** Suppose $0.8\beta \geq 2\epsilon \frac{2-\epsilon}{1-\epsilon}$. Let the pair of vertices u, v belong to the same important group, then u, v are in 0.8β -agreement.

▷ **Claim 23.** Suppose $2\epsilon \leq 1 - 3\beta$. Let u, v belong to two disjoint important groups, then u, v are not in 3β -agreement.

Following Claim 20 and Claim 21 the clusters returned by the algorithm are disjoint and everywhere dense. Next, we show the properties related to important groups. First, let u be a vertex in some important group C , then u has at least a $1 - \epsilon$ fraction of its neighbors within C , and according to Claim 22, it is in 0.8β -agreement with all vertices in C . Thus, every vertex that belongs to an important group is ϵ -heavy, which also implies that any such vertex is part of a non-singleton cluster.

Now, assume u, v belong to the same important group C , and that u belongs to a cluster $A^3(h)$ created in step 3 of the algorithm, we will show that $C \subseteq A^3(h)$. Because u, v are in 0.8β -agreement, u also belongs to the set $A^3(v)$. However, the intersection of $A^3(h)$ and $A^3(v)$ at vertex u implies, based on Claim 20, that vertex v necessarily belongs to the cluster $A^3(h)$.

It remains to prove that if u and v belong to disjoint important groups, they cannot be part of the same cluster. Suppose, contrarily, that both u and v belongs to $A^3(h)$. Note that u is ϵ -heavy as it belongs to some important group. As such, since u in $A^3(h)$, the sets $A^3(u)$ and $A^3(h)$ are not disjoint (as both contains u). According to Claim 20, as both u, h are ϵ -heavy, h must also be in $A^3(u)$. Similarly, h belongs to $A^3(v)$ as well. Thus, $A^3(u)$ and $A^3(v)$ intersect at h . However, by Claim 20, this intersection implies that u and v must be in 3β -agreement, contradicting Claim 23. ◀

3.2.2 S-Structural Clustering

So far we have provided an algorithm for V -Structural Clustering. However, what we really need for ℓ_0 Best-Fit Ultrametrics is the more general problem of S -Structural Clustering as defined in Lemma 24. Note that it is different from Agreement Correlation Clustering on the induced subgraph $G[S]$, because S -Structural Clustering also considers edges with only one endpoint in S .

► **Lemma 24.** Suppose $\beta = 5\epsilon(1 + \epsilon)$ for a small enough parameter $\epsilon \leq 1/95$. For every $S \subseteq V$ we can output a set of clusters $\mathcal{C}$. This clustering ensures that for every important group of vertices $C' \subseteq S$, there is a cluster $C \in \mathcal{C}$ such that $C' \subseteq C$, and C does not intersect any other important groups of vertices contained in $S \setminus C'$. Moreover, every cluster $C \in \mathcal{C}$ is everywhere dense.

In the rest of this section we provide a construction of S -Structural Clustering by reducing the problem to V -Structural Clustering. We start by generalizing the definitions 15 and 16 presented earlier.

► **Definition 25** (subset agreement). *Two vertices $u, v \in S$ are in β -agreement inside S if $|N(u) \triangle N(v)| + 2|N(u) \cap N(v) \cap \bar{S}| < \beta \max\{d(u), d(v)\}$, which means that u, v share most of their neighbors inside S . Denote by $A_S(u)$ the set of vertices that are in β -agreement with u inside S .*

► **Definition 26** (subset heavy). *A vertex $u \in S$ is ϵ -heavy inside S if $|N(u) \setminus A_S(u)| < \epsilon d(u)$. This means that most of its neighbors are in agreement with u inside S . Denote by $H_S(u)$ the ϵ -heaviness indicator of vertex u inside S .*

Note that definitions 25 and 26 are more general than definitions 15 and 16, since we can derive the latter ones by substituting $S = V$. The extra term $2|N(u) \cap N(v) \cap \bar{S}|$ has an intuitive meaning that will become clear later during the reduction. Similarly to the previous section we need to approximate the new sets $A_S(u)$, $A_S^3(u)$ and $H_S(u)$, which we do in Section 3.2.3.

We finally prove Lemma 24 by reducing S -Structural Clustering, for a set S in the correlation clustering instance G , to V -Structural Clustering, in a specially constructed instance G_S . The instance G_S can be seen as a transformation of G that preserves all internal edges within S and replaces all external neighbors $v \in \bar{S}$ of internal vertices $u \in S$ with dummy vertices, ensuring that our subset-specific definitions of agreement (Definition 25) and heaviness (Definition 26) applied to G correspond exactly to the original definitions 15 and 16 applied to G_S . Therefore, to produce the desired S -Structural Clustering, it suffices to run Algorithm 1 on S using $A_S(u)$, $A_S^3(u)$, $H_S(u)$ as the parameters. The complete proof can be found in the full version.

3.2.3 Computing Agreements

Building on the algorithm from Section 3.2.2, we now describe its adaptation to the semi-streaming model. Since this algorithm only requires computing approximations to β -agreements in S and heaviness queries, we prove:

► **Lemma 27.** *The following statements hold with high probability:*

1. *For a given $\gamma \in \{\beta, 3\beta\}$ and every $u, v \in S$, we can output:*
 - “yes” if $|N(u) \triangle N(v)| + 2|N(u) \cap N(v) \cap \bar{S}| < 0.8\gamma \max\{d(u), d(v)\}$
 - “no” if $|N(u) \triangle N(v)| + |N(u) \cap N(v) \cap \bar{S}| \geq \gamma \max\{d(u), d(v)\}$
2. *For every $u \in S$, we can output:*
 - “yes” if $|N(u) \setminus A_S(u)| < \epsilon d(u)$
 - “no” if $|N(u) \setminus A_S(u)| > 1.2\epsilon d(u)$

We can now establish S -Structural Clustering in the semi-streaming model.

► **Theorem 28.** *Suppose $\beta = 5\epsilon(1 + \epsilon)$ for a small enough parameter $\epsilon \leq 1/95$. Given access to the sketches of all vertices in S , and w.h.p., for every $S \subseteq V$ we can output a set of clusters $\mathcal{C}$. This clustering ensures that for every important group of vertices $C' \subseteq S$, there is a cluster $C \in \mathcal{C}$ such that $C' \subseteq C$, and C does not intersect any other important groups of vertices contained in $S \setminus C'$. Moreover, every cluster $C \in \mathcal{C}$ is everywhere dense.*

3.3 Main Algorithm

Due to space limitations, we provide only a description and pseudocode of the main algorithm.

To run our main algorithm it suffices to obtain access to certain black-boxes established in the previous sections. Ideally, we would like to have access to a summary of the input distances, to estimations of neighborhood sizes, and to be able to repeatedly compute S -

Structural Clustering for instances given by an adaptive adversary. We show that even though we cannot generally guarantee the last requirement, it suffices to guarantee it for a particular (technical) type of adaptive adversary.

We first need the following definitions. For a weight $w \in D$, let $\hat{w}$ be the smallest weight in $\tilde{D}$ such that $\hat{w} > w$. Similarly, for any w we let $\tilde{w}$ be the largest value in $\tilde{D}$ smaller than w . We say that $\tilde{D}$ is a *compressed set* if for $w \notin \tilde{D}$, $\tilde{D}$ has the property that $d_w(u) \leq (1 + \delta)d_{\tilde{w}}(u)$ for all u . Finally let $\widetilde{d_w}(u)$ be a function with $\widetilde{d_w}(u) \in [(1 - \lambda)d_w(u), (1 + \lambda)d_w(u)]$ for a sufficiently small constant λ .

Our algorithm (see Algorithm 2 for the pseudocode) is a divisive algorithm running S -Structural Clustering at each level to divide a cluster. However, it then performs a different division strategy for the largest cluster. This different strategy for the largest cluster allows us to guarantee that each vertex only participates in a logarithmic number of S -Structural Clustering computations, and is only possible if the size of the largest cluster has not dropped by a constant factor.

More formally, our algorithm takes as argument a set S (initially the whole V) and a distance w (initially the maximum distance). First, it creates a tree-node A at distance $w/2$ from the leaves, whose leaves-descendants are all the vertices in S . Then it uses an S -Structural Clustering subroutine, and for each cluster C' with size at most $0.99|S|$ it recurses on $(C', \tilde{w})$. The roots of the trees created from each of these recursions then become children of A .

Subsequently, for the largest cluster C we perform the following postprocessing: Let $w' \leftarrow \tilde{w}$ and $w'' \leftarrow \widetilde{w'}$.

- If there are at most $0.99|S|$ vertices u in S with large estimated degree $\widetilde{d_{w''}}(u)$ (larger than $0.66|S|$), then we recurse on (C, w') ; the root of the tree created from this recursion becomes a child of A .
- Otherwise, we let R contain the vertices u whose estimated degree $\widetilde{d_{w''}}(u)$ is small (less than $0.65|S|$), and recurse on (R, w') . The root of the tree created from this recursion (let us call it A') becomes a child of A , and then we update $A \leftarrow A'$. Finally, we repeat the postprocessing again (but this time on $C \setminus R$ (instead of R) at level $\tilde{w}$ (instead of w)).

■ **Algorithm 2** $\ell_0(S, w)$.

```

1: Mark  $S$  at level  $w$  as a core cluster
2: if  $|S| \leq 1$  then return
3: obtain  $\mathcal{C} = \{C_1, \dots, C_k\}$  using an  $S$ -Structural Clustering subroutine on  $(V, E_{\tilde{w}})$ 
4: for all  $u, v$  in different clusters of  $\mathcal{C}$  do  $T(uv) \leftarrow w$ 
5: for all  $C' \in \mathcal{C}$  with  $|C'| \leq 0.99|S|$  do  $\ell_0(C', \tilde{w})$ 
6: if  $\exists C \in \mathcal{C}$  with  $|C| > 0.99|S|$  then
7:    $w' \leftarrow \tilde{w}, w'' \leftarrow \widetilde{w'}$ 
8:   while  $|C| > 0.99|S|$  and  $|\{u \in C \mid \widetilde{d_{w''}}(u) > 0.66|S|\}| > 0.99|S|$  do
9:      $R \leftarrow \{u \in C \mid \widetilde{d_{w''}}(u) < 0.65|S|\}$ 
10:    for all  $u \in R, v \in C \setminus R$  do
11:       $T(uv) \leftarrow w'$ 
12:     $\ell_0(R, w')$ 
13:     $C \leftarrow C \setminus R, w' \leftarrow \tilde{w'}, w'' \leftarrow \widetilde{w''}$ 
14:   $\ell_0(C, w')$ 

```

The idea of the postprocessing is that nodes whose degree drops significantly cannot be in a huge cluster without a big cost. The challenging part is showing that keeping the rest of the nodes in C is sufficient.

References

- 1 Richa Agarwala, Vineet Bafna, Martin Farach, Mike Paterson, and Mikkel Thorup. On the approximability of numerical taxonomy. *SIAM J. Comput.*, 28(3):1073–1085, 1999.
- 2 Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation clustering in data streams. *Algorithmica*, 83:1980–2017, 2021. doi:10.1007/S00453-021-00816-9.
- 3 Nir Ailon and Moses Charikar. Fitting tree metrics: Hierarchical clustering and phylogeny. *SIAM J. Comput.*, 40(5):1275–1291, 2011. Announced at FOCS 2005. doi:10.1137/100806886.
- 4 Federico Ardila. Subdominant matroid ultrametrics. *Annals of Combinatorics*, 8:379–389, 2005.
- 5 Sepehr Assadi, Vaggos Chatziafratis, Jakub Lacki, Vahab Mirrokni, and Chen Wang. Hierarchical clustering in graph streams: Single-pass algorithms and space lower bounds. In *COLT*, volume 178 of *Proceedings of Machine Learning Research*, pages 4643–4702. PMLR, 2022. URL: <https://proceedings.mlr.press/v178/assadi22a.html>.
- 6 Sepehr Assadi and Chen Wang. Sublinear time and space algorithms for correlation clustering via sparse-dense decompositions. In *ITCS*, volume 215 of *LIPICs*, pages 10:1–10:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ITCS.2022.10.
- 7 Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. In *43rd Symposium on Foundations of Computer Science (FOCS 2002)*, page 238. IEEE Computer Society, 2002. doi:10.1109/SFCS.2002.1181947.
- 8 Soheil Behnezhad, Moses Charikar, Vincent Cohen-Addad, Alma Ghafari, and Weiyun Ma. Fully dynamic correlation clustering: Breaking 3-approximation. *CoRR*, abs/2404.06797, 2024. doi:10.48550/arXiv.2404.06797.
- 9 Soheil Behnezhad, Moses Charikar, Weiyun Ma, and Li-Yang Tan. Almost 3-approximate correlation clustering in constant rounds. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 720–731, 2022. doi:10.1109/FOCS54457.2022.00074.
- 10 Soheil Behnezhad, Moses Charikar, Weiyun Ma, and Li-Yang Tan. Single-pass streaming algorithms for correlation clustering. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 819–849. SIAM, 2023. doi:10.1137/1.9781611977554.CH33.
- 11 Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Cliff Stein, and Madhu Sudan. Fully dynamic maximal independent set with polylogarithmic update time. In *FOCS*, pages 382–405. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00032.
- 12 Daniel Irving Bernstein. L-infinity optimization to bergman fans of matroids with an application to phylogenetics. *SIAM Journal on Discrete Mathematics*, 34(1):701–720, 2020. doi:10.1137/18M1218741.
- 13 Daniel Irving Bernstein and Colby Long. L-infinity optimization to linear spaces and phylogenetic trees. *SIAM Journal on Discrete Mathematics*, 31(2):875–889, 2017. doi:10.1137/16M1101027.
- 14 Ronald L Breiger, Scott A Boorman, and Phipps Arabie. An algorithm for clustering relational data with applications to social network analysis and comparison with multidimensional scaling. *Journal of Mathematical Psychology*, 12(3):328–383, 1975.
- 15 Mélanie Cambus, Fabian Kuhn, Etna Lindy, Shreyas Pai, and Jara Uitto. A $(3 + \epsilon)$ -approximate correlation clustering algorithm in dynamic streams. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2861–2880. SIAM, 2024. doi:10.1137/1.9781611977912.101.
- 16 Nairen Cao, Vincent Cohen-Addad, Euiwoong Lee, Shi Li, Alantha Newman, and Lukas Vogl. Understanding the cluster linear program for correlation clustering. In *STOC*, pages 1605–1616. ACM, 2024. doi:10.1145/3618260.3649749.
- 17 Amir Carmel, Debarati Das, Evangelos Kipouridis, and Evangelos Pipis. Fitting tree metrics and ultrametrics in data streams, 2025. arXiv:2504.17776.

- 18 L. L. Cavalli-Sforza and A. W. F. Edwards. Phylogenetic analysis models and estimation procedures. *The American Journal of Human Genetics*, 19:233–257, 1967.
- 19 Moses Charikar and Ruiquan Gao. Improved approximations for ultrametric violation distance. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1704–1737. SIAM, 2024. doi:10.1137/1.9781611977912.68.
- 20 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *J. Comput. Syst. Sci.*, 71(3):360–383, 2005. Announced at FOCS 2003. doi:10.1016/J.JCSS.2004.10.012.
- 21 Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. Near optimal LP rounding algorithm for correlationclustering on complete and complete k-partite graphs. In *STOC*, pages 219–228. ACM, 2015. doi:10.1145/2746539.2746604.
- 22 Victor Chepoi and Bernard Fichet. ℓ_∞ -approximation via subdominants. *Journal of mathematical psychology*, 44(4):600–616, 2000.
- 23 Flavio Chierichetti, Nilesch N. Dalvi, and Ravi Kumar. Correlation clustering in mapreduce. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*, pages 641–650. ACM, 2014. doi:10.1145/2623330.2623743.
- 24 Vincent Cohen-Addad, Debarati Das, Evangelos Kipouridis, Nikos Parotsidis, and Mikkel Thorup. Fitting distances by tree metrics minimizing the total error within a constant factor. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 468–479. IEEE, 2021. doi:10.1109/FOCS52979.2021.00054.
- 25 Vincent Cohen-Addad, Rémi de Joannis de Verclos, and Guillaume Lagarde. Improving ultrametrics embeddings through coresets. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 2060–2068. PMLR, 2021. URL: <http://proceedings.mlr.press/v139/cohen-addad21a.html>.
- 26 Vincent Cohen-Addad, Chenglin Fan, Euiwoong Lee, and Arnaud De Mesmay. Fitting metrics and ultrametrics with minimum disagreements. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 301–311. IEEE, 2022. doi:10.1109/FOCS54457.2022.00035.
- 27 Vincent Cohen-Addad, Karthik C. S., and Guillaume Lagarde. On efficient low distortion ultrametric embedding. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 2078–2088. PMLR, 2020. URL: <http://proceedings.mlr.press/v119/cohen-addad20a.html>.
- 28 Vincent Cohen-Addad, Silvio Lattanzi, Andreas Maggiori, and Nikos Parotsidis. Online and consistent correlation clustering. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 4157–4179. PMLR, 2022. URL: <https://proceedings.mlr.press/v162/cohen-addad22a.html>.
- 29 Vincent Cohen-Addad, Silvio Lattanzi, Slobodan Mitrovic, Ashkan Norouzi-Fard, Nikos Parotsidis, and Jakub Tarnawski. Correlation clustering in constant many parallel rounds. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 2069–2078, 2021. URL: <http://proceedings.mlr.press/v139/cohen-addad21b.html>.
- 30 Vincent Cohen-Addad, Euiwoong Lee, Shi Li, and Alantha Newman. Handling correlated rounding error via preclustering: A 1.73-approximation for correlation clustering. In *FOCS*, pages 1082–1104. IEEE, 2023. doi:10.1109/FOCS57990.2023.00065.
- 31 Vincent Cohen-Addad, Euiwoong Lee, and Alantha Newman. Correlation clustering with sherali-adams. In *FOCS*, pages 651–661. IEEE, 2022. doi:10.1109/FOCS54457.2022.00068.
- 32 Vincent Cohen-Addad, David Rasmussen Lolck, Marcin Pilipczuk, Mikkel Thorup, Shuyi Yan, and Hanwen Zhang. Combinatorial correlation clustering. In *STOC*, pages 1617–1628, 2024. doi:10.1145/3618260.3649712.
- 33 Mina Dalirrooyfard, Konstantin Makarychev, and Slobodan Mitrovic. Pruned pivot: Correlation clustering algorithm for dynamic, parallel, and local computation models. *CoRR*, abs/2402.15668, 2024. doi:10.48550/arXiv.2402.15668.

- 34 William H.E. Day. Computational complexity of inferring phylogenies from dissimilarity matrices. In *Bulletin of Mathematical Biology*, volume 49(4), pages 461–467, 1987.
- 35 Patrik D’haeseleer. How does gene expression clustering work? *Nature Biotechnology*, 23:1499–1501, 2005.
- 36 Andreas Dress, Barbara Holland, Katharina T Huber, Jack H Koolen, Vincent Moulton, and Jan Weyer-Menkhoﬀ. δ additive and δ ultra-additive maps, gromov’s trees, and the farris transform. *Discrete Applied Mathematics*, 146(1):51–73, 2005.
- 37 Geet Duggal, Rob Patro, Emre Sefer, Hao Wang, Darya Filippova, Samir Khuller, , and Carl Kingsford. Resolving spatial inconsistencies in chromosome conformation measurements. *Algorithms for Molecular Biology*, 8(1):1–10, 2013.
- 38 Chenglin Fan, Benjamin Raichel, and Gregory Van Buskirk. Metric violation distance: Hardness and approximation. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 196–209. SIAM, 2018. doi:10.1137/1.9781611975031.14.
- 39 Martin Farach, Sampath Kannan, and Tandy Warnow. A robust model for finding optimal evolutionary trees. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 137–145, 1993. doi:10.1145/167088.167132.
- 40 Anna C. Gilbert, Albert Gu, Christopher Ré, Atri Rudra, and Mary Wootters. Sparse recovery for orthogonal polynomial transforms. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPICs*, pages 58:1–58:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ICALP.2020.58.
- 41 Anna C. Gilbert and Lalit Jain. If it ain’t broke, don’t fix it: Sparse metric repair. In *55th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2017, Monticello, IL, USA, October 3-6, 2017*, pages 612–619. IEEE, 2017. doi:10.1109/ALLERTON.2017.8262793.
- 42 Boulos Harb, Sampath Kannan, and Andrew McGregor. Approximating the best-fit tree under ℓ_p norms. In *APPROX-RANDOM*, pages 123–133, 2005. doi:10.1007/11538462_11.
- 43 Patrick K. Kimes, Yufeng Liu, David Neil Hayes, and James Stephen Marron. Statistical significance for hierarchical clustering. *Biometrics*, 73(3):811–821, 2017.
- 44 Evangelos Kipouridis. Fitting tree metrics with minimum disagreements. In *31st Annual European Symposium on Algorithms (ESA 2023)*, 2023.
- 45 Mirko Křivánek. The complexity of ultrametric partitions on graphs. *Information processing letters*, 27(5):265–270, 1988. doi:10.1016/0020-0190(88)90090-7.
- 46 Bipul Kumar, Arun Sharma, Sanket Vatavwala, and Prashant Kumar. Digital mediation in business-to-business marketing: A bibliometric analysis. *Industrial Marketing Management*, 85:126–140, 2020.
- 47 Pei Lee, Laks VS Lakshmanan, and Evangelos E Milios. Incremental cluster evolution tracking from highly dynamic network data. In *2014 IEEE 30th International Conference on Data Engineering*, pages 3–14. IEEE, 2014.
- 48 Sanghoon Lee and M.M. Crawford. Unsupervised multistage image classification using hierarchical clustering with a bayesian similarity measure. *IEEE Transactions on Image Processing*, 14(3):312–320, 2005. doi:10.1109/TIP.2004.841195.
- 49 Sebastian Lühr and Mihai Lazarescu. Incremental clustering of dynamic data streams using connectivity based representative points. *Data & knowledge engineering*, 68(1):1–27, 2009. doi:10.1016/J.DATK.2008.08.006.
- 50 Bin Ma, Lusheng Wang, and Louxin Zhang. Fitting distances by tree metrics with increment error. *Journal of combinatorial optimization*, 3:213–225, 1999. doi:10.1023/A:1009837726913.
- 51 Konstantin Makarychev and Sayak Chakrabarty. Single-pass pivot algorithm for correlation clustering. keep it simple! In *Advances in Neural Information Processing Systems*, volume 36, pages 6412–6421, 2023.

- 52 Pedro Pereira Rodrigues, Joao Gama, and Joao Pedroso. Hierarchical clustering of time-series data streams. *IEEE transactions on knowledge and data engineering*, 20(5):615–627, 2008. doi:10.1109/TKDE.2007.190727.
- 53 Peter H.A. Sneath and Robert R. Sokal. Numerical taxonomy. *Nature*, 193(4818):855–860, 1962.
- 54 Peter H.A. Sneath and Robert R. Sokal. Numerical taxonomy. the principles and practice of numerical classification. *Freeman*, 1963.