Multi Layer Peeling for Linear Arrangement and Hierarchical Clustering

Yossi Azar
School of Computer Science, Tel-Aviv University, Israel

Danny Vainstein
School of Computer Science, Tel-Aviv University, Israel

Abstract

We present a new multi-layer peeling technique to cluster points in a metric space. A well-known non-parametric objective is to embed the metric space into a simpler structured metric space such as a line (i.e., Linear Arrangement) or a binary tree (i.e., Hierarchical Clustering). Points which are close in the metric space should be mapped to close points/leaves in the line/tree; similarly, points which are far in the metric space should be far in the line or on the tree. In particular we consider the Maximum Linear Arrangement problem [20] and the Maximum Hierarchical Clustering problem [12] applied to metrics.

We design approximation schemes ($1 - \epsilon$ approximation for any constant $\epsilon > 0$) for these objectives. In particular this shows that by considering metrics one may significantly improve former approximations (0.5 for Max Linear Arrangement and 0.74 for Max Hierarchical Clustering). Our main technique, which is called multi-layer peeling, consists of recursively peeling off points which are far from the "core" of the metric space. The recursion ends once the core becomes a sufficiently densely weighted metric space (i.e. the average distance is at least a constant times the diameter) or once it becomes negligible with respect to its inner contribution to the objective. Interestingly, the algorithm in the Linear Arrangement case is much more involved than that in the Hierarchical Clustering case, and uses a significantly more delicate peeling.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Hierarchical clustering, Linear arrangements, Metric embeddings

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.13

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2305.01367`

Funding *Yossi Azar*: Research supported in part by the Israel Science Foundation (grant No. 2304/20).

1 Introduction

Unsupervised learning plays a major role in the field of machine learning. Arguably the most prominent type of unsupervised learning is done through clustering. Abstractly, in this setting we are given a set of data points with some notion of pairwise relations which is captured via a metric space (such that closer points are more similar). In order to better understand the data, the goal is to embed this space into a simpler structured space while preserving the original pairwise relationships. A prevalent solution in this domain is to build a flat clustering (or partition) of the data (e.g., by using the k-means algorithm). However, these types of solutions ultimately fail to capture all pairwise relations (e.g., intra-cluster relations). To overcome this difficulty, often the metric space is mapped to structures that may capture all pairwise relations - in our case into a Linear Arrangement (LA) or a Hierarchical Clustering (HC).

The idea of embedding spaces by using a Linear Arrangement or Hierarchical Clustering structure is not new. These types of solutions have been extensively used in practice (e.g., see [11, 30, 5, 6, 26]) and have also been extensively researched from a theoretical point of view (e.g., see [13, 12, 24, 10, 17, 20]). Notably, the Linear Arrangement type objectives were first considered by Hansen [19] who considered the embedding of graphs into 2-dimensional and higher planes. On the other hand, the study of Hierarchical Clustering type objectives was initiated by Dasgupta [13] - spurring a fruitful line of work resulting in many novel algorithms. In practice, more often than not, the data considered adheres to the triangle inequality (in particular guaranteeing that if point a is similar, equivalently close, to points b and c then so are b and c) and thus may be captured by a metric (e.g., see [9, 25, 26]). The first objective we consider is the Max Linear Arrangement objective.

▶ **Definition 1.** *Let $G = (V, w)$ denote a metric (specifically, w satisfies the triangle inequality) with $|V| = n$. In the* ***Max Linear Arrangement problem*** *our goal is to return a 1-1 mapping $y : V \to [n]$ so as to maximize $\sum_{i,j} w_{i,j} y_{i,j}$, where $y_{i,j} = |y_i - y_j|$.*

The second objective we consider is the Max Hierarchical Clustering objective.

▶ **Definition 2.** *Let $G = (V, w)$ denote a metric (specifically, w satisfies the triangle inequality). In the* ***Max Hierarchical Clustering problem*** *our goal is to return a binary HC tree T such that its leaves are in a 1-1 correspondence with V. Furthermore, we would like to return T so as to maximize $\sum_{i,j} w_{i,j}|T_{i,j}|$, where T_{ij} is the subtree rooted at the lowest-common-ancestor of the leaves i and j in the Hierarchical Clustering tree T and $|T_{i,j}|$ is the number of leaves in $T_{i,j}$.*

These objectives were first considered by Hassin and Rubinstein [20] and Cohen-Addad et al. [12] (respectively) with respect to the non-metric case. For these (non-metric) objectives the best known approximation ratios are 0.5 for the Linear Arrangement objective [20] and 0.74 for the Hierarchical Clustering objective [25]). The former was achieved by was achieved by bisecting the data points randomly and thereafter greedily arranging each set and the latter was achieved by approximating the Balanced Max-2-SAT problem.

As stated earlier, more often than not, the data considered in practical applications adheres to the triangle inequality. Therefore, our results' merits are two fold. First, we offer a generalized framework to tackle these types of embedding objectives. Second, our results show that by applying this natural assumption we may significantly improve former best known approximations (from 0.5 (LA) and 0.74 (HC) to $1 - \epsilon$ for any constant $\epsilon > 0$).

Our Results

We provide the following results.

- We design a general framework in order to tackle the embedding of metric spaces into simpler structured spaces (see Algorithm 1). We then concretely apply our framework to both the Linear Arrangement and Hierarchical Clustering settings. For an extended discussion see Our Techniques.
- We apply our framework to the Linear Arrangement case. In this case we prove that our applied algorithm (2) is an EPRAS (see Definition 8) - i.e., for any constant $\epsilon > 0$ it yields a $1 - \epsilon$ approximation.
- We apply our framework to the Hierarchical Clustering case. In this case we prove that our applied algorithm (4) is an EPRAS (see Definition 8) - i.e., for any constant $\epsilon > 0$ it yields a $1 - \epsilon$ approximation.

Our Techniques

Our generic multi-layer peeling approach appears in Algorithm 1. We begin by checking whether the metric space is sufficiently densely weighted (i.e., whether the average distance is at least a constant times the diameter, or equivalently the metric's weighted density (see Definition 3) is constant). If this is the case then we apply a specific algorithm that handles such instances. In the LA case we devise our own algorithm (see Algorithm 3). Algorithm 3 leverages the General Graph Partitioning algorithm of Goldreich et al. [18] in order to "guess" an optimal graph partition that induces an almost optimal linear arrangement. In the HC case we leverage the work of Vainstein et al. [31].

If, however, the metric is not sufficiently densely weighted, then we observe that it must contain a **core** - a subset of nodes containing almost all data points with a diameter significantly smaller than the original metric's. Our general algorithm then peels off data points *far* from the core (in the LA setting) or *not in* the core (in the HC setting). We then embed these peeled off points; by placing them on one of the extreme sides of the line (in the LA setting) or by arranging them in a ladder structure (in the HC case; see Definition 11). Thus, we are left with handling the core (in the HC setting) or the extended core (in the LA setting).

Once again we consider two cases - either the total weight within the (extended) core is small enough, in which case we embed the core arbitrarily. Otherwise, we recurse on the instance induced by these data points. We claim that in every recursion step the density of the (extended) core increases significantly until eventually the recursion ends either when the (extended) core is sufficiently densely weighted or the total weight within the (extended) core is small enough.

Our proof is based on several claims. First, we consider the metric's (extended) core compared to the peeled off layer. Since our algorithm embeds the two sets separately, we need to bound the resulting loss in objective value. We show that the weights within the peeled off layer contribute negligibly towards the objective while the weights between the peeled off layer and the (extended) core, contribute significantly. Hence, it makes sense then to peel off this layer in order to maximize the gain in objective value.

While the aforementioned is enough to bound the loss in a single recursion step, it is not enough. The number of recursion steps may not be constant which, in principle, may cause a blow up of the error. Nevertheless, we show that the error in each level is bounded by a geometric sequence and hence is dominated by the error of the deepest recursion step. Consequently, we manage to upper bound the total accumulated error by a constant that we may take to be as small as we wish.

While at large this describes our proof techniques, the algorithm and analysis of LA objective is a bit more nuanced as we will be considering 3 sets: the metric's core, the peeled off layer, and any remaining points which together with the core are labeled as the extended core. In this case, to be able to justify peeling off a layer, we must choose the layer more aggressively. Specifically, we define this layer as points that are sufficiently far from the core (rather than any point outside the core, as in the HC case). Fortunately, this defined layer (see Algorithm 2) fits our criteria (of our general algorithm, Algorithm 1).

Related Work

While the concept of hierarchical clustering has been around for a long time, the HC objective is relatively recent. In their seminal work, Dasgupta [13] considered the problem of HC from an optimization view point. Thereafter, Cohen-Addad et al. [12] were the first to consider the objective we use in our manuscript. In their work they showed that the well known Average-Linkage algorithm yields an approximation of $\frac{2}{3}$. Subsequently, Charikar et al. [8] improved upon this result through the use of semidefinite programming - resulting in a 0.6671 approximation. Finally, Naumov et al. [25] improved this to 0.74 by approximating the Balanced Max-2-SAT problem. With respect to the Max LA objective, Hassin and Rubinstein [20] were first to consider the problem. Through an approach of bisection and then greedily arranging the points, Hassin and Rubinstein managed to achieve a 0.5 approximation. We note that the previous mentioned results all hold for arbitrary weights, while our main contribution is showing that by assuming the triangle inequality (i.e., metric-based dissimilarity weights) we may achieve PTAS's for both objectives. We further note that with respect to metric-based dissimilarity weights, specifically an L1 metric, Rajagopalan et al. [26] proved a 0.9 approximation through the use of random cut trees.

Both objectives have been originally studied with respect to their minimization variants. The minimum LA setting was first considered by Hansen [19]. Hansen leveraged the work of Leighton and Rao [23] on balanced separators in order to approximate the minimum linear arrangement objective to facor of $O(\log^2 n)$. Following several works improving upon this result, both Charikar et al. [10] and Feige and Lee [17] leveraged the novel work of Arora et al. [4] on rounding of semidefinite programs, and combined this with the rounding algorithm of Rao and Reicha [27] in order to show a $O(\sqrt{\log n}\log\log n)$ approximation. For further reading on these are related types of objectives see [16, 27, 29, 28]. On the other hand, as mentioned earlier the minimum HC setting was introduced by Dasgupta [13] and extensively studied as well (e.g., see [13, 12, 7, 8, 1, 2, 31]).

Most related to our work is that of de la Vega and Kenyon [15]. In their work they provide a PTAS for the Max Cut problem given a metric. The algorithm works by first creating a graph of clones (wherein each original vertex is cloned a number of times that is based on its outgoing weight in the original metric) with the property of being dense. It thereafter solves the problem in this new graph by applying the algorithm of de la Vega and Karpinski [14]. For our objectives (HC and LA) such an approach seems to fail - specifically due to the fact that our objectives take into consideration the number of nodes in every induced cut and the cloned graph inflates the number of nodes which in turn inflates our objective values. Thus, for our considered types of objectives we need the more intricate process of iterative peeling (and subsequently terminating the process with more suited algorithms that leverage the General Graph Partitioning algorithm of Goldreich et al. [18]). It is worth while mentioning that there has also been an extensive study of closely related objectives with respect to dense instances (e.g. see [22, 3, 21]). However these types of approaches seem to fall short since our considered metrics need not be dense.

2 Multi-Layer Peeling Framework

Before defining our algorithms we need the following definitions.

▶ **Definition 3.** *Let $G = (V, w)$ denote a metric and $U \subset V$ denote a subset of its nodes. We introduce the following notations: (1) let $\mathbf{D_U} = \max_{i,j\in U} w_{i,j}$ denote U's **diameter**, (2) let $\mathbf{W_U} = \sum_{i,j\in U} w_{i,j}$ denote U's **sum of weights**, (3) let $\mathbf{n_U} = |U|$ denote U's **size** and (4) let $\rho_\mathbf{U} = \frac{W_U}{n_U^2 D_U}$ denote U's **weighted density**[1].*

[1] Typically the density is defined with respect to $\binom{n}{2}$. For ease of presentation, we chose to define it with respect to n^2 - the proofs remain the same using the former definition.

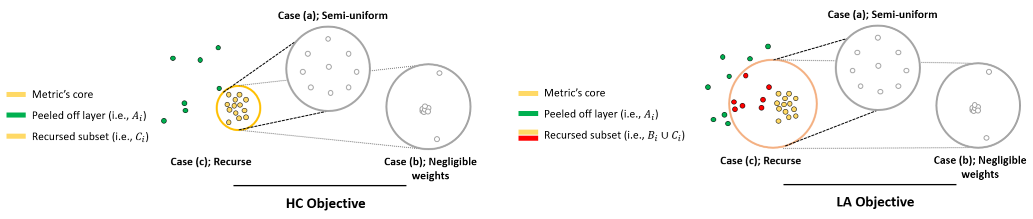
Case (a); Semi-uniform
Metric's core
Peeled off layer (i.e., A_i)
Recursed subset (i.e., C_i)
Case (c); Recurse
Case (b); Negligible weights
HC Objective
Case (a); Semi-uniform
Metric's core
Peeled off layer (i.e., A_i)
Recursed subset (i.e., $B_i \cup C_i$)
Case (c); Recurse
Case (b); Negligible weights
LA Objective

Figure 1 A recursion step (case (c)) and the two possible halting steps (cases (a) and (b)). The yellow points define the metric's core. In the HC case we peel off both red and green points in a single step, while in the LA we must be more delicate and only peel off the green points.

All our algorithms will make use of the following simple yet useful structural lemma that states that for small-density instances there exists a large cluster of nodes with a small diameter. The proof appears in the full version.

▶ **Lemma 4.** *For any metric $G = (V, w)$ there exists a set $U \subset V$ such that $D_U \leq 4D_V\sqrt{\rho_V}$ and $n_U \geq n_V(1 - \sqrt{\rho_V})$.*

▶ **Definition 5.** *Given a metric $G = (V, w)$ we denote $U \subset V$ as guaranteed by Lemma 4 as a metric's* ***core****.*

Note that the core can be found algorithmicaly simply through brute force (while the core need not be unique, our algorithms will choose one arbitrarily).

Throughout our paper we consider different metric-based objectives. In order to solve them, we apply the same recipe - if the instance is sufficiently densely weighted, apply an algorithm for these types of instances. Otherwise, the algorithm detects the metric's core (which is a small-diameter subset containing almost all nodes) and peel off (and subsequently embed) a layer of data points that are far from the core. The algorithm then considers the core; if it is sufficiently small (in terms of inner weights) then we embed the core arbitrarily and halt. Otherwise, we recurse on the core. Our algorithms for both objectives (LA and HC) will follow the same structure as defined in Algorithm 1.

Algorithm 1 General Algorithm.

```
if the instance is sufficiently densely weighted then                    // case (a)
    Solve it using $ALG_{d-w}$.
else
    Let $C$ denote the metric's core (as defined by Definition 5).
    Define the layer to peel off $A \subset V \setminus C$ appropriately.
    Embed $A$.
    if $W_{V \setminus A}$ is negligible then  Embed $V \setminus A$ arbitrarily and return. ; // case (b)
    else  Continue recursively on $V \setminus A$ ;                        // case (c)
```

We denote by cases (a) and (b) the different cases for which the algorithm may terminate and by case (c) the recursive step. We further denote by ALG_{d-w} an auxiliary algorithm that will handle sufficiently densely weighted instances. (These algorithms will differ according to the different objectives).

Henceforth, given an algorithm ALG and metric G we denote by $ALG(G)$ the algorithm's returned embedding. We note that when clear from context we overload the notation and denote $ALG(G)$ as the embedding's value under the respective objectives. Equivalently, we will use the term $OPT(G)$ for the optimal embedding.

Our different algorithms will be similarly defined and thus so will their analyses. Thus, we introduce a general scheme for analyzing such algorithms. Let k denote the number of recursive calls our algorithm performs. Furthermore, let G_i denote the instance the algorithm is called upon in step i for $i = 0, 1, \dots, k$. (I.e., $G = G_0$ and $ALG(G_k)$ does not perform a recursive step, meaning that it terminates with case (a) or (b)). We first observe that by applying a simple averaging argument we get the following useful observation.

▶ **Observation 6.** *If there exist $\alpha_i, \beta_i, \gamma_i > 0$ such that $ALG(G_i) \geq \alpha_i + ALG(G_{i+1})$ and $OPT(G_i) \leq \beta_i + \gamma_i OPT(G_{i+1})$ for all $i = 0, \dots, k-1$ then*

$$\frac{ALG(G)}{OPT(G)} \geq \frac{\sum_{i=0}^{k-1} \alpha_i + ALG(G_k)}{\sum_{i=0}^{k-1} (\beta_i \Pi_{j=0}^{i-1} \gamma_j) + (\Pi_{i=0}^{k-1} \gamma_i) OPT(G_k)}$$
$$\geq \min\{\min_i \{\frac{\alpha_i}{\beta_i \Pi_{j=0}^{i-1} \gamma_j}\}, \frac{ALG(G_k)}{(\Pi_{i=0}^{k-1} \gamma_i) OPT(G_k)}\}.$$

Thus, in order to analyze a given algorithm, it will be enough to set the values of α_i, β_i and γ_i, and further analyze the approximation ratio of $\frac{ALG(G_k)}{OPT(G_k)}$ for the different terminating cases (cases (a) and (b)).

3 Notations and Preliminaries

We introduce the following notation to ease our presentation later on.

▶ **Definition 7.** *Given a metric $G = (V, w)$, a solution $\mathbf{SOL(G)}$ for the LA objective and disjoints sets $A, B \subset V$ we define: $\mathbf{SOL(G)|_A} = \sum_{i,j \in A} w_{i,j} y_{i,j}$ and $\mathbf{SOL(G)|_{A,B}} = \sum_{i \in A, j \in B} w_{i,j} y_{i,j}$. For the HC objective the notations are defined symmetrically by replacing $y_{i,j}$ with $|T_{i,j}|$.*

We will make use of algorithms belonging to the following class of algorithms.

▶ **Definition 8.** *An algorithm is considered an Efficient Polytime Randomized Approximation Scheme (EPRAS) if for any $\epsilon > 0$ the algorithm has expected running time of $f(\frac{1}{\epsilon}) n^{O(1)}$ and approximates the optimal solution's value up to a factor of $1 - \epsilon$.*

We will frequently use the following (simple) observations and thus we state them here.

▶ **Observation 9.** *Given values $\alpha_i \geq 0$, $\alpha \in (0, \frac{1}{k(k+1)})$ and $k \in \mathbb{N}$ we have: **(1)** $\Pi_i (1 - \alpha_i) \geq 1 - \sum_i \alpha_i$, **(2)** $1 + k\alpha < \frac{1}{1 - k\alpha} < 1 + (k+1)\alpha$ and **(3)** $1 + k\alpha < e^{k\alpha} < 1 + (k+1)\alpha$.*

The following facts will prove useful in our subsequent proofs and are therefore stated here.

▶ **Fact 10.** *Given a metric G, if the optimal linear arrangement under the LA objective is $OPT_{LA}(G)$ and the optimal hierarchical clustering under the HC objective is $OPT_{HC}(G)$ then we have $OPT_{LA}(G) \geq \frac{1}{3} n \sum_{i,j} w_{i,j} y_{i,j}$ and $OPT_{HC}(G) \geq \frac{2}{3} n \sum_{i,j} w_{i,j} |T_{i,j}|$.*

We note that the HC portion of Fact 10 has been used widely in the literature (e.g., see proof in [12]). The LA portion of Fact 10 is mentioned in Hassin and Rubinstein [20]. Finally, in the HC section we make use of "ladder" HC trees. We define them here.

▶ **Definition 11.** *We define a "ladder" as an HC tree that cuts a single data point from the rest at every cut (or internal node).*

4 The Linear Arrangement Objective

We will outline the section as follows. We begin by presenting our algorithms (first the algorithm that handles case (a) and thereafter the general algorithm). We will then bound the algorithm's approximation guarantee (by following the bounding scheme of Observation 6). Finally, we will analyze the algorithm's running time.

4.1 Defining the Algorithms

Here we begin by applying our general algorithm to the linear arrangement problem (which we will denote simply as ALG). The algorithm uses, as a subroutine, an algorithm to handle case (a). We denote this subroutine as ALG_{d-w} and define it following the definition of ALG.

4.1.1 Defining ALG

Here we apply our general algorithm (Algorithm 1) to the linear arrangement setting. In order to do so, roughly speaking, we define the layer to peel off A as the set of all points which are "far" from the metric's core. We also introduce a subroutine to handle densely weighted instances, ALG_{d-w}.

Algorithm 2 Linear Arrangement Algorithm (ALG).

```
if $\rho \geq \epsilon^6$ then solve it using $ALG_{d-w}$. ;                       // case (a)
else
    Let $C$ denote the metric's core (as defined by Lemma 4).
    Let $A$ denote all data points that are of distance $\geq \epsilon^2 D_V$ from $C$.
    Place $A$ to the left of $V \setminus A$. Arrange $A$ arbitrarily.
    if $W_{V \setminus A} < \epsilon W_V$ then Arrange $V \setminus A$ arbitrarily and return. ;   // case (b)
    else Continue recursively on $V \setminus A$. ;                                     // case (c)
```

The set $V \setminus \{A \cup C\}$ will be used frequently in the upcoming proofs and thus we give it its own notation.

▶ **Definition 12.** *Denote $B = V \setminus \{A \cup C\}$ where A and C are defined as in Algorithm 2.*

4.1.2 Defining ALG_{d-w}

Here we will introduce an algorithm to handle case (a) type instances. Before formally defining the algorithm, we will first provide some intuition. Towards that end we first introduce the following definition.

▶ **Definition 13.** *Consider $OPT(G_k)$'s embedding into the line, $[n]$. Partition $[n]$ into $\frac{1}{\epsilon}$ consecutive sets each of size ϵn and let P_i^* denote the points embedded by $OPT(G_k)$ into the i'th consecutive set. Furthermore, denote by $P^* = \{P_i^*\}$ the induced partition of the metric.*

Later on, we will show that $OPT(G_k)$'s objective value is closely approximated by the value generated solely from inter-partition-set edges (i.e., any (u, v) where u, v lie in different partition sets of P^*). While $OPT(G_k)$ cannot be found algorithmically, assuming the above holds, it is enough for ALG_{d-w} to guess the partition P^*. Indeed, that is exactly what we will do, by using the general graph partitioning algorithm of Goldreich et al. [18].

We denote the General Graph Partitioning algorithm of Goldreich et al. [18] as $PT(G, \Phi, \epsilon_{err})$. See Definition 21 for a definition of Φ and ϵ_{err} (these will be defined by ALG_{d-w} as well) and see Theorem 22 for the tester's guarantees. We are now ready to define our algorithm that handles sufficiently densely weighted instances (Algorithm 3).

Algorithm 3 LA Algorithm for Sufficiently Densely Weighted Instances (ALG_{d-w}).

```
Let $k = \frac{1}{\epsilon}$ denote the size of the partition.
for $\{\mu_{j,j'}\}_{j \le k, j' \le k, j \ne j'} \subset \{i\epsilon^9 n^2 D_V : i \in \mathbb{N} \wedge i \le \frac{1}{\epsilon^7}\}$ do
    Let $\Phi = \{\epsilon n, \epsilon n\}_{j=1}^{k} \cup \{\mu_{j,j'}, \mu_{j,j'}\}_{j,j'=1}^{k}$.
    Run $PT(G, \Phi, \epsilon_{err} = \epsilon^9)$. Let $P$ denote the output partition (if succeeded).
    Let $\widehat{y}$ denote the linear arrangement obtained from embedding $P$ consecutively on
      the line (and arbitrarily within the partition sets).
    Compute the value $\sum_e w_e \widehat{y}_e$ for $P$.
Return the partition with maximum $\sum_e w_e \widehat{y}_e$ value.
```

4.2 Analyzing the Approximation Ratio of *ALG*

Now that we have defined ALG we are ready to analyze its approximation ratio. Recall that by Observation 6 it is enough to analyze the approximation ratio of cases (a), (b) and the total loss incurred by the recursion steps (i.e., by setting α_i, β_i and γ_i).

4.2.1 Structural Lemmas

Recall that we defined k to be the number of recursion steps used by ALG and that G_i is the instance that ALG is applied to at recursion step i. Further recall that given G_i, $ALG(G_i)$ partitioned the instance into A_i, B_i and C_i and that, informally, by Lemma 4 n_{C_i} contains the majority of the data points and D_{C_i} is relatively small compared to D_{V_i}.

By the definition of C_i, A_i could be considered as a set of outliers. Therefore, intuitively it makes sense to split A_i from C_i. In order to prove our algorithm's approximation ratio we will show that in fact one does not lose too much compared to optimal solution, by splitting A_i from C_i. In order to do so we will show that in fact, both the values of ALG and OPT will be roughly equal to $\frac{1}{2}nW_{A_i,C_i}$ (which makes sense intuitively since C_i is of low diameter and contains many points and A_i are the points that are far from this cluster).

The following lemmas consider 2 types of algorithms - algorithms that split A_i and C_i and algorithms that do not. Furthermore, they show that in fact, by the structural properties of A_i and C_i, if we consider the values generated by these 2 types of algorithms restricted to the objective value generated by the inter-weights W_{A_i,C_i}, are approximately equal. We begin by lower bounding the value generated by algorithms that split A_i and C_i. Due to lack of space, we defer the following proofs to the full version.

▶ **Lemma 14.** *Given the two disjoint sets C_i and A_i and a linear arrangement y that places all nodes in A_i to the left of all nodes in C_i we are guaranteed that*

$$\sum_{c \in C_i, a \in A_i} w_{a,c} y_{a,c} \ge \frac{n_{C_i}}{2}(W_{C_i,A_i} - n_{C_i} n_{A_i} D_{C_i}).$$

Due to the fact that C_i is a small cluster containing most of the data points the above lemma reduces to the following corollary.

▶ **Corollary 15.** *Given any linear arrangement y that places all nodes in A_i to the left of all nodes in C_i we are guaranteed that*

$$\sum_{a\in A_i,c\in C_i} w_{a,c}y_{a,c} \geq \frac{1}{2}nW_{A_i,C_i}(1-\frac{5\sqrt{\rho}}{\epsilon^2})$$

Now that we have lower bounded algorithms that split A_i and C_i we will upper bound algorithms that do not have this restriction. (Note that we begin by handling the case where one of the disjoint sets is a single data point and thereafter generalize it to two disjoint sets).

▶ **Lemma 16.** *Given a set C_i and a point $p \notin C_i$, we are guaranteed that*

$$\sum_{c\in C_i} w_{p,c}y_{p,c} \leq (W_{p,C_i}+n_{C_i}D_{C_i})(n-\frac{n_{C_i}}{2}).$$

We are now ready to upper bound the inter-objective-value of two sets of disjoint points.

▶ **Lemma 17.** *Given the two disjoint sets C_i and A_i and any linear arrangement y we are guaranteed that*

$$\sum_{c\in C_i,a\in A_i} w_{a,c}y_{a,c} \leq (n-\frac{n_{C_i}}{2})(W_{C_i,A_i}+n_{C_i}n_{A_i}D_{C_i}).$$

Due to the fact that C_i is a small cluster containing most of the data points the lemma reduces to the following corollary.

▶ **Corollary 18.** *Given any linear arrangement y we are guaranteed that*

$$\sum_{a\in A_i,c\in C_i} w_{a,c}y_{a,c} \leq \frac{1}{2}nW_{A_i,C_i}(1+\frac{9\sqrt{\rho}}{\epsilon^2}).$$

We will want to show that the objective values of both ALG and OPT (and some other intermediate values that will be defined later on) are approximately determined by their value on the inter-weights of W_{A_i,C_i}. In order to do so, we first introduce the following structural lemma that will help us explain this behaviour.

▶ **Lemma 19.** *Given an instance G and sets A, B and C as defined by $ALG(G)$ we have $W_A + W_{A,B} \leq 2\frac{\sqrt{\rho}}{\epsilon^2}W_{A,C}$.*

4.2.2 Analyzing the Approximation Ratio of Case (a) of ALG

We first give an overview the approximation ratio analysis. Recall the definition of P^* (Definition 13). The first step towards our proof, is to show that instead of trying to approximate $OPT(G_k)$, it will be enough to consider its value restricted to intra-partition-set weights with respect to P^*. Even more, for such weights $w_{u,v}$, incident to P_i^* and P_{i+j}^*, it will be enough to assume that their generated value towards the objective (i.e., the value $y_{u,v}$) is only $(j-1)\epsilon n$ (while it may be as large as $(j+1)\epsilon n$). Formally, this will be done in Lemma 20 (whose proof is deferred to the full version).

Next, recall that ALG_{d-w} tries to guess the partition P^* (up to some additive error) and let P denote the partition guessed by ALG_{d-w}. Observe that if guessed correctly, the value generated towards ALG's objective for any intra-partition-set weight crossing between P_i and P_{i+j} is at least $|P_{i+1}| + \cdots |P_{i+j-1}|$ and if we managed to guess the set sizes as well then this value is exactly $(j-1)\epsilon n$ (equivalent to that of OPT's). This will be done in Proposition 23.

▶ **Lemma 20.** *Given the balanced line partition of set sizes ϵn, denoted as P^*, we have*

$$OPT(G_k) \leq (1+13\epsilon) \sum_{\substack{1\leq i\leq k-1 \\ 1\leq j\leq k-i}} W_{P_i^*,P_{i+j}^*}(|P_{i+1}^*| + \cdots + |P_{i+j-1}^*|).$$

Before proving Proposition 23 we state the properties of the general graph partitioning algorithm of Goldreich et al. [18].

▶ **Definition 21** ([18])**.** *Let $\Phi = \{\lambda_j^{LB}, \lambda_j^{UB}\}_{j=1}^k \cup \{\mu_{j,j'}^{LB}, \mu_{j,j'}^{UB}\}_{j,j'=1}^k$ denote a set of non-negative values such that $\lambda_j^{LB} \leq \lambda_j^{UB}$ and $\mu_{j,j'}^{LB} \leq \mu_{j,j'}^{UB}$. We define $\mathcal{GP}_\Phi$ the set of graphs G on n vertices that have a k partition $(V_1,\ldots,V_k)$ upholding the following constraints*

$$\forall j:\ \lambda_j^{LB} \leq \frac{|V_j|}{n} \leq \lambda_j^{UB}; \quad \forall j,j':\ \mu_{j,j'}^{LB} \leq \frac{W_{V_j,V_{j'}}}{n^2} \leq \mu_{j,j'}^{UB}.$$

▶ **Theorem 22** ([18])**.** *Given inputs $G=(V,w)$ with $|V|=n$ and $w: V\times V \to [0,1]$ describing the graph and Φ describing bounds on the wanted partition, ϵ_{err}, the algorithm $PT(G,\Phi,\epsilon_{err})$ has expected running time[2] of*

$$\exp\big(\log(\frac{1}{\epsilon_{err}}) \cdot (\frac{O(1)}{\epsilon_{err}})^{k+1}\big) + O(\frac{\log\frac{k}{\epsilon_{err}}}{\epsilon_{err}^2}) \cdot n.$$

Furthermore, if $G \in \mathcal{GP}_\Phi$ as in Definition 21 then the algorithm outputs a partition satisfying

- $\forall j:\ \lambda_j^{LB} - \epsilon_{err} \leq \frac{|V_j|}{n} \leq \lambda_j^{UB} + \epsilon_{err}$,
- $\forall j,j':\ \mu_{j,j'}^{LB} - \epsilon_{err} \leq \frac{W_{V_j,V_{j'}}}{n^2} \leq \mu_{j,j'}^{UB} + \epsilon_{err}$.

We are now ready to prove Proposition 23.

▶ **Proposition 23.** *If ALG terminates in case (a) then $\frac{ALG_{d-w}(G_k)}{OPT(G_k)} = \frac{ALG(G_k)}{OPT(G_k)} \geq 1-20\epsilon$.*

Proof. Let $P=\{P_i\}$ denote the partition returned by $PT(G_k,\Phi,\epsilon_{err})$ and recall that its number of sets is $k=\frac{1}{\epsilon}$ and that $\epsilon_{err}=\epsilon^9$. We first observe that by Theorem 22 we are guaranteed that the error in $|P_i|$ compared to $|P_i^*| = \epsilon n$ is at most $|P_i| \geq \epsilon n - \epsilon_{err} n$ (due to the fact that in Φ we requested sets of size exactly ϵn). Therefore

$$ALG_{d-w} \geq \sum_{\substack{1\leq i\leq k-1 \\ 1\leq j\leq k-i}} W_{P_i,P_{i+j}}(|P_{i+1}| + \cdots + |P_{i+j-1}|) \geq \sum_{\substack{1\leq i\leq k-1 \\ 1\leq j\leq k-i}} (j-1)(\epsilon n - \epsilon_{err} n) W_{P_i,P_{i+j}}, \tag{1}$$

where $W_{P_i,P_{i+j}}$ denotes the weight crossing between P_i and P_{i+j}. For ease of presentation we will remove the subscript in the summation henceforth.

Consider the difference between the cut size of $W_{P_i,P_{i+j}}$ and $W_{P_i^*,P_{i+j}^*}$. Their difference originates from two errors: (1) the error that incurred by the PT algorithm (see Theorem 22) and (2) the error ALG_{d-w} incurred in order to guess the partition of $OPT(G_k)$ (see Algorithm 3). Therefore,

$$W_{P_i,P_{i+j}} \geq W_{P_i^*,P_{i+j}^*} - \epsilon_{err} n^2 D_V - \epsilon^9 n^2 D_V = W_{P_i^*,P_{i+j}^*} - 2\epsilon^9 n^2 D_V$$

where the last equality is since $\epsilon_{err} = \epsilon^9$. Combining this with inequality 1 yields

$$\begin{aligned} ALG_{d-w} \geq (\epsilon n - \epsilon_{err} n) \cdot \sum (j-1) W_{P_i^*,P_{i+j}^*} - (\epsilon n - \epsilon_{err} n) \cdot 2(\epsilon^9 n^2) D_V \sum (j-1) \geq \\ (\epsilon n - \epsilon_{err} n) \cdot \sum (j-1) W_{P_i^*,P_{i+j}^*} - 2n^3\epsilon^7 D_V, \end{aligned} \tag{2}$$

[2] We remark that the original algorithm contains a probability of error δ, that appears in the running time. We disregard this error and bound the expected running time of the algorithm.

where the last inequality follows since $\epsilon_{err} > 0$ and $\sum(j-1) = \sum_{i=1}^{k}\sum_{j=i+1}^{k}(j-1) \leq k^3 = \epsilon^{-3}$.

Due to the fact that we are in case (a) we have that $\frac{W}{n^2 D_V} = \rho \geq \epsilon^6$. By Fact 10 we have that $OPT \geq \frac{1}{3}nW$ and therefore $2n^3\epsilon^7 D_V$ can be bounded by $2n^3\epsilon^7 D_V \leq 2\epsilon nW \leq 6\epsilon OPT$. Thus we get $2n^3\epsilon^7 D_V \leq 6\epsilon OPT(G_k)$. Combining this with inequality 2 yields

$$ALG_{d-w} \geq (\epsilon n - \epsilon_{err} n) \cdot \sum(j-1)W_{P_i^*,P_{i+j}^*} - 6\epsilon OPT(G_k). \tag{3}$$

On the other hand, recall that P^* denotes the balanced partition where all sets are of size ϵn. Therefore, by Lemma 20 we therefore get

$$\begin{aligned} OPT(G_k) \leq (1+13\epsilon)\sum W_{P_i^*,P_{i+j}^*}(|P_{i+1}^*| + \cdots + |P_{i+j-1}^*|) = \\ (1+13\epsilon)\sum(j-1)(\epsilon n)W_{P_i^*,P_{i+j}^*} = \epsilon n(1+13\epsilon)\cdot\sum(j-1)W_{P_i^*,P_{i+j}^*}. \end{aligned} \tag{4}$$

Combining inequalities 3 and 4 yields

$$\begin{aligned} ALG_{d-w} \geq \frac{\epsilon n - \epsilon_{err} n}{\epsilon n(1+13\epsilon)} OPT(G_k) - 6\epsilon OPT(G_k) = \\ \frac{1-\epsilon^8}{1+13\epsilon} OPT(G_k) - 6\epsilon OPT(G_k) \geq (1-20\epsilon) OPT(G_k), \end{aligned}$$

thereby concluding the proof. ◀

4.2.3 Analyzing the Approximation Ratio of Case (b) of ALG

Using our structural lemmas we will analyze the approximation ratio of ALG applied to G_k under the assumption that the algorithm terminated in case (b) (i.e., that $\rho < \epsilon^6$ and $W_{B\cup C} \leq \epsilon W_{G_k}$). The full proof is deferred to the full version.

▶ **Proposition 24.** *If ALG terminates in case (b) then $\frac{ALG(G_k)}{OPT(G_k)} \geq 1 - 33\epsilon$.*

Sketch. The proof follows the following path. Due to the fact that most of the instance's density is centered at the metric's core C, the majority of $OPT(G_k)$'s objective is derived from weights incident to C. Since we are case (b), the weight of $W_{B\cup C}$ is negligible and therefore we will show that in fact $OPT(G_k)$'s objective is defined by $OPT(G_k)|_{A,C}$. Thereafter, we show that in fact the best strategy to optimize for weights in $W_{A,C}$ is to place A at one extreme of the line and C at the other - which, fortunately, is what $ALG(G_k)$ (approximately) does - thereby approximating $OPT(G_k)$. ◀

4.2.4 Setting the Values α_i, β_i and γ_i

Due to lack of space, the following proofs are deferred to the full version.

▶ **Proposition 25.** *For A_i and C_i as defined by our algorithm applied to G_i and for $\alpha_i = \frac{1}{2}nW_{A,C}(1 - \frac{5\sqrt{\rho}}{\epsilon^2})$, we have $ALG(G_i) \geq \alpha_i + ALG(G_{i+1})$.*

▶ **Proposition 26.** *Let $G_i = (V_i, w_i)$ and $G_{i+1} = (V_{i+1}, w_{i+1})$ denote the instances defined by the i and $i+1$ recursion steps. Furthermore let $\beta_i = \frac{1}{2}n_{V_i}W_{A_i,C_i}(1 + \frac{13\sqrt{\rho}}{\epsilon^2})$ and $\gamma_i = 1 + 4\sqrt{\rho_i}$. Therefore, $OPT(G_i) \leq \beta_i + \gamma_i OPT(G_{i+1})$.*

Thus, we have managed to set the values of α_i, β_i and γ_i as follows.

▶ **Definition 27.** *We define the values α_i, β_i and γ_i as follows*

$$\alpha_i = \frac{1}{2}nW_{A_i,C_i}(1 - \frac{5\sqrt{\rho_i}}{\epsilon^2}); \quad \beta_i = \frac{1}{2}n_{V_i}W_{A_i,C_i}(1 + \frac{13\sqrt{\rho_i}}{\epsilon^2}); \quad \gamma_i = 1 + 4\sqrt{\rho_i}. \tag{5}$$

4.2.5 Putting it all Together

Now that we have analyzed the terminal cases of the algorithm (cases (a) and (b)) and that we have set the values of α_i, β_i and γ_i we will to combine these results to prove ALG's approximation ratio (as in Observation 9). In order to so we must therefore bound the values $\min_i\{\frac{\alpha_i}{\beta_i \Pi_{j=0}^{i-1}\gamma_j}\}$ and $\frac{ALG(G_k)}{(\Pi_{i=0}^{k-1}\gamma_i)OPT(G_k)}$. However, before doing so we will first show that $\Pi_{j=0}^{i-1}\gamma_j$ converges. Recall that $\gamma_i = 1+4\sqrt{\rho_i}$. The following lemma shows that the instances' densities (ρ_i) increase at a fast enough rate (exponentially) in order for $\Pi_{j=0}^{i-1}\gamma_j$ to converge.

▶ **Lemma 28.** *For all $i = 1, \ldots, k-1$ we are guaranteed that $\rho_{i+1} \geq 4\rho_i$.*

Proof. Let V denote the set of nodes of G_i. Recall the notations A, B and C defined by our algorithm applied to V (in particular, the set of nodes of G_{i+1} is exactly $B \cup C$). Therefore, if we denote by D_{B-C} the largest distance between any point in B and its closest point in C, then $D_{B\cup C} \leq 2D_{B-C} + D_C \leq 2\epsilon^2 D_V + 4D_V\sqrt{\rho_i}$, where the first inequality follow from the triangle inequality and the second follows due to the fact that B is defined as the set of all points of distance at most ϵ^2 from C. Therefore,

$$\rho_{i+1} = \frac{W_{B\cup C}}{n_{B\cup C}^2 \cdot D_{B\cup C}} \geq \frac{W_V}{n_V^2 \cdot D_V}\left(\frac{\epsilon}{2\epsilon^2+4\sqrt{\rho_i}}\right) = \rho_i\left(\frac{\epsilon}{2\epsilon^2+4\sqrt{\rho_i}}\right), \tag{6}$$

where the equalities follows by the definition of ρ_i and the inequality follows due to the fact that $W_{B\cup C} \geq \epsilon W_V$ (which follows due to the fact that we are in case (c)), $n_{B\cup C} \leq n_V$ and $D_{B\cup C} \leq (2\epsilon^2 + 4\sqrt{\rho_i})D_V$ (as stated above). Since we are in case (c), we are guaranteed that $\rho_i \leq \epsilon^6$ and therefore

$$\frac{\epsilon}{2\epsilon^2+4\sqrt{\rho_i}} \geq \frac{\epsilon}{2\epsilon^2+4\epsilon^3} \geq \frac{1}{3\epsilon}, \tag{7}$$

since $\epsilon \leq 10^{-2}$. Combining inequalities 6 and 7, and since $\epsilon < 10^{-2}$ yields $\rho_{i+1} \geq \rho_i(\frac{\epsilon}{2\epsilon^2+4\sqrt{\rho_i}}) \geq \frac{\rho_i}{3\epsilon} \geq 4\rho_i$, thereby concluding the proof. ◀

We are now ready to show that $\Pi_{j=0}^{i-1}\gamma_j$ converges.

▶ **Lemma 29.** *For $\gamma_i = 1+4\sqrt{\rho_i}$ we have $\Pi_{j=0}^{i-1}\gamma_j \leq 1+5\sqrt{\rho_i}$.*

Proof. Observe that $\Pi_{j=0}^{i-1}(1+4\sqrt{\rho_j}) \leq e^{4\cdot\sum_j\sqrt{\rho_j}} \leq e^{4\sqrt{\rho_i}} \leq 1+5\sqrt{\rho_i}$, where the first inequality follows from Observation 9, the second follows since $\sqrt{\rho_j}$ are exponentially increasing (see full version) and the third inequality follows again by Observation 9 combined with the fact that $\rho < \epsilon^2$ and $\epsilon < 10^{-2}$. ◀

Next we leverage the former lemma to bound $\min_i\{\frac{\alpha_i}{\beta_i \Pi_{j=0}^{i-1}\gamma_j}\}$ and $\frac{ALG(G_k)}{(\Pi_{i=0}^{k-1}\gamma_i)OPT(G_k)}$.

▶ **Proposition 30.** *For α_i, β_i and γ_i as in Definition 27, we have $\min_i\{\frac{\alpha_i}{\beta_i \Pi_{j=0}^{i-1}\gamma_j}\} \geq 1-23\epsilon$.*

Proof. We first bound $\frac{\alpha_i}{\beta_i}$. By the definitions of α_i and β_i we have

$$\frac{\alpha_i}{\beta_i} = \frac{1-\frac{5\sqrt{\rho_i}}{\epsilon^2}}{1+\frac{13\sqrt{\rho_i}}{\epsilon^2}} \geq \left(1-\frac{5\sqrt{\rho_i}}{\epsilon^2}\right)\left(1-\frac{13\sqrt{\rho_i}}{\epsilon^2}\right) \geq 1-\frac{18\sqrt{\rho_i}}{\epsilon^2}, \tag{8}$$

where the first inequality follows from the definitions of α_i and β_i and the rest of the inequalities follow since $\epsilon < 10^2$ and $\rho < \epsilon^6$.

By Lemma 29 we are guaranteed that $\Pi_{j=0}^{i-1}\gamma_j \leq 1+5\sqrt{\rho_i}$. Combining this with inequality 8 yields

$$\frac{\alpha_i}{\beta_i \Pi_{j=0}^{i-1}\gamma_j} \geq \frac{1-\frac{18\sqrt{\rho_i}}{\epsilon^2}}{1+5\sqrt{\rho_i}} \geq (1-\frac{18}{\epsilon^2}\sqrt{\rho_i})(1-5\sqrt{\rho_i}) \geq 1-\frac{23}{\epsilon^2}\sqrt{\rho_i},$$

and since ρ_i only increases and $\rho_{k-1} \leq \epsilon^6$ we have $\min_i\{\frac{\alpha_i}{\beta_i \Pi_{j=0}^{i-1}\gamma_j}\} \geq 1-\frac{23}{\epsilon^2}\sqrt{\rho_{k-1}} \geq 1-23\epsilon$, thereby concluding the proof. ◀

▶ **Proposition 31.** *For $\gamma_i = 1+4\sqrt{\rho_i}$ we have $\frac{ALG(G_k)}{(\Pi_{i=0}^{k-1}\gamma_i)OPT(G_k)} \geq 1-34\epsilon$.*

Proof. By Propositions 23 and 24 we are guaranteed that $\frac{ALG(G_k)}{OPT(G_k)} \geq 1-33\epsilon$. On the other hand by by Lemma 29 we are guaranteed that $\Pi_{i=0}^{k-2}\gamma_i \leq 1+5\sqrt{\rho_{k-1}}$. Therefore, if $k=1$ then $\frac{ALG(G_k)}{(\Pi_{i=0}^{k-1}\gamma_i)OPT(G_k)} = \frac{ALG(G_k)}{OPT(G_k)} \geq 1-33\epsilon$. Otherwise, we have

$$\frac{ALG(G_k)}{(\Pi_{i=0}^{k-1}\gamma_i)OPT(G_k)} \geq \frac{1-33\epsilon}{(1+4\sqrt{\rho_{k-1}})(1+5\sqrt{\rho_{k-1}})} \geq \frac{1-33\epsilon}{(1+4\epsilon^3)(1+5\epsilon^3)} \geq 1-34\epsilon,$$

where the second inequality follows since $\rho_{k-1} < \epsilon^6$ (since we recursed to step k) and the subsequent inequalities follow since $\epsilon < 10^{-3}$ - thereby concluding the proof. ◀

Finally, we combine Propositions 30 and 31 to bound ALG's approximation ratio.

▶ **Theorem 32.** *For any metric G, $\frac{ALG(G)}{OPT(G)} \geq 1-34\epsilon$.*

4.3 Analyzing the Running Time of ALG

Consider the definition of ALG. We observe that in each recursion step, the algorithm finds the layer to peel off, A, and then recurses. Therefore the running time is defined by the sum of these recursion steps, plus the terminating cases (i.e., either case (a) or case (b)). Recall that case (a) applies ALG_{d-w} on the instance, while case (b) arranges the instance arbitrarily. Therefore, a bound on cases (a) and (b) is simply a bound on the running time of ALG_{d-w} which is given by Lemma 33 (whose proof appears in the full version).

▶ **Lemma 33.** *Given an instance G, the running time of $ALG_{d-w}(G)$ is at most $(\frac{1}{\epsilon^7})^{\frac{1}{\epsilon^2}} \cdot O(n^2)$.*

▶ Remark 34. A bi-product of Lemma 28 is that the number of recursion steps is bounded by $O(\log n)$. The proof follows similarly to the proof of Lemma 48 substituting the inequality $\rho_{i+1} \geq 4\epsilon\sqrt{\rho_i}$ with $\rho_{i+1} \geq 4\sqrt{\rho_i}$ (which holds due to Lemma 28).

We are now ready to analyze the running time of ALG. (The proof is deferred to the full version.)

▶ **Theorem 35.** *The algorithm ALG is an EPRAS (with running time $O(n^2 \log n)$ plus the running time of ALG_{d-w}).*

▶ Remark 36. We remark that one may improve the running time by replacing ALG_{d-w} with any faster algorithm while slightly degrading the quality of the approximation.

5 The Hierarchical Clustering Objective

The section is outlined as follows. We begin by presenting our algorithms (first the algorithm to handle case (a) and subsequently the general algorithm). Thereafter we will bound the algorithm's approximation guarantee (by following the bounding scheme of Observation 6). Finally, we will analyze the algorithm's running time.

5.1 Defining the Algorithms

As in the linear arragement setting, we will begin by applying our general algorithm to the linear arrangement problem (which we will denote simply as ALG). The algorithm uses, as a subroutine, an algorithm to handle case (a). We denote this subroutine as ALG_{d-w} and define it following the definition of ALG.

5.1.1 Defining ALG

Here we apply our general algorithm (Algorithm 1) to the hierarchical clustering setting. In order to do so, roughly speaking, we define the layer to peel off A as all points outside of the metric's core.

Algorithm 4 Hierarchical Clustering Algorithm (ALG).

```
if $\rho \geq \epsilon^2$ then Solve the instance using $ALG_{d-w}$. ;        // case (a)
else
    Let $C$ denote the metric's core (as defined by Lemma 4).
    Let $A = V \setminus C$ denote the rest of the points.
    Arrange $A$ as a (arbitrary) ladder and denote the tree by $T_A$.
    if $W_C < 16\epsilon \cdot W_V$ then                                       // case (b)
        Arrange $C$ arbitrarily and denote the resulting tree by $T_C$.
        Attach $T_C$'s root as a child of the bottom most internal node of $T_A$ and return.
    else                                                                       // case (c)
        Continue recursively on $C$ and denote the resulting tree by $T_C$.
        Attach $T_C$'s root as a child of the bottom most internal node of $T_A$ and return.
```

▶ Remark 37. Note that Algorithm 4 conforms to the general Algorithm 1 since $C = V \setminus A$.

5.1.2 Defining ALG_{d-w}

We will use the algorithm of Vainstein et al. [31] as ALG_{d-w}. As part of their algorithm they make use of the general graph partitioning algorithm of Goldreich et al. [18] which is denoted by $PT(\cdot)$. Since we will use $PT(\cdot)$ to devise our own algorithm for the LA objective we refer the reader to Definition 21 and Theorem 22 for a more in-depth explanation of the $PT(\cdot)$ algorithm. We restate ALG_{d-w} in Algorithm 5 as defined in Vainstein et al. [31].

Algorithm 5 HC Algorithm for Sufficiently Densely Weighted Instances (ALG_{d-w}).

```
Enumerate over all trees $T$ with $k = \frac{1}{\epsilon}$ internal nodes.
for each such $T$ do
    for $\{\lambda_i\}_{i \leq k} \subset \{i\epsilon^2 n : i \in \mathbb{N} \wedge i \leq \frac{3}{\epsilon}\}$ do
        for $\{\mu_{j,j'}\}_{j \leq k, j' \leq k, j \neq j'} \subset \{i\epsilon^3 n^2 D_V : i \in \mathbb{N} \wedge i \leq \frac{9}{\epsilon}\}$ do
            Let $\Phi = \{\lambda_i, \lambda_i\}_{i=1}^{k} \cup \{\mu_{j,j'}, \mu_{j,j'}\}_{j,j'=1}^{k}$.
            Run $PT(G, \Phi, \epsilon_{err} = \epsilon^3)$. Let $P$ denote the output partition (if succeeded).
            Compute the HC objective value based on $T$ and $P$.
Return the partition $P$ and tree $T$ with maximal HC objective value.
```

5.2 Analyzing the Approximation Ratio of ALG

Now that we have defined ALG we are ready to analyze its approximation ratio. Recall that by Observation 6 it is enough to analyze the approximation ratio of cases (a), (b) and the total approximation loss generated by the recursion steps (i.e., by finding α_i, β_i and γ_i).

5.2.1 Analyzing the Approximation Ratio of Case (a) of ALG

In order to analyse the approximation ratio of ALG_{d-w} in our setting we must first recall the definition of instances with not-all-small-weights (as defined by Vainstein et al. [31]).

▶ **Definition 38.** *A metric G is said to have not all small weights if there exist constants (with respect to n_V) $c_0, c_1 < 1$ such that the fraction of weights smaller than $c_0 \cdot D_V$ is at most $1 - c_1$.*

The following theorem was presented in Vainstein et al. [31].

▶ **Theorem 39.** *For any constant $\xi > 0$ and any metric $G = (V, w)$ with not all small weights (with constants c_0 and c_1) we are guaranteed that $\frac{ALG_{d-w}(G)}{OPT(G)} \geq 1 - O(\frac{\xi}{c_0 \cdot c_1})$ and that ALG_{d-w}'s expected running time is at most $f(\frac{1}{\xi}) \cdot n^2$.*

Applying the above theorem with $\xi = \epsilon^5$ to our metric instance G_k yields Proposition 40 (whose proof is deferred to the full version).

▶ **Proposition 40.** *If ALG terminates in case (a) then $\frac{ALG_{d-w}(G_k)}{OPT(G_k)} = \frac{ALG(G_k)}{OPT(G_k)} \geq 1 - \epsilon$.*

5.2.2 Analyzing the Approximation Ratio of Case (b) of ALG

▶ **Proposition 41.** *If ALG terminates in case (b) then $\frac{ALG(G_k)}{OPT(G_k)} \geq 1 - 17\epsilon$.*

Proof. The proof appears in the full version. ◀

5.2.3 Setting the Values α_i, β_i and γ_i

Due to lack of space, we defer the following proofs to the full version.

▶ **Lemma 42.** *For A_i and C_i as defined by our algorithm applied to G_i and for $\alpha_i = n_{V_i}(W_{A_i} + W_{A_i,C_i})(1 - \sqrt{\rho_i})$ we have $ALG(G_i) \geq \alpha_i + ALG(G_{i+1})$.*

▶ **Lemma 43.** *Let $G_i = (V_i, w_i)$ and $G_{i+1} = (V_{i+1}, w_{i+1})$ denote the instances defined by the i and $i+1$ recursion steps. Furthermore, let $\beta_i = n_{V_i}(W_{A_i} + W_{A_i,C_i})$ and $\gamma_i = 1 + 2\sqrt{\rho_i}$. Therefore, $OPT(G_i) \leq \beta_i + \gamma_i OPT(G_{i+1})$.*

Thus, we combine these values in Definition 44.

▶ **Definition 44.** *We define the values α_i, β_i and γ_i as follows*

$$\alpha_i = n_{V_i}(W_{A_i} + W_{A_i,C_i})(1 - \sqrt{\rho_i}); \quad \beta_i = n_{V_i}(W_{A_i} + W_{A_i,C_i}); \quad \gamma_i = 1 + 2\sqrt{\rho_i}.$$

5.2.4 Putting it all Together

Now that we have analyzed the terminal cases of the algorithm (cases (a) and (b)) and that we have set the values of α_i, β_i and γ_i we will combine these results to prove ALG's approximation ratio (as in Observation 6). Due to lack of space we defer the proofs of this section to the full version.

▶ **Proposition 45.** *For α_i, β_i and γ_i as in Definition 44, we have* $\min_i\{\frac{\alpha_i}{\beta_i \Pi_{j=0}^{i-1}\gamma_j}\} \geq 1 - 4\epsilon$.

▶ **Proposition 46.** *For $\gamma_i = 1 + 2\sqrt{\rho_i}$ we have* $\frac{ALG(G_k)}{(\Pi_{i=0}^{k-1}\gamma_i)OPT(G_k)} \geq 1 - 23\epsilon$.

▶ **Theorem 47.** *For any metric G,* $\frac{ALG(G)}{OPT(G)} \geq 1 - 23\epsilon$.

5.3 Analyzing the Running Time of ALG

Consider the definition of ALG. In each recursion step, the algorithm finds the layer to peel off and then recurses. Therefore the running time is defined by the sum of these recursion steps, plus the terminating cases (i.e., either case (a) or case (b)). Recall that case (a) applies ALG_{d-w} on the instance, while case (b) arranges the instance arbitrarily. Therefore, a bound on cases (a) and (b) is simply a bound on the running time of ALG_{d-w} which is given by Theorem 39 [31]. In Lemma 48 we bound the number of recursion steps and subsequently prove Theorem 49 (the proofs of which appears in the full version).

▶ **Lemma 48.** *The number of recursion steps performed by Algorithm 4 is bounded by* $O(\log\log n)$.

▶ **Theorem 49.** *The algorithm ALG is an EPRAS (with running time* $O(n^2 \log\log n)$ *plus the running time of* ALG_{d-w}*).*

▶ Remark 50. We remark that one may improve the running time by replacing ALG_{d-w} with any faster algorithm while slightly degrading the quality of the approximation.

References

1 Sara Ahmadian, Vaggos Chatziafratis, Alessandro Epasto, Euiwoong Lee, Mohammad Mahdian, Konstantin Makarychev, and Grigory Yaroslavtsev. Bisect and conquer: Hierarchical clustering via max-uncut bisection. *CoRR*, abs/1912.06983, 2019. arXiv:1912.06983.

2 Noga Alon, Yossi Azar, and Danny Vainstein. Hierarchical clustering: A 0.585 revenue approximation. In Jacob D. Abernethy and Shivani Agarwal, editors, *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pages 153–162. PMLR, 2020. URL: http://proceedings.mlr.press/v125/alon20b.html.

3 Sanjeev Arora, David R. Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of np-hard problems. *J. Comput. Syst. Sci.*, 58(1):193–210, 1999. doi:10.1006/jcss.1998.1605.

4 Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 222–231. ACM, 2004. doi:10.1145/1007352.1007355.

5 Kevin Aydin, MohammadHossein Bateni, and Vahab S. Mirrokni. Distributed balanced partitioning via linear embedding. *Algorithms*, 12(8):162, 2019. doi:10.3390/a12080162.

6 MohammadHossein Bateni, Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Raimondas Kiveris, Silvio Lattanzi, and Vahab S. Mirrokni. Affinity clustering: Hierarchical clustering at scale. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6864–6874, 2017. URL: https://proceedings.neurips.cc/paper/2017/hash/2e1b24a664f5e9c18f407b2f9c73e821-Abstract.html.

7 Moses Charikar and Vaggos Chatziafratis. Approximate hierarchical clustering via sparsest cut and spreading metrics. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 841–854, 2017.

8 Moses Charikar, Vaggos Chatziafratis, and Rad Niazadeh. Hierarchical clustering better than average-linkage. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2291–2304, 2019.

9 Moses Charikar, Vaggos Chatziafratis, Rad Niazadeh, and Grigory Yaroslavtsev. Hierarchical clustering for euclidean data. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, pages 2721–2730, 2019. URL: `http://proceedings.mlr.press/v89/charikar19a.html`.

10 Moses Charikar, Mohammad Taghi Hajiaghayi, Howard J. Karloff, and Satish Rao. $l^2{}_2$ spreading metrics for vertex ordering problems. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 1018–1027. ACM Press, 2006. URL: `http://dl.acm.org/citation.cfm?id=1109557.1109670`.

11 Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. Batch active learning at scale. *CoRR*, abs/2107.14263, 2021. `arXiv:2107.14263`.

12 Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. Hierarchical clustering: Objective functions and algorithms. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 378–397, 2018.

13 Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 118–127, 2016.

14 Wenceslas Fernandez de la Vega and Marek Karpinski. Polynomial time approximation of dense weighted instances of MAX-CUT. *Electron. Colloquium Comput. Complex.*, 64, 1998. URL: `https://eccc.weizmann.ac.il/eccc-reports/1998/TR98-064/index.html`, `arXiv:TR98-064`.

15 Wenceslas Fernandez de la Vega and Claire Kenyon. A randomized approximation scheme for metric MAX-CUT. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 468–471. IEEE Computer Society, 1998. `doi:10.1109/SFCS.1998.743497`.

16 Guy Even, Joseph Naor, Satish Rao, and Baruch Schieber. Divide-and-conquer approximation algorithms via spreading metrics (extended abstract). In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 62–71. IEEE Computer Society, 1995. `doi:10.1109/SFCS.1995.492463`.

17 Uriel Feige and James R. Lee. An improved approximation ratio for the minimum linear arrangement problem. *Inf. Process. Lett.*, 101(1):26–29, 2007. `doi:10.1016/j.ipl.2006.07.009`.

18 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.

19 Mark D. Hansen. Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 604–609. IEEE Computer Society, 1989. `doi:10.1109/SFCS.1989.63542`.

20 Refael Hassin and Shlomi Rubinstein. Approximation algorithms for maximum linear arrangement. *Inf. Process. Lett.*, 80(4):171–177, 2001. `doi:10.1016/S0020-0190(01)00159-4`.

21 Marek Karpinski and Warren Schudy. Linear time approximation schemes for the gale-berlekamp game and related minimization problems. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 313–322. ACM, 2009. doi:10.1145/1536414.1536458.

22 Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 95–103. ACM, 2007. doi:10.1145/1250790.1250806.

23 Frank Thomson Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999. doi:10.1145/331524.331526.

24 Benjamin Moseley and Joshua Wang. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3094–3103, 2017.

25 Stanislav Naumov, Grigory Yaroslavtsev, and Dmitrii Avdiukhin. Objective-based hierarchical clustering of deep embedding vectors. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 9055–9063. AAAI Press, 2021. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17094.

26 Anand Rajagopalan, Fabio Vitale, Danny Vainstein, Gui Citovsky, Cecilia M. Procopiuc, and Claudio Gentile. Hierarchical clustering of data streams: Scalable algorithms and approximation guarantees. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8799–8809. PMLR, 2021. URL: http://proceedings.mlr.press/v139/rajagopalan21a.html.

27 Satish Rao and Andréa W. Richa. New approximation techniques for some ordering problems. In Howard J. Karloff, editor, *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 25-27 January 1998, San Francisco, California, USA*, pages 211–218. ACM/SIAM, 1998. URL: http://dl.acm.org/citation.cfm?id=314613.314703.

28 R. Ravi, Ajit Agrawal, and Philip N. Klein. Ordering problems approximated: Single-processor scheduling and interval graph completion. In Javier Leach Albert, Burkhard Monien, and Mario Rodríguez-Artalejo, editors, *Automata, Languages and Programming, 18th International Colloquium, ICALP91, Madrid, Spain, July 8-12, 1991, Proceedings*, volume 510 of *Lecture Notes in Computer Science*, pages 751–762. Springer, 1991. doi:10.1007/3-540-54233-7_180.

29 Paul D. Seymour. Packing directed circuits fractionally. *Comb.*, 15(2):281–288, 1995. doi:10.1007/BF01200760.

30 Baris Sumengen, Anand Rajagopalan, Gui Citovsky, David Simcha, Olivier Bachem, Pradipta Mitra, Sam Blasiak, Mason Liang, and Sanjiv Kumar. Scaling hierarchical agglomerative clustering to billion-sized datasets. *CoRR*, abs/2105.11653, 2021. arXiv:2105.11653.

31 Danny Vainstein, Vaggos Chatziafratis, Gui Citovsky, Anand Rajagopalan, Mohammad Mahdian, and Yossi Azar. Hierarchical clustering via sketches and hierarchical correlation clustering. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 559–567. PMLR, 2021. URL: http://proceedings.mlr.press/v130/vainstein21a.html.