Hierarchical Clustering: Objective Functions and Algorithms

Vincent Cohen-Addad*[1,2], Varun Kanade†[3,4], Frederik Mallmann-Trenn‡[5], and Claire Mathieu[6]

[1]Department of Computer Science, University of Copenhagen, Denmark
[2]Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6, Paris, France
[3]University of Oxford
[4]The Alan Turing Institute
[5]Massachusetts Institute for Technology
[6]École normale supérieure, CNRS, PSL Research University

Abstract

Hierarchical clustering is a recursive partitioning of a dataset into clusters at an increasingly finer granularity. Motivated by the fact that most work on hierarchical clustering was based on providing algorithms, rather than optimizing a specific objective, [19] framed similarity-based hierarchical clustering as a combinatorial optimization problem, where a 'good' hierarchical clustering is one that minimizes some cost function. He showed that this cost function has certain desirable properties, such as in order to achieve optimal cost, disconnected components must be separated first and that in 'structureless' graphs, *i.e.,* cliques, all clusterings achieve the same cost.

We take an axiomatic approach to defining 'good' objective functions for both similarity and dissimilarity-based hierarchical clustering. We characterize a set of *admissible* objective functions (that includes the one introduced by Dasgupta) that have the property that when the input admits a 'natural' ground-truth hierarchical clustering, the ground-truth clustering has an optimal value.

Equipped with a suitable objective function, we analyze the performance of practical algorithms, as well as develop better and faster algorithms for hierarchical clustering. For similarity-based hierarchical clustering, [19] showed that a simple recursive sparsest-cut based approach achieves an $O(\log^{3/2} n)$-approximation on worst-case inputs. We give a more refined analysis of the algorithm and show that it in fact achieves an $O(\sqrt{\log n})$-approximation[1]. This improves upon the LP-based $O(\log n)$-approximation of [33]. For dissimilarity-based hierarchical clustering, we show that the classic average-linkage algorithm gives a factor 2 approximation, and provide a simple and better algorithm that gives a factor 3/2 approximation. This aims at explaining the success of these heuristics in practice. Finally, we consider a 'beyond-worst-case' scenario through a generalisation of the stochastic block model for hierarchical clustering. We show that Dasgupta's cost function also has desirable properties for these inputs and we provide a simple algorithm that for graphs generated according to this model yields a $1 + o(1)$ factor approximation.

1 Introduction

A *hierarchical clustering* is a recursive partitioning of a dataset into successively smaller clusters. The input is a weighted graph whose edge weights represent pairwise similarities or dissimilarities between datapoints. A hierarchical clustering is represented by a rooted tree where each leaf represents a datapoint and each internal node represents a cluster containing its descendant leaves. Computing a hierarchical clustering is a fundamental problem in data analysis; it is routinely used to analyze, classify, and pre-process large datasets. A hierarchical clustering provides useful information about data that can be used, *e.g.,* to divide a digital image into distinct regions of different

*The project leading to this application has received funding from the European Unions Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 748094.

†This work was supported in part by The Alan Turing Institute under the EPSRC grant EP/N510129/1.

‡This work was carried out while a student at the École normale supérieure and Simon Fraser University. This work was supported in part by NSF Award Numbers BIO-1455983, CCF-1461559, and CCF-0939370.

[1][16] independently proved that the sparsest-cut based approach achieves a $O(\sqrt{\log n})$ approximation.

granularities, to identify communities in social networks at various societal levels, or to determine the ancestral tree of life. Developing robust and efficient algorithms for computing hierarchical clusterings is of importance in several research areas, such as machine learning, big-data analysis, and bioinformatics.

Compared to flat partition-based clustering (the problem of dividing the dataset into k parts), hierarchical clustering has received significantly less attention from a theory perspective. Partition-based clustering is typically framed as minimizing a well-defined objective such as k-means, k-medians, *etc.* and (approximation) algorithms to optimize these objectives have been a focus of study for at least two decades. On the other hand, hierarchical clustering has rather been studied at a more procedural level in terms of algorithms used in practice. Such algorithms can be broadly classified into two categories, *agglomerative* heuristics which build the candidate cluster tree bottom up, *e.g.,* average-linkage, single-linkage, and complete-linkage, and *divisive* heuristics which build the tree top-down, *e.g.,* bisection k-means, recursive sparsest-cut *etc.* Dasgupta [19] identified the lack of a well-defined objective function as one of the reasons why the theoretical study of hierarchical clustering has lagged behind that of partition-based clustering.

Defining a Good Objective Function. What is a 'good' output tree for hierarchical clustering? Let us suppose that the edge weights represent similarities (similar datapoints are connected by edges of high weight)[2]. Dasgupta [19] frames hierarchical clustering as a combinatorial optimization problem, where a good output tree is a tree that minimizes some cost function; but which function should that be? Each (binary) tree node is naturally associated to a cut that splits the cluster of its descendant leaves into the cluster of its left subtree on one side and the cluster of its right subtree on the other, and Dasgupta defines the objective to be the sum, over all tree nodes, of the total weight of edges crossing the cut multiplied by the cardinality of the node's cluster. In what sense is this good? Dasgupta argues that it has several attractive properties: (1) if the graph is disconnected, *i.e.,* data items in different connected components have nothing to do with one another, then the hierarchical clustering that minimizes the objective function begins by first pulling apart the connected components from one another; (2) when the input is a (unit-weight) clique then no particular structure is favored and all binary trees have the same cost; and (3) the cost function also behaves in a desirable manner for data containing a planted partition. Finally, an attempt to generalize the cost function leads to functions that violate property (2).

In this paper, we take an axiomatic approach to defining a 'good' cost function. We remark that in many applications, for example in phylogenetics, there exists an unknown 'ground truth' hierarchical clustering— the actual ancestral tree of life—from which the similarities are generated (possibly with noise), and the goal is to infer the underlying ground truth tree from the available data. In this sense, a cluster tree is good insofar as it is isomorphic to the (unknown) ground-truth cluster tree, and thus a natural condition for a 'good' objective function is one such that for inputs that admit a 'natural' ground-truth cluster tree, the value of the ground-truth tree is optimal. We provide a formal definition of inputs that admit a ground-truth cluster tree in Section 2.2.

We consider, as potential objective functions, the class of all functions that sum, over all the nodes of the tree, the total weight of edges crossing the associated cut times some function of the cardinalities of the left and right clusters (this includes the class of functions considered by Dasgupta [19]). In Section 3 we characterize the 'good' objective functions in this class and call them *admissible* objective functions. We prove that for any objective function, for any ground-truth input, the ground-truth tree has optimal cost (w.r.t to the objective function) *if and only if* the objective function (1) is symmetric (independent of the left-right order of children), (2) is increasing in the cardinalities of the child clusters, and (3) for (unit-weight) cliques, has the same cost for all binary trees (Theorem 3.1). Dasgupta's objective function is admissible in terms of the criteria described above.

In Section 5, we consider random graphs that induce a natural clustering. This model can be seen as a noisy version of our notion of ground-truth inputs and a hierarchical stochastic block model. We show that the ground-truth tree has optimal *expected cost* for any admissible objective function. Furthermore, we show that the ground-truth tree has cost at most $(1+o(1))\mathrm{OPT}$ with high probability for the objective function introduced by Dasgupta [19].

Algorithmic Results The objective functions identified in Section 3 allow us to (1) quantitatively compare the performances of algorithms used in practice and (2) design better and faster approximation algo-

[2]This entire discussion can equivalently be phrased in terms of dissimilarities without changing the essence.

rithms.[3]

Algorithms for Similarity Graphs: Dasgupta [19] shows that the *recursive ϕ-approximate sparsest cut* algorithm, that recursively splits the input graph using a ϕ-approximation to the sparsest cut problem, outputs a tree whose cost is at most $O(\phi \log n \cdot \mathrm{OPT})$. Roy and Pokutta [33] recently gave an $O(\log n)$-approximation by providing a linear programming relaxation for the problem and providing a clever rounding technique. Charikar and Chatziafratis [16] showed that the recursive ϕ-sparsest cut algorithm of Dasgupta gives an $O(\phi)$-approximation. In Section 4, we obtain an independent proof showing that the ϕ-approximate sparsest cut algorithm is an $O(\phi)$-approximation (Theorem 4.1)[4]. Our proof is quite different from the proof of [16] and relies on a charging argument. Combined with the celebrated result of Arora et al. [2], this yields an $O(\sqrt{\log n})$-approximation. The results stated here apply to Dasgupta's objective function; the approximation algorithms extend to other objective functions, though the ratio depends on the specific function being used. We conclude our analysis of the worst-case setting by showing that all the linkage-based algorithms commonly used in practice can perform rather poorly on worst-case inputs (see Sec. **??**).

Algorithms for Dissimilarity Graphs: Many of the algorithms commonly used in practice, *e.g.,* linkage-based methods, assume that the input is provided in terms of pairwise dissimilarity (*e.g.,* points that lie in a metric space). As a result, it is of interest to understand how they fare when compared using admissible objective functions for the dissimilarity setting. When the edge weights of the input graph represent dissimilarities, the picture is considerably different from an approximation perspective. For the analogue of Dasgupta's objective function in the dissimilarity setting, we show that the average-linkage algorithm (see Algorithm 3) achieves a 2-approximation (Theorem 6.1). This stands in contrast to other practical heuristic-based algorithms, which may have an approximation guarantee as bad as $\Omega(n^{1/4})$ (see [17]). Thus, using this objective-function based approach, one can conclude that the average-linkage algorithm is the more robust of the practical algorithms, perhaps explaining its success in practice. We also provide a new, simple, and better algorithm, the locally densest-cut algorithm,[5] which we show gives a 3/2-approximation (Theorem 6.3). Our results extend to any admissible objective function, though the exact approximation factor depends on the specific choice.

Structured Inputs and Beyond-Worst-Case Analysis: The recent work of Roy and Pokutta [33] and Charikar and Chatziafratis [16] have shown that obtaining constant approximation guarantees for worst-case inputs is beyond current techniques (see Section 1.2). Thus, we consider inputs that admit a 'natural' ground-truth cluster tree. For such inputs, we show that essentially all the practical algorithms do the right thing, in that they recover the ground-truth cluster tree. Since real-world inputs might exhibit a noisy structure, we consider more general scenarios:

- We consider a natural generalization of the classic stochastic block model that generates random graphs with a hidden ground-truth hierarchical clustering. We provide a simple algorithm based on singular value decomposition (SVD) and agglomerative methods that achieves a $(1 + o(1))$-approximation for Dasgupta's objective function (in fact, it recovers the ground-truth tree) with high probability. Interestingly, this algorithm is very similar to approaches used in practice for hierarchical clustering.
- We introduce the notion of a *δ-adversarially perturbed* ground-truth input, which can be viewed as being obtained from a small perturbation to an input that admits a natural ground truth cluster tree. This approach bears similarity to the stability-based conditions used by Balcan et al. [8] and Bilu and Linial [12]. We provide an algorithm that achieves a δ-approximation in both the similarity and dissimilarity settings, independent of the objective function used as long as it is admissible according to the criteria used in Section 3.

1.1 Summary of Our Contributions

Our work makes significant progress towards providing a more complete picture of objective-function based hierarchical clustering and understanding the success of the classic heuristics for hierarchical clustering.

[3] For the objective function proposed in his work, Dasgupta [19] shows that finding a cluster tree that minimizes the cost function is NP-hard. This directly applies to the admissible objective functions for the dissimilarity setting as well. Thus, the focus turns to developing approximation algorithms.

[4] Our analysis shows that the algorithm achieves a 6.75ϕ-approximation and the analysis of [16] yields a 8ϕ-approximation guarantee. This minor difference is of limited impact since the best approximation guarantee for sparsest-cut is $O(\sqrt{\log n})$.

[5] We say that a cut (A, B) is locally dense if moving a vertex from A to B or from B to A does not increase the density of the cut. One could similarly define locally-sparsest-cut.

- Characterization of 'good' objective functions. We prove that for any ground-truth input, the ground-truth tree has strictly optimal cost for an objective function *if and only if*, the objective function (1) is symmetric (independent of the left-right order of children), (2) is monotone in the cardinalities of the child clusters, and (3) for unit-weight cliques, gives the same weight to all binary trees (Theorem 3.1). We refer to such objective functions as *admissible*; according to these criteria Dasgupta's objective function is admissible.

- Worst-case approximation. First, for similarity-based inputs, we provide a new proof that the recursive ϕ-approximate sparsest cut algorithm is an $O(\phi)$-approximation (hence an $O(\sqrt{\log n})$-approximation) (Theorem 4.1) for Dasgupta's objective function. Second, for dissimilarity-based inputs, we show that the classic average-linkage algorithm is a 2-approximation (Theorem 6.1), and provide a new algorithm which we prove is a 3/2-approximation (Theorem 6.3). All those results extend to other cost functions but the approximation ratio is function-dependent.

- Beyond worst-case. First, stochastic models. We consider the *hierarchical stochastic block model* (Definition 5). We give a simple algorithm based on SVD and classic agglomerative methods that, with high probability, recovers the ground-truth tree and show that this tree has cost that is $(1+o(1))$OPT with respect to Dasgupta's objective function (Theorem 5.2). Second, adversarial models. We introduce the notion of δ-perturbed inputs, obtained by a small adversarial perturbation to ground-truth inputs, and give a simple δ-approximation algorithm (Theorem 7.5).

- Perfect inputs, perfect reconstruction. For ground-truth inputs, we note that the algorithms used in practice (the linkage algorithms, the bisection 2-centers, etc.) correctly reconstruct a ground truth tree (Theorems 7.1, 7.2, 7.3). We introduce a simple, faster algorithm that is also optimal on ground-truth inputs (Theorem 7.4).

Techniques: A key observation that is used in several of our results is that *ultrametrics* are the natural notion to model *perfect inputs* for hierarchical clustering. (Although the output of our algorithms (and that of Dasgupta [19]) are trees and not ultrametrics, it is easy to see that data arising from tree metrics does not exhibit perfect hierarchical stucture.) The use of ultrametrics in this context is motivated by the fact that they exhibit a strong relationship with hierarchical clustering heuristics (see [14]). We use ultrametrics to define ideal inputs for hierarchical clustering (Section 2) and to identify "good" objective functions (Section 3). We also use them to introduce a random graph model for hierarchical clustering (Section 5). Elsewhere, we use techniques such as concentration of measure phenomena and charging schemes for analysing approximation ratios in our results.

Proofs. Due to space limitations most proofs are omitted from this extended abstract and appear in the full version [17].

1.2 Related Work The recent paper of Dasgupta [19] served as the starting point of this work. Dasgupta [19] defined an objective function for hierarchical clustering and thus formulated the question of constructing a cluster tree as a combinatorial optimization problem. Dasgupta also showed that the resulting problem is NP-hard and that the recursive ϕ-sparsest-cut algorithm achieves an $O(\phi \log n)$-approximation. Dasgupta's results have been improved in two subsequent papers. Roy and Pokutta [33] wrote an integer program for the hierarchical clustering problem using a combinatorial characterization of the ultrametrics induced by Dasgupta's cost function. They also provide a spreading metric LP and a rounding algorithm based on sphere/region-growing that yields an $O(\log n)$-approximation. Finally, they show that no polynomial size SDP can achieve a constant factor approximation for the problem and that under the Small Set Expansion (SSE) hypothesis, no polynomial-time algorithm can achieve a constant factor approximation.

Charikar and Chatizafratis [16] also gave a proof that the problem is hard to approximate within any constant factor under the Small Set Expansion hypothesis. They also proved that the recursive ϕ-sparsest cut algorithm produces a hierarchical clustering with cost at most $O(\phi\text{OPT})$; their techniques appear to be significantly different from ours. Additionally, [16] introduce a spreading metric SDP relaxation for the hierarchical clustering problem introduced by Dasgupta that has integrality gap $O(\sqrt{\log n})$ and a spreading metric LP relaxation that yields an $O(\log n)$-approximation to the problem.

On hierarchical clustering more broadly. There is an extensive literature on hierarchical clustering and its applications. It will be impossible to discuss most of it here; for some applications the reader may refer to *e.g.,* [25, 34, 23, 15]. Algorithms for hierarchical clustering have received a lot of at-

tention from a practical perspective. For a definition and overview of *agglomerative* algorithms (such as average-linkage, complete-linkage, and single-linkage) *e.g.,* [24] and for *divisive algorithms* see *e.g.,* [35].

Most previous theoretical work on hierarchical clustering aimed at evaluating the cluster tree output by the linkage algorithms using the traditional objective functions for partition-based clustering, *e.g.,* considering k-median or k-means cost of the clusters induced by the top levels of the tree *e.g.,* [32, 20, 28]. Previous work also proved that average-linkage can be useful to recover an underlying partition-based clustering when it exists under certain stability conditions [8, 9]. The approach of this paper is different: we aim at associating a cost or a value to each hierarchical clustering and finding the best hierarchical clustering with respect to these objective functions.

In Section 3, we take an axiomatic approach toward *objective functions*. Axiomatic approach toward a *qualitative analysis of algorithms* for clustering where taken before. For example, the celebrated result of Kleinberg [26] (see also [37]) showed that there is no algorithm satisfying three natural axioms simultaneously. This approach was applied to hierarchical clustering algorithms by Carlsson and M'emoli [14] who showed that in the case of hierarchical clustering one gets a positive result, unlike the impossibility result of Kleinberg. Their focus was on finding an ultrametric (on the datapoints) that is the closest to the metric (in which the data lies) in terms of the Gromov-Hausdorf distance. Our approach is completely different as we focus on defining objective functions and use these for *quantitative analyses of algorithms*.

Our condition for inputs to have a ground-truth cluster tree, and especially their δ-adversarially perturbed versions, can be to be in the same spirit as that of the stability condition of Bilu and Linial [12] or Bilu et al. [11]: the input induces a natural clustering to be recovered whose cost is optimal. It bears some similarities with the "strict separation" condition of Balcan et al. [8], while we do not require the separation to be strict, we do require some additional hierarchical constraints. There are a variety of stability conditions that aim at capturing some of the structure that real-world inputs may exhibit *e.g.,* [4, 7, 8, 31]. Some of them induce a condition under which an underlying clustering can be mostly recovered *e.g.,* [12, 6, 7], for deterministic conditions and *e.g.,* [1, 13, 21, 18, 10] for probabilistic conditions). Imposing other conditions allows one to bypass hardness-of-approximation results for classical clustering objectives (such as k-means), and design efficient approximation algorithms *e.g.,* [3, 5, 27].

Eldridge et al. [22] also investigate the question of understanding hierarchical cluster trees for random graphs generated from graphons. Their goal is quite different from ours—they consider the "single-linkage tree" obtained using the graphon as the ground-truth tree and investigate how a cluster tree that has low *merge distortion* with respect to this *single-linkage tree* can be obtained.[6] This is quite different from the approach taken in our work which is primarily focused on understanding performance with respect to admissible cost functions.

2 Preliminaries

2.1 Notation An undirected weighted graph $G = (V, E, w)$ is defined by a finite set of vertices V, a set of edges $E \subseteq \{\{u, v\} \mid u, v \in V\}$ and a weight function $w : E \to \mathbb{R}_+$, where $\mathbb{R}_+$ denotes non-negative real numbers. We will only consider graphs with positive weights in this paper. To simplify notation (and since the graphs are undirected) we let $w(u, v) = w(v, u) = w(\{u, v\})$. When the weights on the edges are not pertinent, we simply denote graphs as $G = (V, E)$. When G is clear from the context, we denote $|V|$ by n and $|E|$ by m. We define $G[U]$ to be the subgraph induced by the nodes of U.

A *cluster tree* or *hierarchical clustering* T for graph G is a rooted binary tree with exactly $|V|$ leaves, each of which is labeled by a distinct vertex $v \in V$.[7] Given a graph $G = (V, E)$ and a cluster tree T for G, for nodes $u, v \in V$ we denote by $\mathrm{LCA}_T(u, v)$ the lowest common ancestor (furthest from the root) of u and v in T.

For any internal node N of T, we denote the subtree of T rooted at N by T_N.[8] Moreover, for any node N of T, define $V(N)$ to be the set of leaves of the subtree rooted at N. Additionally, for any two trees T_1, T_2, define the *union* of T_1, T_2 to be the tree whose root has two children C_1, C_2 such that the subtree rooted at C_1 is T_1 and the subtree rooted at C_2 is T_2.

Finally, given a weighted graph $G = (V, E, w)$, for any set of vertices $A \subseteq V$, let $w(A) = \sum_{a,b \in A} w(a, b)$ and for any set of edges E_0, let $w(E_0) = \sum_{e \in E_0} w(e)$. Finally, for any sets of vertices

[6] This is a simplistic characterization of their work. However, a more precise characterization would require introducing a lot of terminology from their paper, which is not required in this paper.

[7] In general, one can look at trees that are not binary. However, it is common practice to use binary trees in the context of hierarchical trees. Also, for results presented in this paper nothing is gained by considering trees that are not binary.

[8] For any tree T, when we refer to a subtree T' (of T) rooted at a node N, we mean the connected subgraph containing all the leaves of T that are descendant of N.

$A, B \subseteq V$, let $w(A,B) = \sum_{a\in A, b\in B} w(a,b)$.

2.2 Ultrametrics

Definition 1. (Ultrametric) *A metric space (X,d) is an ultrametric if for every $x,y,z \in X$, $d(x,y) \leqslant \max\{d(x,z), d(y,z)\}$.*

Similarity Graphs Generated from Ultrametrics We say that a weighted graph $G = (V,E,w)$ is a *similarity graph generated from an ultrametric*, if there exists an ultrametric (X,d), such that $V \subseteq X$, and for every $x,y \in V, x \neq y$, $e = \{x,y\}$ exists, and $w(e) = f(d(x,y))$, where $f : \mathbb{R}_+ \to \mathbb{R}_+$ is a non-increasing function.[9]

Dissimilarity Graphs Generated from Ultrametrics We say that a weighted graph $G = (V,E,w)$ is a *dissimilarity graph generated from an ultrametric*, if there exists an ultrametric (X,d), such that $V \subseteq X$, and for every $x,y \in V, x \neq y$, $e = \{x,y\}$ exists, and $w(e) = f(d(x,y))$, where $f : \mathbb{R}_+ \to \mathbb{R}_+$ is a non-decreasing function.

Minimal Generating Ultrametric For a weighted undirected graph $G = (V,E,w)$ generated from an ultrametric (either similarity or dissimilarity), in general there may be several ultrametrics and corresponding functions f mapping distances in the ultrametric to weights on the edges, that generate the same graph. It is useful to introduce the notion of a minimal ultrametric that generates G.

We focus on similarity graphs here; the notion of minimal generating ultrametric for dissimilarity graphs is easily obtained by suitable modifications. Let (X,d) be an ultrametric that generates $G = (V,E,w)$ and f the corresponding function mapping distances to similarities. Then we consider the ultrametric $(V,\widetilde{d})$ defined as follows: (i) $\widetilde{d}(u,u) = 0$ and (ii) for $u \neq v$,

$$\begin{aligned} &\widetilde{d}(u,v) = \widetilde{d}(v,u) \\ &= \max_{u',v'} \{d(u',v') \mid f(d(u',v')) = f(d(u,v))\} \end{aligned} \tag{2.1}$$

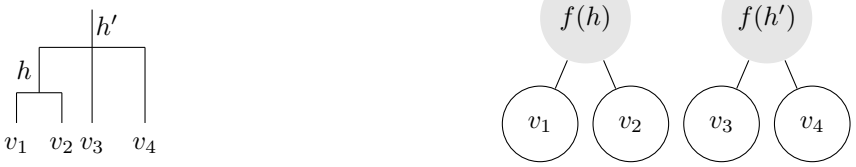
It remains to be seen that $(V,\widetilde{d})$ is indeed an ultrametric. First, notice that by definition, $\widetilde{d}(u,v) \geqslant d(u,v)$ and hence clearly $\widetilde{d}(u,v) = 0$ if and only if $u = v$ as d is the distance in an ultrametric. The fact that $\widetilde{d}$ is symmetric is immediate from the definition. The only part remaining to check is the so called *isosceles triangles with longer equal sides* conditions—the ultrametric requirement that for any u,v,w, $d(u,v) \leqslant \max\{d(u,w), d(v,w)\}$ implies that all triangles are isosceles and the two sides that are equal are at least as large as the third side. Let $u,v,w \in V$, and assume without loss of generality that according to the distance d of (V,d), $d(u,w) = d(v,w) \geqslant d(u,v)$. From (2.1) it is clear that $\widetilde{d}(u,w) = \widetilde{d}(v,w) \geqslant d(u,w)$. Also, from (2.1) and the non-increasing nature of f it is clear that if $d(u,v) \leqslant d(u',v')$, then $\widetilde{d}(u,v) \leqslant \widetilde{d}(u',v')$. Thence, $(V,\widetilde{d})$ is an ultrametric. The advantage of considering the minimal ultrametric is the following: if $\mathcal{D} = \{\widetilde{d}(u,v) \mid u,v \in V, u \neq v\}$ and $\mathcal{W} = \{w(u,v) \mid u,v \in V, u \neq v\}$, then the restriction of f from $\mathcal{D} \to \mathcal{W}$ is actually a bijection. This allows the notion of a generating tree to be defined in terms of distances in the ultrametric or weights, without any ambiguity. Applying an analogous definition and reasoning yields a similar notion for the dissimilarity case.

Definition 2. (Generating Tree) *Let $G = (V,E,w)$ be a graph generated by a minimal ultrametric (V,d) (either a similarity or dissimilarity graph). Let T be a rooted binary tree with $|V|$ leaves and $|V|-1$ internal nodes; let $\mathcal{N}$ denote the internal nodes and L the set of leaves of T and let $\sigma : L \to V$ denote a bijection between the leaves of T and nodes of V. We say that T is a generating tree for G, if there exists a weight function $W : \mathcal{N} \to \mathbb{R}_+$, such that for $N_1, N_2 \in \mathcal{N}$, if N_1 appears on the path from N_2 to the root, $W(N_1) \leqslant W(N_2)$. Moreover for every $x,y \in V$, $w(\{x,y\}) = W(LCA_T(\sigma^{-1}(x), \sigma^{-1}(y)))$.*

The notion of a generating tree defined above more or less corresponds to what is referred to as a *dendogram* in the machine learning literature *e.g.,* [14]. More formally, a dendogram is a rooted tree (not necessarily binary), where the leaves represent the datapoints. Every internal node in the tree has associated with it a height function h which is the distance between any pairs of datapoints for which it is the least common ancestor. It is a well-known fact that a set of points in an ultrametric can be represented using a dendogram *e.g.,* [14]. A dendogram can easily be modified to obtain a generating tree in the sense of Definition 2: an internal node with k children is replace by an arbitrary binary tree with k leaves and the children of the nodes in the dendogram are attached to these k leaves. The height h of this node is used to give the weight $W = f(h)$ to all the $k-1$ internal nodes added when replacing this node. Figure 1 shows this transformation.

Ground-Truth Inputs

[9] In some cases, we will say that $e = \{x,y\} \notin E$, if $w(e) = 0$. This is fine as long as $f(d(x,y)) = 0$.



(a) Dendogram on 4 nodes.

(b) Generating tree equivalent to dendogram

Figure 1: Dendogram and equivalent generating tree.

Definition 3. (Ground-Truth Input.) *We say that a graph G is a* ground-truth input *if it is a similarity or dissimilarity graph generated from an ultrametric. Equivalently, there exists a tree T that is generating for G.*

Motivation. We briefly describe the motivation for defining graphs generated from an ultrametric as ground-truth inputs. We'll focus the discussion on similarity graphs, though essentially the same logic holds for dissimilarity graphs. As described earlier, there is a natural notion of a *generating tree* associated with graphs generated from ultrametrics. This tree itself can be viewed as a cluster tree. The clusters obtained using the generating tree have the property that any two nodes in the same cluster are at least as similar to each other as they are to points outside this cluster; and this holds at every level of granularity. Furthermore, as observed by Carlsson and M'emoli [14], many practical hierarchical clustering algorithms such as the linkage based algorithms, actually output a dendogram equipped with a height function, that corresponds to an ultrametric embedding of the data. While their work focuses on algorithms that find embeddings in ultrametrics, our work focuses on finding cluster trees. We remark that these problems are related but also quite different.

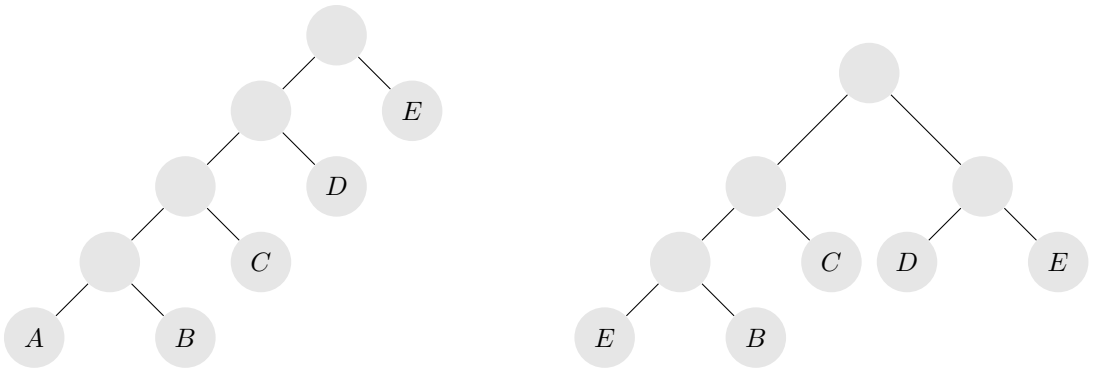
Furthermore, our results show that the linkage algorithms (and some other practical algorithms), recover a generating tree when given as input graphs that are generated from an ultrametric. Finally, we remark that relaxing the notion further leads to instances where it is hard to define a 'natural' ground-truth tree. Consider a similarity graph generated by a *tree-metric* rather than an ultrametric, where the tree is the caterpillar graph on 5 nodes (see Fig. 2(a)). Then, it is hard to argue that the tree shown in Fig. 2(b) is not a more suitable cluster tree. For instance, D and E are more similar to each other than D is to B or A. In fact, it is not hard to show that by choosing a suitable function f mapping distances from this tree metric to similarities, Dasgupta's objective function is minimized by the tree shown in Fig. 2(b), rather than the 'generating' tree in Fig. 2(a).

3 Quantifying Output Value: An Axiomatic Approach

3.1 Admissible Cost Functions Let us focus on the similarity case; in this case we use *cost* and *objective* interchangeably. Let $G = (V, E, w)$ be an undirected weighted graph and let T be a cluster tree for graph G. We want to consider cost functions for cluster trees that capture the quality of the hierarchical clustering produced by T. Following the recent work of Dasgupta [19], we adopt an approach in which a cost is assigned to each internal node of the tree T that corresponds to the quality of the split at that node.

The Axiom. A natural property we would like the cost function to satisfy is that a cluster tree T has minimum cost if and only if T is a generating tree for G. Indeed, the objective function can then be used to indicate whether a given tree is generating and so, whether it is an underlying ground-truth hierarchical clustering. Hence, the objective function acts as a "guide" for finding the correct hierarchical classification. Note that there may be multiple trees that are generating for the same graph. For example, if $G = (V, E, w)$ is a clique with every edge having the same weight then every tree is a generating tree. In these cases, all the generating tree are *valid* ground-truth hierarchical clusterings.

Following Dasgupta [19], we restrict the search space for such cost functions. For an internal node N in a clustering tree T, let $A, B \subseteq V$ be the leaves of the subtrees rooted at the left and right child of N respectively. We define the cost Γ of the tree T as the sum of the cost at every internal node N in the tree, and at an individual node N we consider cost



(a) (b)

Figure 2: (a) Caterpillar tree on 5 nodes with unit-weight edges used to define a tree metric. (b) A candidate cluster tree for the data generated using the tree metric

functions γ of the form

$$\Gamma(T) = \sum_N \gamma(N), \tag{3.2}$$

$$\gamma(N) = \left(\sum_{x \in A, y \in B} w(x,y) \right) \cdot g(|A|, |B|) \tag{3.3}$$

We remark that Dasgupta [19] defined $g(a,b) = a+b$.

Definition 4. (Admissible Cost Function) *We say that a cost function γ of the form (3.2,3.3) is* admissible *if it satisfies the condition that for all similarity graphs $G = (V, E, w)$ generated from a minimal ultrametric (V, d), a cluster tree T for G achieves the minimum cost if and only if it is a generating tree for G.*

Remark 1. *Analogously, for the dissimilarity setting we define* admissible value functions *to be the functions of the form (3.2,3.3) that satisfy: for all dissimilarity graph G generated from a minimal ultrametric (V, d), a cluster tree T for G achieves the maximum value if and only if it is a generating tree for G.*

Remark 2. *The RHS of* (3.3) *has linear dependence on the weight of the cut (A, B) in the subgraph of G induced by the vertex set $A \cup B$ as well as on an arbitrary function of the number of leaves in the subtrees of the left and right child of the internal node creating the cut (A, B). For the purpose of hierarchical clustering this form is fairly natural and indeed includes the specific cost function introduced by Dasgupta [19]. We could define the notion of admissibility for other forms of the cost function similarly and it would be of interest to understand whether they have properties that are desirable from the point of view of hierarchical clustering.*

3.2 Characterizing Admissible Cost Functions In this section, we give an almost complete characterization of admissible cost functions of the form (3.3). The following theorem shows that cost functions of this form are admissible if and only if they satisfy three conditions: that all cliques must have the same cost, symmetry and monotonicity.

Theorem 3.1. *Let γ be a cost function of the form* (3.3) *and let g be the corresponding function used to define γ. Then γ is admissible if and only if it satisfies the following three conditions.*

1. *Let $G = (V, E, w)$ be a clique, i.e., for every $x, y \in V$, $e = \{x, y\} \in E$ and $w(e) = 1$ for every $e \in E$. Then the cost $\Gamma(T)$ for every cluster tree T of G is identical.*

2. *For every $n_1, n_2 \in \mathbb{N}$, $g(n_1, n_2) = g(n_2, n_1)$.*

3. *For every $n_1, n_2 \in \mathbb{N}$, $g(n_1 + 1, n_2) > g(n_1, n_2)$.*

3.2.1 Characterizing g that satisfy conditions of Theorem 3.1 Theorem 3.1 give necessary and sufficient conditions on g for cost functions of the form (3.3) be admissible. However, it leaves open the question of the existence of functions satisfying the criteria and also characterizing the functions g themselves. The fact that such functions exist already follows from the work of Dasgupta [19], who showed that if $g(n_1, n_2) = n_1 + n_2$, then all cliques have the same cost. Clearly, g is monotone and symmetric and thus satisfies the condition of Theorem 3.1.

In order to give a more complete characterization, we define g as follows: Suppose $g(\cdot, \cdot)$ is symmetric, we define $g(n, 1)$ for all $n \geqslant 1$ so that $g(n, 1)/(n + 1)$

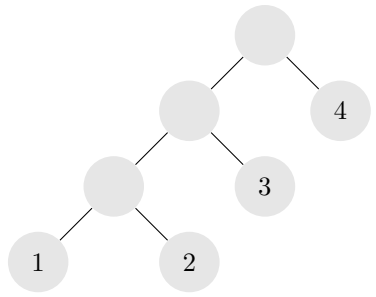


Figure 3: The caterpillar cluster tree for a clique with 4 nodes.

is non-decreasing.[10] We consider a particular cluster tree for a clique that is defined using a caterpillar graph, *i.e.,* a cluster tree where the right child of any internal node is a leaf labeled by one of the nodes of G and the left child is another internal node, except at the very bottom. Figure 3 shows a caterpillar cluster tree for a clique on 4 nodes. The cost of the clique on n nodes, say $\kappa(n)$, using this cluster tree is given by

$$\kappa(n) = \sum_{i=0}^{n-1} i \cdot g(i,1)$$

Now, we enforce the condition that all cliques have the same cost by defining $g(n_1, n_2)$ for $n_1, n_2 > 1$ suitably, in particular,

$$g(n_1,n_2) = \frac{\kappa(n_1+n_2) - \kappa(n_1) - \kappa(n_2)}{n_1 \cdot n_2} \tag{3.4}$$

Thus it only remains to be shown that g is strictly increasing. We show that for $n_2 \leqslant n_1$, $g(n_1+1, n_2) > g(n_1, n_2)$. In order to show this it suffices to show that,

$$\begin{aligned} &n_1(\kappa(n_1+n_2+1) - \kappa(n_1+1) - \kappa(n_2)) \\ &\quad - (n_1+1)(\kappa(n_1+n_2) - \kappa(n_1) - \kappa(n_2)) > 0 \end{aligned}$$

[10]The function proposed by Dasgupta [19] is $g(n,1) = n+1$, so this ratio is always 1.

Thus, consider

$$\begin{aligned} &n_1(\kappa(n_1+n_2+1) - \kappa(n_1+1) - \kappa(n_2)) \\ &\quad - (n_1+1)(\kappa(n_1+n_2) - \kappa(n_1) - \kappa(n_2)) \\ &= n_1(\kappa(n_1+n_2+1) - \kappa(n_1+n_2) - \kappa(1) \\ &\qquad - \kappa(n_1+1) + \kappa(n_1) + \kappa(1)) \\ &\quad - (\kappa(n_1+n_2) - \kappa(n_1) - \kappa(n_2)) \\ &= n_1(n_1+n_2)g(n_1+n_2,1) - n_1^2 g(n_1,1) \\ &\quad - (\kappa(n_1+n_2) - \kappa(n_1) - \kappa(n_2)) \\ &\geqslant n_1(n_1+n_2)g(n_1+n_2,1) - n_1^2 g(n_1,1) \\ &\quad - \sum_{i=n_1}^{n_1+n_2-1} i \cdot g(i,1) \\ &\geqslant \frac{g(n_1+n_2,1)}{n_1+n_2+1} \cdot ((n_1(n_1+n_2)(n_1+n_2+1) \\ &\qquad - n_1^2(n_1+1) - \sum_{i=n_1}^{n_1+n_2-1} i(i+1)) \\ &> 0 \end{aligned}$$

Above we used the fact that $g(n,1)/(n+1)$ is non-decreasing in n and some elementary calculations. This shows that the objective function proposed by Dasgupta [19] is by no means unique. Only in the last step, do we get an inequality where we use the condition that $g(n,1)/(n+1)$ is increasing. Whether this requirement can be relaxed further is also an interesting direction.

3.2.2 Characterizing Objective Functions for Dissimilarity Graphs When the weights of the edges represent dissimilarities instead of similarities, one can consider objective functions of the same form as (3.3). As mentioned in Remark 1, the difference in this case is that the goal is to maximize the objective function and hence the definition of admissibility now requires that generating trees have a value of the objective that is strictly larger than any tree that is not generating.

The characterization of admissible objective functions as given in Theorem 3.1 for the similarity case continues to hold in the case of dissimilarities. The proof follows in the same manner by appropriately switching the direction of the inequalities when required.

4 Similarity-Based Inputs: Approximation Algorithms

In this section, we analyze the recursive ϕ-sparsest-cut algorithm (see Algorithm 1) that was described previously in [19]. For clarity, we work with the cost

function introduced by Dasgupta [19]: The goal is to find a tree T minimizing $\text{cost}(T) = \sum_{N \in T} \text{cost}(N)$ where for each node N of T with children N_1, N_2, $\text{cost}(N) = w(V(N_1), V(N_2)) \cdot V(N)$. We show that the ϕ-sparsest-cut algorithm achieves a 6.75ϕ-approximation. (Charikar and Chatziafratis [16] also proved an $O(\phi)$ approximation for Dasgupta's function.) Our proof also yields an approximation guarantee not just for Dasgupta's cost function but more generally for any admissible cost function, but the approximation ratio depends on the cost function.

The ϕ-sparsest-cut algorithm (Algorithm 1) constructs a binary tree top-down by recursively finding cuts using a ϕ-approximate sparsest cut algorithm, where the *sparsest-cut* problem asks for a set A minimizing the *sparsity* $w(A, V\backslash A)/(|A||V\backslash A|)$ of the cut $(A, V\backslash A)$.

Algorithm 1 Recursive ϕ-Sparsest-Cut Algorithm for Hierarchical Clustering

1: **Input**: An edge weighted graph $G = (V, E, w)$.
2: $\{A, V\backslash A\} \leftarrow$ cut with sparsity $\leqslant \phi \cdot \min_{S \subset V} w(S, V\backslash S)/(|S||V\backslash S|)$
3: Recurse on $G[A]$ and on $G[V\backslash A]$ to obtain trees T_A and $T_{V\backslash A}$
4: **return** the tree whose root has two children, T_A and $T_{V\backslash A}$.

THEOREM 4.1. [11] *For any graph $G = (V, E)$, and weight function $w : E \to \mathbb{R}_+$, the ϕ-sparsest-cut algorithm (Algorithm 1) outputs a solution of cost at most $\frac{27}{4}\phi OPT$.*

Proof. Let $G = (V, E)$ be the input graph and n denote the total number of vertices of G. Let T denote the tree output by the algorithm and T^* be any arbitrary tree. We will prove that $\text{cost}(T) \leqslant \frac{27}{4}\phi \text{cost}(T^*)$. [12]

Recall that for an arbitrary tree T_0 and node N of T_0, the vertices corresponding to the leaves of the subtree rooted at N is denoted by $V(N)$. Consider the node N_0 of T^* that is the first node reached by the walk from the root that always goes to the child tree with the higher number of leaves, stopping when the subtree of T^* rooted at N_0 contains fewer than $2n/3$ leaves. The *balanced cut* (BC) of T^* is the cut $(V(N_0), V - V(N_0))$. For a given node N with children N_1, N_2, we say that the cut induced by N is the sum of the weights of the edges that have one extremity in $V(N_1)$ and the other in $V(N_2)$.

Let $(A \cup C, B \cup D)$ be the cut induced by the root node u of T, where A, B, C, D are such that $(A \cup B, C \cup D)$ is the balanced cut of T^*. Since $(A \cup C, B \cup D)$ is a ϕ-approximate sparsest cut:

$$\frac{w(A \cup C, B \cup D)}{|A \cup C| \cdot |B \cup D|} \leqslant \phi \frac{w(A \cup B, C \cup D)}{|A \cup B| \cdot |C \cup D|}.$$

By definition of N_0, $A \cup B$ and $C \cup D$ both have size in $[n/3, 2n/3]$, so the product of their sizes is at least $(n/3)(2n/3) = 2n^2/9$; developing $w(A \cup B, C \cup D)$ into four terms, we obtain

$$\begin{aligned} &w(A \cup C, B \cup D) \\ &\quad \leqslant \phi \frac{9}{2n^2} |A \cup C||B \cup D| \cdot \\ &\qquad (w(A, C) + w(A, D) + w(B, C) + w(B, D)) \\ &\quad \leqslant \phi \frac{9}{2} \Bigg[\frac{|B \cup D|}{n} w(A, C) + w(A, D) + \\ &\qquad w(B, C) + \frac{|A \cup C|}{n} w(B, D) \Bigg], \end{aligned}$$

and so the cost induced by node u of T^* satisfies

$$\begin{aligned} &n \cdot w(A \cup C, B \cup D) \\ &\quad \leqslant \frac{9}{2}\phi |B \cup D| w(A, C) + \frac{9}{2}\phi |A \cup C| w(B, D) \\ &\qquad + \frac{9}{2}\phi n (w(A, D) + w(B, C)). \end{aligned}$$

To account for the cost induced by u, we thus assign a charge of $(9/2)\phi|B \cup D|w(e)$ to each edge e of (A, C), a charge of $(9/2)\phi|A \cup C|w(e)$ to each edge e of (B, D), and a charge of $(9/2)\phi n w(e)$ to each edge e of (A, D) or (B, C).

When we do this for every node u of T, how much does each edge get charged?

LEMMA 4.1. *Let $G = (V, E)$ be a graph on n nodes. We consider the above charging scheme for T and T^*. Then, an edge $(v_1, v_2) \in E$ gets charged at most $(9/2)\phi \min((3/2)|V(LCA_{T^*}(v_1, v_2))|, n) w(e)$ overall, where $LCA_{T^*}(v_1, v_2)$ denotes the lowest common ancestor of v_1 and v_2 in T^*.*

We temporarily defer the proof and first see how Lemma 4.1 implies the theorem. Observe (as in [19]) that $\text{cost}(T^*) = \sum_{\{u,v\} \in E} |V(\text{LCA}_{T^*}(u, v))| w(u, v)$. Thanks to Lemma 4.1, when we sum charges assigned because of every node N of T, overall we obtain

$$\begin{aligned} \text{cost}(T) &\leqslant \frac{9}{2}\phi \sum_{\{v_1, v_2\} \in E} \frac{3}{2} |V(\text{LCA}_{T^*}(v_1, v_2))| w(v_1, v_2) \\ &= \frac{27}{4}\phi \text{cost}(T^*). \end{aligned}$$

[11] For Dasgupta's function, this was already proved in [16] with a different constant. The present, independent proof, uses a different method.

[12] The following paragraph bears similarities with the first part of the analysis of [19, Lemma 11] but we obtain a more fine-grained analysis by introducing a charging scheme.

Proof. [Proof of Lemma 4.1]

The lemma is proved by induction on the number of nodes of the graph. (The base case is obvious.) For the inductive step, consider the cut $(A \cup C, B \cup D)$ induced by the root node u of T.

- Consider the edges that cross the cut. First, observe that edges of (A, B) or of (C, D) never get charged at all. Second, an edge $e = \{v_1, v_2\}$ of (A, D) or of (B, C) gets charged $(9/2)\phi n w(e)$ when considering the cost induced by node u, and does not get charged when considering any other node of T. In T^*, edge e is separated by the cut $(A \cup B, C \cup D)$ induced by N_0, so the least common ancestor of v_1 and v_2 is the parent node of N_0 (or above), and by definition of N_0 we have $|V(\mathrm{LCA}_{T*}(v_1, v_2))| \geqslant 2n/3$, hence the lemma holds for e.

- An edge $e = \{v_1, v_2\}$ of $G[A] \cup G[C]$ does not get charged when considering the cut induced by node u. Apply Lemma 4.1 to $G[A \cup C]$ for the tree $T^*_{A\cup C}$ defined as the subtree of T^* induced by the vertices of $A \cup C$[13]. By induction, the overall charge to e due to the recursive calls for $G[A \cup C]$ is at most $(9/2)\phi \min((3/2)|V(\mathrm{LCA}_{T^*_{A\cup C}}(v_1, v_2))|, |A \cup C|)w(e)$. By definition of $T^*_{A\cup C}$, we have $|V(\mathrm{LCA}_{T^*_{A\cup C}}(v_1, v_2))| \leqslant |V(\mathrm{LCA}_{T*}(v_1, v_2))|$, and $|A \cup C| \leqslant n$, so the lemma holds for e.

- An edge $\{v_1, v_2\}$ of (A, C) gets a charge of $(9/2)\phi|B \cup D|w(e)$ plus the total charge to e coming from the recursive calls for $G[A \cup C]$ and the tree $T^*_{A\cup C}$. By induction the latter is at most

$$(9/2)\phi \min((3/2)|V(\mathrm{LCA}_{T^*_{A\cup C}}(v_1, v_2))|, |A \cup C|)w(e) \leqslant (9/2)\phi|A \cup C|w(e).$$

  Overall the charge to e is at most $(9/2)\phi n w(e)$. Since the cut induced by node u_0 of T^* separates v_1 from v_2, we have $|V(\mathrm{LCA}_{T*}(v_1, v_2))| \geqslant 2n/3$, hence the lemma holds for e. For edges of (B, D) or of $G[B] \cup G[D]$, a symmetrical argument applies.

REMARK 3. *The recursive ϕ-sparsest-cut algorithm achieves an $O(f_n\phi)$-approximation for any admissible cost function f, where $f_n = \max_n f(n)/f(\lceil n/3 \rceil)$. Indeed, adapting the definition of the balanced cut as in [19] and rescaling the charge by a factor of f_n imply the result.*

[13] note that $T^*_{A\cup C}$ is not necessarily the optimal tree for $G[A \cup C]$, which is why the lemma was stated in terms of every tree T^*, not just on the optimal tree.

We complete our study of classical algorithms for hierarchical clustering by showing that the standard agglomerative heuristics can perform poorly (Theorems 7.6, 7.7). Thus, the sparsest-cut-based approach seems to be more reliable in the worst-case. To understand better the success of the agglomerative heuristics, we restrict our attention to ground-truth inputs (Section 7), and random graphs (Section 5), and show that in these contexts these algorithms are efficient.

5 Admissible Objective Functions and Algorithms for Random Inputs

In this section, we initiate a *beyond-worst-case* analysis of the hierarchical clustering problem (see also Section 7.3). We study admissible objective functions in the context of random graphs that have a natural hierarchical structure; for this purpose, we consider a suitable generalization of the stochastic block model to hierarchical clustering.

We show that, for admissible cost functions, an underlying ground-truth cluster tree has optimal expected cost. Additionally, for a subfamily of admissible cost functions (called *smooth*, see Defn. 6) which includes the cost function introduced by Dasgupta, we show the following: The cost of the ground-truth cluster tree is with high probability sharply concentrated (up to a factor of $(1 + o(1))$ around its expectation), and so of cost at most $(1 + o(1))$OPT. This is further evidence that optimising admissible cost functions is an appropriate strategy for hierarchical clustering.

We also provide a simple algorithm based on the SVD based approach of McSherry [30] followed by a standard agglomerative heuristic that yields a hierarchical clustering which is, up to a factor $(1+o(1))$, optimal with respect to smooth admissible cost functions.

5.1 A Random Graph Model For Hierarchical Clustering We describe the random graph model for hierarchical clustering, called the hierarchical block model. This model has already been studied earlier, e.g., [29]. However, prior work has mostly focused on statistical hypothesis testing and exact recovery in some regimes. We will focus on understanding the behaviour of admissible objective functions and algorithms to output cluster trees that have almost optimal cost in terms of the objective function.

We assume that there are k "bottom"-level clusters that are then arranged in a hierarchical fashion. In order to model this we will use a similarity graph on k nodes generated from an ultrametric (see Sec. 2.2). There are $n_1, \ldots, n_k$ nodes in each of the

k clusters. Each edge is present in the graph with a probability that is a function of the clusters in which their endpoints lie and the underlying graph on k nodes generated from the ultrametric. The formal definition follows.

Definition 5. Hierarchical Stochastic Block Model (HSBM) *A hierarchical stochastic block model with k bottom-level clusters is defined as follows:*

- *Let $\widetilde{G}_k = (\widetilde{V}_k, \widetilde{E}_k, w)$ be a graph generated from an ultrametric (see Sec. 2.2), where $|\widetilde{V}_k| = k$ for each $e \in \widetilde{E}_k$, $w(e) \in (0,1)$.*[14] *Let $\widetilde{T}_k$ be a tree on k leaves, let $\widetilde{\mathcal{N}}$ denote the internal nodes of $\widetilde{T}$ and $\widetilde{L}$ denote the leaves; let $\widetilde{\sigma} : \widetilde{L} \to [k]$ be a bijection. Let $\widetilde{T}$ be generating for $\widetilde{G}_k$ with weight function $\widetilde{W} : \widetilde{\mathcal{N}} \to (0,1)$ (see Defn. 2).*
- *For each $i \in [k]$, let $p_i \in (0,1]$ be such that $p_i > \widetilde{W}(N)$, if N denotes the parent of $\widetilde{\sigma}^{-1}(i)$ in $\widetilde{T}$.*
- *For each $i \in [k]$, there is a fixed constant $f_i \in (0,1)$; furthermore $\sum_{i=1}^{k} f_i = 1$.*

Then a random graph $G = (V, E)$ on n nodes with sparsity parameter $\alpha_n \in (0,1]$ is defined as follows: $(n_1, \ldots, n_k)$ is drawn from the multinomial distribution with parameters $(n, (f_1, \ldots, f_k))$. Each vertex $i \in [n]$ is assigned a label $\psi(i) \in [k]$, so that exactly n_j nodes are assigned the label j for $j \in [k]$. An edge (i,j) is added to the graph with probability $\alpha_n p_{\psi(i)}$ if $\psi(i) = \psi(j)$ and with probability $\alpha_n \widetilde{W}(N)$ if $\psi(i) \neq \psi(j)$ and N is the least common ancestor of $\widetilde{\sigma}^{-1}(i)$ and $\widetilde{\sigma}^{-1}(j)$ in $\widetilde{T}$. The graph $G = (V, E)$ is returned without any labels.

As the definition is rather long and technical, a few remarks are in order.

- Rather than focusing on an arbitrary hierarchy on n nodes, we assume that there are k clusters (which exhibit no further hierarchy) and there is a hierarchy on these k clusters. The model assumes that k is fixed, but in future work, it may be interesting to study models where k itself may be a (modestly growing) function of n. The condition $p_i > \widetilde{W}(N)$ (where N is the parent of $\widetilde{\sigma}^{-1}(i)$) ensures that nodes in cluster i are strictly more likely to connect to each other than to nodes from any other cluster.
- The graphs generated can be of various sparsities, depending on the parameter α_n. If $\alpha_n \in (0,1)$ is a fixed constant, we will get dense graphs (with $\Omega(n^2)$ edges), however if $\alpha_n \to 0$ as $n \to \infty$, sparser graphs may be achieved. This is similar to the approach taken by Wolfe and Olhede [36] when considering random graph models generated according to graphons.

We define the *expected graph*, $\bar{G}$, which is a complete graph where an edge (i,j) has weight $p_{i,j}$ where $p_{i,j}$ is the probability with which it appears in the random graph G. In order to avoid ambiguity, we denote by $\Gamma(T; G)$ and $\Gamma(T; \bar{G})$ the costs of the cluster tree T for the unweighted (random) graph G and weighted graph $\bar{G}$ respectively. Observe that due to linearity (see Eqns. (3.2) and (3.3)), for any tree T and any admissible cost function, $\Gamma(T; \bar{G}) = \mathbb{E}\left[\,\Gamma(T; G)\,\right]$, where the expectation is with respect to the random choices of edges in G (in particular this holds even when conditioning on $n_1, \ldots, n_k$).

Furthermore, note that $\bar{G}$ itself is generated from an ultrametric and the generating trees for $\bar{G}$ are obtained as follows: Let $\widetilde{T}_k$ be any generating tree for $\widetilde{G}_k$, let $\hat{T}_1, \hat{T}_2, \ldots, \hat{T}_k$ be any binary trees with $n_1, \ldots, n_k$ leaves respectively. Let the weight of every internal node of $\hat{T}_i$ be p_i and replace each leaf l in $\widetilde{T}_k$ by $\hat{T}_{\widetilde{\sigma}(l)}$. In particular, this last point allows us to derive Proposition 5.1. We refer to any tree that is generating for the expected graph $\bar{G}$ as a *ground-truth tree* for G.

Remark 4. *Although it is technically possible to have $n_i = 0$ for some i under the model, we will assume in the rest of the section that $n_i > 0$ for each i. This avoids getting into the issue of degenerate ground-truth trees; those cases can be handled easily, but add no expository value.*

5.2 Objective Functions and Ground-Truth Tree In this section, we assume that the graphs represent similarities. This is clearly more natural in the case of unweighted graphs; however, all our results hold in the dissimilarity setting and the proofs are essentially identical.

Proposition 5.1. *Let Γ be an admissible cost function. Let G be a graph generated according to an* HSBM *(See Defn. 5). Let ψ be the (hidden) function mapping the nodes of G to $[k]$ (the bottom-level*

[14] In addition to $\widetilde{G}_k$ being generated from an ultrametric, we make the further assumption that the function $f : \mathbb{R}_+ \to \mathbb{R}_+$, that maps ultrametric distances to edge weights, has range $(0,1)$, so that the weight of an edge can be interpreted as a probability of an edge being present. We rule out $w(e) = 0$ as in that case the graph is disconnected and each component can be treated separately.

clusters). Let T be a ground-truth tree for G Then,

$$\mathbb{E}\left[\,\Gamma(T) \mid \psi\,\right] \leqslant \min_{T'} \mathbb{E}\left[\,\Gamma(T') \mid \psi\,\right].$$

Moreover, for any tree T', $\mathbb{E}\left[\,\Gamma(T) \mid \psi\,\right] = \mathbb{E}\left[\,\Gamma(T') \mid \psi\,\right]$ if and only if T' is a ground-truth tree.

DEFINITION 6. *Let γ be a cost function defined using the function $g(\cdot,\cdot)$ (see Defn. 4). We say that the cost function Γ (as defined in Eqn. 3.2) satisfies the* smoothness *property if*

$$g_{\max} := \max\{g(n_1,n_2) \mid n_1+n_2=n\} = O\left(\frac{\kappa(n)}{n^2}\right),$$

where $\kappa(n)$ is the cost of a unit-weight clique of size n under the cost function Γ.

FACT 5.1. *The cost function introduced by Dasgupta [19] satisfies the smoothness property.*

THEOREM 5.1. *Let $\alpha_n = \omega(\sqrt{\log n/n})$. Let Γ be an admissible cost function satisfying the smoothness property (Defn. 6). Let k be a fixed constant and G be a graph generated from an* HSBM *(as per Defn. 5) where the underlying graph $\widetilde{G}_k$ has k nodes and the sparsity factor is α_n. Let ψ be the (hidden) function mapping the nodes of G to $[k]$ (the bottom-level clusters). For any binary tree T with n leaves labelled by the vertices of G, the following holds with high probability:*

$$|\Gamma(T) - \mathbb{E}\left[\,\Gamma(T) \mid \psi\,\right]| \leqslant o(\mathbb{E}\left[\,\Gamma(T) \mid \psi\,\right]).$$

The expectation is taken only over the random choice of edges. In particular if T^* is a ground-truth tree for G, then, with high probability,

$$\Gamma(T^*) \leqslant (1+o(1)) \min_{T'} \Gamma(T') = (1+o(1))\,OPT.$$

5.3 Algorithm for Clustering in the HSBM In this section, we provide an algorithm for obtaining a hierarchical clustering of a graph generated from an HSBM. The algorithm is quite simple and combines approaches that are used in practice for hierarchical clustering: SVD projections and agglomerative heuristics. See Algorithm 2 for a complete description.

THEOREM 5.2. *Let $\alpha_n = \omega(\sqrt{\log n/n})$. Let Γ be an admissible cost function (Defn. 4) satisfying the smoothness property (Defn 6). Let k be a fixed constant and G be a graph generated from an* HSBM *(as per Defn. 5) where the underlying graph $\widetilde{G}_k$ has k nodes and the sparsity factor is α_n. Let T be a ground-truth tree for G. With high probability, Algorithm 2 with parameter k on graph G outputs a tree T' that satisfies $\Gamma(T) \leqslant (1+o(1))\,OPT$.*

Algorithm 2 Agglomerative Algorithm for Recovering Ground-Truth Tree of an HSBM Graph

1: **Input:** Graph $G=(V,E)$ generated from an HSBM.
2: **Parameter:** A constant k.
3: Apply (SVD) projection algorithm of [30, Thm. 12] with parameters G, k, $\delta = |V|^{-2}$, to get $\zeta(1),\ldots,\zeta(|V|) \in \mathbb{R}^{|V|}$ for vertices in V, where $\dim(\mathrm{span}(\zeta(1),\ldots,\zeta(|V|))) = k$.
4: Run the single-linkage algorithm (Alg. 6) on the points $\{\zeta(1),\ldots,\zeta(|V|)\}$ until there are exactly k clusters. Let $\mathcal{C} = \{C_1^\zeta,\ldots,C_k^\zeta\}$ be the clusters (of points $\zeta(i)$) obtained. Let $C_i \subseteq V$ denote the set of vertices corresponding to the cluster C_i^ζ.
5: **while** there are at least two clusters in $\mathcal{C}$ **do**
6: Take the pair of clusters C_i, C_j of $\mathcal{C}$ that maximizes $\frac{\mathrm{cut}(C_i,C_j)}{|C_i|\cdot|C_j|}$
7: $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C_i\} \setminus \{C_j\} \cup \{C_i \cup C_j\}$
8: **end while**
9: The sequence of merges in the while-loop (Steps 5 to 8) induces a hierarchical clustering tree on $\{C_1,\ldots,C_k\}$, say T_k' with k leaves (represented by $C_1,\ldots,C_k$). Replace each leaf of T_k' by an arbitrary binary tree on $|C_k|$ leaves labelled according to the vertices C_k to obtain T.
10: Repeat the algorithm $k' = 2k\log n$ times. Let $T^1,\ldots T^{k'}$ be the corresponding hierarchical clustering trees.
11: **Output:** Tree T^i (out of the k' candidates) that minimises $\Gamma(T_i)$.

REMARK 5. *In an* HSBM, *k is a fixed constant. Thus, even if k is not known in advance, one can simply run the Algorithm 2 with all possible different values (constantly many) and return the solution with the minimal cost $\Gamma(T)$.*

Let $G=(V,E)$ be the input graph generated according to an HSBM. Let T be the tree output by Algorithm 2. We divide the proof into two claims that correspond to the outcome of Step 3 and the while-loop (Steps 5 to 8) of Algorithm 2.

We use a result of McSherry [30] who considers a random graph model with k clusters that is (slightly) more general than the HSBM considered here. The difference is that there is no hierarchical structure on top of the k clusters in his setting; however, his goal is also simply to identify the k clusters and not any hierarchy upon them. The following theorem is derived from [30, Obs. 11 and Thm.12].

THEOREM 5.3. ([30]) *Let s be the size of the smallest cluster (of the k clusters) and δ be the confidence*

parameter. Assume that for all u, v belonging to different clusters with with adjacency vectors $\mathbf{u}, \mathbf{v}$ (i.e., u_i is 1 if the edge (u, i) exists in G and 0 otherwise) satisfy

$$\|\mathbb{E}[\mathbf{u}] - \mathbb{E}[\mathbf{v}]\|_2^2 \geqslant c \cdot k \cdot (n/s + \log(n/\delta))$$

for a large enough constant c, where $\mathbb{E}[\mathbf{u}]$ is the entry-wise expectation. Then, the algorithm of McSherry [30, Thm. 12] with parameters G, k, δ projects the columns of the adjacency matrix of G to points $\{\zeta(1), \ldots, \zeta(|V|)\}$ in a k-dimensional subspace of $\mathbb{R}^{|V|}$ such that the following holds w.p. at least $1 - \delta$ over the random graph G and with probability $1/k$ over the random bits of the algorithm. There exists $\eta > 0$ such that for any u in the i^{th} cluster and v in the j^{th} cluster:

1. *if $i = j$ then $\|\zeta(u) - \zeta(v)\|_2^2 \leqslant \eta$;*
2. *if $i \neq j$ then $\|\zeta(u) - \zeta(v)\|_2^2 > 2\eta$,*

Recall that $\psi : V \to [k]$ is the (hidden) labelling assigning each vertex of G to one of the k bottom-level clusters. Let $C_i^* = \{v \in V \mid \psi(v) = i\}$. Recall that $n_i = |V(C_i^*)|$.

The algorithm of [30, Thm. 12] might fail for two reasons. The first reason is that the random choices by the algorithm yield an incorrect clustering. This happens w.p. at most $1 - 1/k$ and we can simply repeat the algorithm sufficiently many times to be sure that at least once we get the desired result, *i.e.,* the projections satisfy the conclusion of Thm. 5.3. Claims 2 and 3 show that in this case, Steps 5 to 8 of Alg. 2 produce a tree that has cost close to optimal. The second reason for failure is the randomness in the (random) edge choices. We need to carefully take these random edge choices into account, which are only made once and therefore cannot be repeated. We explicitly describe the bad events below and show that these occur with very low probability. Ultimately, the algorithm simply outputs a tree that has the least cost among all the ones produced (and one of them is guaranteed to have cost $(1 + o(1))$OPT) with high probability.

In order to apply our algorithm (and therefore also McSherry's algorithm) we need that the graph does not "deviate" too much from its expectation. For this reason we define the following three bad events. Let $\bar{\mathcal{E}}_1$ be the event that there exists i, such that $n_i < f_i n/2$, *i.e.,* at least one of the bottom-level clusters has size that is not representative. Let $\mathcal{E}_1$ be the complement of $\bar{\mathcal{E}}_1$. If $\mathcal{E}_1$ holds the term n/s that appears in Thm. 5.3 is a constant. The second bad event is that McSherry's algorithm fails due to the random choice of edges. This happens with probability at most δ (note that the condition in Theorem 5.3 depends on δ), which we set at $\delta = \frac{1}{|V|^2}$. We denote the complement of this event $\mathcal{E}_2$. We remark that $\mathcal{E}_2$ depends only on random choices made in regards to edges (not node labels) and hence holds with probability $1 - \delta$ conditioned on $\mathcal{E}_1$.

Furthermore, we define $\mathcal{E}_3$ as follows. Let $C_1^*, \ldots, C_k^*$ be the hidden bottom-level clusters, *i.e.,* $C_i^* = \{v \mid \psi(v) = i\}$; notice that the definition of $\{C_i^*\}$ depends only on the random assignments of labels to vertices and not the random choice of edges. For the partition $C_1^*, \ldots, C_k^*$ of V and for any S_1, S_2, where S_1 and S_2 are disjoint and both sets are unions of some cluster sets from $\{C_1^*, \ldots, C_k^*\}$ the following holds:

$$\left| \mathbb{E}\left[\frac{\mathrm{cut}(S_1, S_2)}{|S_1| \cdot |S_2|} \right] - \frac{w(S_1, S_2)}{|S_1| \cdot |S_2|} \right| \leqslant \frac{1}{\log n} \tag{5.5}$$

In Claim 1 we show that the event $\mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3$ occurs with high probability.

In order to prove Theorem 5.2 we establish the following claims.

Claim 1. *With high probability, the event $\mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3$ holds.*

Observation 1. *Let $\alpha_n = \omega(\sqrt{\log n/n})$. Let G be generated by an* HSBM. *Assume that event $\mathcal{E}_1$ occurs. Let u, v be two nodes such that $i = \psi(u) \neq \psi(v) = j$. Let $\mathbf{u}$ and $\mathbf{v}$ denote the random variables corresponding to the columns of u and v in the adjacency matrix of G. Then,*

$$\|\mathbb{E}[\mathbf{u} \mid \{\mathcal{E}_1\}] - \mathbb{E}[\mathbf{v} \mid \{\mathcal{E}_1\}]\|_2^2 = \omega(\log n)$$

The expectations are with respect to the random choice of the edges.

Claim 2. *Let $\alpha_n = \omega(\sqrt{\log n/n})$. Let G be generated by an* HSBM. *Let $C_1^*, \ldots, C_k^*$ be the hidden bottom-level clusters, i.e., $C_i^* = \{v \mid \psi(v) = i\}$. Assume that event $\mathcal{E}_1$ and $\mathcal{E}_2$ occur. With probability at least $\Omega(1/k)$, the clusters obtained after Step 4 correspond to the assignment ψ, i.e., there exists a permutation $\pi : [k] \to [k]$, such that $C_j = C_{\pi(j)}^*$.*

Claim 3. *Let $\alpha_n = \omega(\sqrt{\log n/n})$. Let G be generated according to an* HSBM *and let T^* be a ground-truth tree for G. Assume that the events $\mathcal{E}_1, \mathcal{E}_2$, and $\mathcal{E}_3$ occur and that furthermore, the clusters obtained after Step 4 correspond to the assignment ψ, i.e., there exists a permutation $\pi : [k] \to [k]$ such that for each $v \in C_i$, $\psi(v) = \pi(i)$. Then, the sequence of merges in the while-loop (Steps 5 to 8) followed by Step 9 produces w.p. $\Omega(1/k)$ a tree T such that $\Gamma(T) \leqslant (1 + o(1))OPT$.*

We are ready to prove Theorem 5.2.

Proof. [Proof of Theorem 5.2] Conditioning on $\mathcal{E}_1 \cap \mathcal{E}_2\mathcal{E}_3$ which occurs w.h.p. we get from Claims 2 and 3 that w.p. at least $\Omega(1/k)$ the tree T^i (the i^{th} run of the algorithm) obtained in step 9 satisfies $\Gamma(T^i) \leqslant (1 + o(1))OPT$. It is possible to boost this probability by running Algorithm 2 multiple times. Running it $\Omega(k \log n)$ times and taking the tree with the smallest $\Gamma(T^i)$ yields the result.

6 Dissimilarity-Based Inputs: Approximation Algorithms

In this section, we consider general dissimilarity inputs and admissible objective functions for these inputs. For ease of exposition, we focus on a particular admissible objective function for dissimilarity inputs. Find T maximizing the value function corresponding to Dasgupta's cost function of Section 4: $\text{val}(T) = \sum_{N \in T} \text{val}(N)$ where for each node N of T with children N_1, N_2, $\text{val}(N) = w(V(N_1), V(N_2)) \cdot V(N)$. This optimization problem is NP-Hard [19], hence we focus on approximation algorithms.

We show (Theorem 6.1) that average-linkage achieves a 2 approximation for the problem. We then introduce a simple algorithm based on *locally-densest cuts* and show (Theorem 6.3) that it achieves a $3/2+\varepsilon$ approximation for the problem.

We remark that our proofs show that for any admissible objective function, those algorithms have approximation guarantees, but the approximation guarantee depends on the objective function.

We start with the following elementary upper bound on OPT.

FACT 6.1. *For any graph $G = (V, E)$, and weight function $w : E \to \mathbb{R}_+$, we have $OPT \leqslant n \cdot \sum_{e \in E} w(e)$.*

6.1 Average-Linkage We show that average-linkage is a 2-approximation in the dissimilarity setting.

THEOREM 6.1. *For any graph $G = (V, E)$, and weight function $w : E \to \mathbb{R}_+$, the average-linkage algorithm (Algorithm 3) outputs a solution of value at least $n \sum_{e \in E} w(e)/2 \geqslant OPT/2$.*

When two trees are chosen at Step 4 of Algorithm 3, we say that they are *merged*. We say that all the trees considered at the beginning of an iteration of the while loop are the trees that are *candidates for the merge* or simply the *candidate trees*.

We first show the following lemma and then prove the theorem.

Algorithm 3 Average-Linkage Algorithm for Hierarchical Clustering (dissimilarity setting)

1: **Input:** Graph $G = (V, E)$ with edge weights $w : E \mapsto \mathbb{R}_+$
2: Create n singleton trees.
3: **while** there are at least two trees **do**
4: Take trees roots N_1 and N_2 minimizing $\sum_{x \in V(N_1), y \in V(N_2)} w(x, y)/(|V(N_1)||V(N_2)|)$
5: Create a new tree with root N and children N_1 and N_2
6: **end while**
7: **return** the resulting binary tree T

LEMMA 6.1. *Let T be the output tree and A, B be the children of the root. We have,*

$$\frac{w(V(A), V(B))}{|V(A)| \cdot |V(B)|} \geqslant \frac{w(V(A))}{|V(A)| \cdot (|V(A)| - 1)} + \frac{w(V(B))}{|V(B)| \cdot (|V(B)| - 1)}.$$

6.2 A Simple and Better Approximation Algorithm for Worst-Case Inputs In this section, we introduce a very simple algorithm (Algorithm 5) that achieves a better approximation guarantee. The algorithm follows a divisive approach by recursively computing locally-densest cuts using a local search heuristic (see Algorithm 4). This approach is similar to the recursive-sparsest-cut algorithm of Section 4. Here, instead of trying to solve the densest cut problem (and so being forced to use approximation algorithms), we solve the simpler problem of computing a locally-densest cut. This yields both a very simple local-search-based algorithm and a good approximation guarantee.

We use the notation $A \oplus x$ to mean the set obtained by adding x to A if $x \notin A$, and by removing x from A if $x \in A$. We say that a cut (A, B) is a *ε/n-locally-densest cut* if for any x,

$$\frac{w(A \oplus x, B \oplus x)}{|A \oplus x| \cdot |B \oplus x|} \leqslant \left(1 + \frac{\varepsilon}{n}\right) \frac{w(A, B)}{|A||B|}.$$

The following local search algorithm computes an ε/n-locally-densest cut.

THEOREM 6.2. *Algorithm 4 computes an ε/n-locally-densest cut in time $\widetilde{O}(n(n + m)/\varepsilon)$.*

THEOREM 6.3. *Algorithm 5 returns a tree of value at least*

$$\frac{2n}{3}(1 - \varepsilon) \sum_e w(e) \geqslant \frac{2}{3}(1 - \varepsilon) OPT,$$

Algorithm 4 Local Search for Densest Cut

Input: Graph $G = (V, E)$ with edge weights $w : E \mapsto \mathbb{R}_+$
Let (u, v) be an edge of maximum weight
$A \leftarrow \{v\}$, $B \leftarrow V \backslash \{v\}$
while $\exists x$: $\frac{w(A \oplus x, B \oplus x)}{|A \oplus x| \cdot |B \oplus x|} > (1 + \varepsilon/n) \frac{w(A,B)}{|A||B|}$ **do**
    $A \leftarrow A \oplus x$, $B \leftarrow B \oplus x$
end while
return (A, B)

Algorithm 5 Recursive Locally-Densest-Cut for Hierarchical Clustering

Input: Graph $G = (V, E)$, with edge weights $w : E \mapsto \mathbb{R}_+$, $\varepsilon > 0$
Compute an ε/n-locally-densest cut (A, B) using Algorithm 4
Recurse on $G[A]$ and $G[B]$ to obtain rooted trees T_A and T_B.
Return the tree T whose root node has two children, T_A and T_B.

in time $\widetilde{O}(n^2(n + m)/\varepsilon)$.

REMARK 6. *The average-linkage and the recursive locally-densest-cut algorithms achieve an $O(g_n)$- and $O(h_n)$-approximation respectively, for any admissible cost function f, where $g_n = \max_n f(n)/f(\lceil n/2 \rceil)$. $h_n = \max_n f(n)/f(\lceil 2n/3 \rceil)$. An almost identical proof yields the result.*

REMARK 7. *In Section* **??**, *we show that other commonly used algorithms, such as complete-linkage, single-linkage, or bisection 2-Center, can perform arbitrarily badly. Hence average-linkage is more robust in that sense.*

7 Perfect Ground-Truth Inputs and Beyond

In this section, we focus on ground-truth inputs. We state that when the input is a perfect ground-truth input, commonly used algorithms (single linkage, average linkage, and complete linkage; as well as some divisive algorithms – the bisection k-Center and sparsest-cut algorithms) yield a tree of optimal cost, hence (by Definition 4) a ground-truth tree. Some of those results are folklore (and straightforward when there are no ties), but we have been unable to pin down a reference, so we include them for completeness (Section 7.1). We also introduce a faster optimal algorithm for "strict" ground-truth inputs (Section 7.2). The proofs present no difficulty. The meat of this section is Subsection 7.3, where we go beyond ground-truth inputs; we introduce δ-adversarially-perturbed ground-truth inputs and design a simple, more robust algorithm that, *for any* admissible objective function, is a δ-approximation.

Algorithm 6 Linkage Algorithm for Hierarchical Clustering (similarity setting)

Input: A graph $G = (V, E)$ with edge weights $w : E \mapsto \mathbb{R}_+$
Create n singleton trees. Root labels: $\mathcal{C} = \{\{v_1\}, \ldots, \{v_n\}\}$
Define dist : $\mathcal{C} \times \mathcal{C} \mapsto \mathbb{R}_+$: $\text{dist}(C_1, C_2) = \begin{cases} \frac{1}{|C_1||C_2|} \sum_{x \in C_1, y \in C_2} w((x, y)) & \textbf{Average Linkage} \\ \min_{x \in C_1, y \in C_2} w((x, y)) & \textbf{Single Linkage} \\ \max_{x \in C_1, y \in C_2} w((x, y)) & \textbf{Complete Linkage} \end{cases}$.
while there are at least two trees **do**
    Take the two trees with root labels C_1, C_2 such that $\text{dist}(C_1, C_2)$ is maximum
    Create a new tree by making those two tree children of a new root node labeled $C_1 \cup C_2$
    Remove C_1, C_2 from $\mathcal{C}$, add $C_1 \cup C_2$ to $\mathcal{C}$, and update dist
end while
return the resulting binary tree T

7.1 Perfect Ground-Truth Inputs are Easy

In the following, we refer to the *tie breaking* rule of Algorithm 6 as the rule followed by the algorithm for deciding which of C_i, C_j or C_k, C_ℓ to merge, when $\max_{C_1, C_2 \in \mathcal{C}} \text{dist}(C_1, C_2) = \text{dist}(C_i, C_j) = \text{dist}(C_k, C_\ell)$.

THEOREM 7.1. [15] *Assume that the input is a (dissimilarity or similarity) ground-truth input. Then, for any admissible objective function, the agglomerative heuristics average-linkage, single-linkage, and complete-linkage (see Algorithm 6) return an optimal solution. This holds no matter the tie breaking rule of Algorithm 6.*

Divisive Heuristics. In this section, we focus on two well-known divisive heuristics: (1) the bisection 2-Center which uses a partition-based clustering objective (the k-Center objective) to divide the input into two (non necessarily equal-size) parts (see Algorithm 7), and (2) the recursive sparsest-cut algorithm, which can be implemented efficiently for ground-truth inputs (Lemma 7.1).

Loosely speaking, we show that this algorithm computes an optimal solution if the optimal solution

[15] This Theorem may be folklore, at least when there are no ties, but we have been unable to find a reference.

Algorithm 7 Bisection 2-Center (similarity setting)

1: **Input:** A graph $G = (V, E)$ and a weight function $w : E \mapsto \mathbb{R}_+$
2: Find $\{u, v\} \subseteq V$ that maximizes $\min_x \max_{y\in\{u,v\}} w(x, y)$
3: $A \leftarrow \{x \mid w(x, u) \geqslant \max_{y\in\{u,v\}} w(x, y)\}$
4: $B \leftarrow V \backslash A$.
5: Apply Bisection 2-Center on $G[A]$ and $G[B]$ to obtain trees T_A,T_B respectively
6: **return** The union tree of T_A, T_B.

is unique. More precisely, for any similarity graph G, we say that a tree T is *strictly generating* for G if there exists a weight function W such that for any nodes N_1, N_2, if N_1 appears on the path from N_2 to the root, then $W(N_1) < W(N_2)$ and for every $x, y \in V$, $w(x, y) = W(\mathrm{LCA}_T(x, y))$. In this case we say that the input is a strict ground-truth input. In the context of dissimilarity, an analogous notion can be defined and we obtain a similar result.

THEOREM 7.2. [16] *For any admissible objective function, the bisection* 2*-Center algorithm returns an optimal solution for any similarity or dissimilarity graph G that is a strict ground-truth input.*

REMARK 8. *To extend our result to (non-strict) ground-truth inputs, one could consider the following variant of the algorithm (which bears similarities with the popular* elbow method *for partition-based clustering): Compute a k-Center clustering for all $k \in \{1, \ldots, n\}$ and partition the graph according to the k-Center clustering of the smallest $k > 1$ for which the value of the clustering increases. Mimicking the proof of Theorem 7.2, one can show that the tree output by the algorithm is generating.*

We now turn to the recursive sparsest-cut algorithm (*i.e.,* the recursive ϕ-sparsest-cut algorithm of Section 4, for $\phi = 1$). The recursive sparsest-cut consists in recursively partitioning the graph according to a sparsest cut of the graph. We show (1) that this algorithm yields a tree of optimal cost and (2) that computing a sparsest cut of a similarity graph generated from an ultrametric can be done in linear time. Finally, we observe that the analogous algorithm for the dissimilarity setting consists in recursively partitioning the graph according to the densest cut of the graph and achieves similar guarantees (and similarly the densest cut of a dissimilarity graph generated from an ultrametric can be computed in linear time).

[16] This Theorem may be folklore, but we have been unable to find a reference.

THEOREM 7.3. [17] *For any admissible objective function, the recursive sparsest-cut (respectively densest-cut) algorithm computes a tree of optimal cost if the input is a similarity (respectively dissimilarity) ground-truth input.*

We then show how to compute a sparsest-cut of a graph that is a ground-truth input.

LEMMA 7.1. *If the input graph is a ground-truth input then the sparsest cut is computed in $O(n)$ time by the following algorithm: pick an arbitrary vertex u, let $w_{\min}$ be the minimum weight of edges adjacent to u, and partition V into $A = \{x \mid w(u, x) > w_{\min}\}$ and $B = V \backslash A$.*

7.2 A Near-Linear Time Algorithm In this section, we propose a simple, optimal, algorithm for computing a generating tree of a ground-truth input. For any graph G, the running time of this algorithm is $O(n^2)$, and $\widetilde{O}(n)$ if there exists a tree T that is *strictly generating* for the input. For completeness we recall that for any graph G, we say that a tree T is strictly generating for G if there exists a weight function W such that for any nodes N_1, N_2, if N_1 appears on the path from N_2 to the root, then $W(N_1) < W(N_2)$ and for every $x, y \in V$, $w(x, y) = W(\mathrm{LCA}_T(x, y))$. In this case we say that the inputs is a *strict ground-truth input.*

The algorithm is described for the similarity setting but could be adapted to the dissimilarity case to achieve the same performances.

Algorithm 8 Fast and Simple Algorithm for Hierarchical Clustering on Perfect Data (similarity setting)

1: **Input:** A graph $G = (V, E)$ and a weight function $w : E \mapsto \mathbb{R}_+$
2: $p \leftarrow$ random vertex of V
3: Let $w_1 > \ldots > w_k$ be the edge weights of the edges that have p as an endpoint
4: Let $B_i = \{v \mid w(p, v) = w_i\}$, for $1 \leqslant i \leqslant k$.
5: Apply the algorithm recursively on each $G[B_i]$ and obtain a collection of trees $T_1, \ldots, T_k$
6: Define T_0^* as a tree with p as a single vertex
7: For any $1 \leqslant i \leqslant k$, define T_i^* to be the union of T_{i-1}^* and T_i
8: Return T_k^*

THEOREM 7.4. *For any admissible objective function, Algorithm 8 computes a tree of optimal cost in time $O(n \log^2 n)$ with high probability if the input is a*

[17] This Theorem may be folklore, at least when there are no ties, but we have been unable to find a reference.

strict ground-truth input or in time $O(n^2)$ if the input is a (non-necessarily strict) ground-truth input.

7.3 Beyond Structured Inputs Since real-world inputs might sometimes differ from our definition of ground-truth inputs introduced in Section 2, we introduce the notion of δ-adversarially-perturbed ground-truth inputs. This notion aims at accounting for noise in the data. We then design a simple and arguably more reliable algorithm (a robust variant of Algorithm 8) that achieves a δ-approximation for δ-adversarially-perturbed ground-truth inputs in $O(n(n+m))$ time. An interesting property of this algorithm is that its approximation guarantee is the same for any admissible objective function.

We first introduce the definition of δ-adversarially-perturbed ground-truth inputs. For any real $\delta \geqslant 1$, we say that a weighted graph $G = (V, E, w)$ is a *δ-adversarially-perturbed ground-truth input* if there exists an ultrametric (X, d), such that $V \subseteq X$, and for every $x, y \in V, x \neq y$, $e = \{x, y\}$ exists, and $f(d(x, y)) \leqslant w(e) \leqslant \delta f(d(x, y))$, where $f : \mathbb{R}^+ \to \mathbb{R}^+$ is a non-increasing function. This defines δ-adversarially-perturbed ground-truth inputs for similarity graphs and an analogous definition applies for dissimilarity graphs.

We now introduce a robust, simple version of Algorithm 8 that returns a δ-approximation if the input is a δ-adversarially-perturbed ground-truth inputs. Algorithm 8 was partitioning the input graph based on a single, random vertex. In this slightly more robust version, the partition is built iteratively: Vertices are added to the current part if there exists at least one vertex in the current part or in the parts that were built before with which they share an edge of high enough weight (see Algorithm 9 for a complete description).

THEOREM 7.5. *For any admissible objective function, Algorithm 9 returns a δ-approximation if the input is a δ-adversarially-perturbed ground-truth input.*

The results presented in this section show that for both the similarity and dissimilarity settings, some of the widely-used heuristics may perform badly. The proofs are neither difficult nor particularly interesting, but the results stand in sharp contrast to those for structured inputs and help motivate our study of inputs beyond worst case.

Similarity Graphs. We show that for very simple input graphs (*i.e.,* unweighted trees), the linkage algorithms (adapted to the similarity setting, see Algorithm 6) may perform badly.

Algorithm 9 Robust and Simple Algorithm for Hierarchical Clustering on δ-adversarially-perturbed ground-truth inputs (similarity setting)

Input: A graph $G = (V, E)$ and a weight function $w : E \mapsto \mathbb{R}_+$, a parameter δ
$p \leftarrow$ arbitrary vertex of V
$i \leftarrow 0$
$\widetilde{V}_i \leftarrow \{p\}$
while $\widetilde{V}_i \neq V$ **do**
    Let $p_1 \in \widetilde{V}_i, p_2 \in V \backslash \widetilde{V}_i$ s.t. (p_1, p_2) is an edge of maximum weight in the cut $(\widetilde{V}_i, V \backslash \widetilde{V}_i)$
    $w_i \leftarrow w(p_1, p_2)$
    $B_i \leftarrow \{u \mid w(p_1, u) = w_i\}$
    while $\exists u \in V \backslash (\widetilde{V}_i \cup B_i)$ s.t. $\exists v \in B_i \cup \widetilde{V}_i$, $w(u, v) \geqslant w_i$ **do**
        $B_i \leftarrow B_i \cup \{u\}$.
    end while
    $\widetilde{V}_{i+1} \leftarrow \widetilde{V}_i \cup B_i$
    $i \leftarrow i + 1$
end while
Let $B_1, \ldots, B_k$ be the sets obtained
Apply the algorithm recursively on each $G[B_i]$ and obtain a collection of trees $T_1, \ldots, T_k$
Define T_0^* as a tree with p as a single vertex
For any $1 \leqslant i \leqslant k$, define T_i^* to be the union of T_{i-1}^* and T_i
Return T_k^*

THEOREM 7.6. *There exists an infinite family of inputs on which the single-linkage and complete-linkage algorithms output a solution of cost* $\Omega(n\,OPT/\log n)$.

THEOREM 7.7. *There exists an infinite family of inputs on which the average-linkage algorithm output a solution of cost* $\Omega(n^{1/3}\,OPT)$.

Dissimilarity Graphs. We now show that single-linkage, complete-linkage, and bisection 2-Center might return a solution that is arbitrarily bad compared to OPT in some cases. Hence, since average-linkage achieves a 2-approximation in the worst-case it seems that it is more robust than the other algorithms used in practice.

THEOREM 7.8. *For each of the single-linkage, complete-linkage, and bisection* 2*-Center algorithms, there exists a family of inputs for which the algorithm outputs a solution of value* $O(OPT/n)$.

PROPOSITION 7.1. *For any input* $\mathcal{I}$ *lying in a metric space, for any solution tree* T *for* $\mathcal{I}$*, we have* $val(T) = O(OPT)$.

Acknowledgments The authors are grateful to Sanjoy Dasgupta for sharing thoughtful comments at various stages of this project.

References

[1] Sanjeev Arora and Ravi Kannan. Learning mixtures of arbitrary gaussians. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 247–257, 2001.

[2] Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2), 2009.

[3] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Stability yields a PTAS for k-median and k-means clustering. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 309–318, 2010.

[4] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Inf. Process. Lett.*, 112(1-2):49–54, 2012.

[5] Pranjal Awasthi and Or Sheffet. Improved spectral-norm bounds for clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, pages 37–49, 2012.

[6] Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Approximate clustering without the approximation. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 1068–1077, 2009.

[7] Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Clustering under approximation stability. *J. ACM*, 60(2):8, 2013.

[8] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. A discriminative framework for clustering via similarity functions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 671–680, New York, NY, USA, 2008. ACM.

[9] Maria-Florina Balcan and Yingyu Liang. Clustering under perturbation resilience. *SIAM J. Comput.*, 45(1):102–155, 2016.

[10] Maria-Florina Balcan, Heiko Röglin, and Shang-Hua Teng. Agnostic clustering. In *Algorithmic Learning Theory, 20th International Conference, ALT 2009, Porto, Portugal, October 3-5, 2009. Proceedings*, pages 384–398, 2009.

[11] Yonatan Bilu, Amit Daniely, Nati Linial, and Michael E. Saks. On the practically interesting instances of MAXCUT. In *30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, February 27 - March 2, 2013, Kiel, Germany*, pages 526–537, 2013.

[12] Yonatan Bilu and Nathan Linial. Are stable instances easy? *Combinatorics, Probability & Computing*, 21(5):643–660, 2012.

[13] S. Charles Brubaker and Santosh Vempala. Isotropic PCA and affine-invariant clustering. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 551–560, 2008.

[14] Gunnar Carlsson and Facundo Mémoli. Characterization, stability and convergence of hierarchical clustering methods. *Journal of Machine Learning Research*, 11:1425–1470, 2010.

[15] Rui M Castro, Mark J Coates, and Robert D Nowak. Likelihood based hierarchical clustering. *IEEE Transactions on signal processing*, 52(8):2308–2321, 2004.

[16] 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.

[17] Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. Hierarchical clustering : Objective functions and algorithms. Available at `https://arxiv.org/abs/1704.02147`, 2017.

[18] Sanjoy Dasgupta. Learning mixtures of gaussians. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New*

York, NY, USA, pages 634–644, 1999.

[19] 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, pages 118–127, New York, NY, USA, 2016. ACM.

[20] Sanjoy Dasgupta and Philip M Long. Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences*, 70(4):555–569, 2005.

[21] Sanjoy Dasgupta and Leonard J. Schulman. A probabilistic analysis of EM for mixtures of separated, spherical gaussians. *Journal of Machine Learning Research*, 8:203–226, 2007.

[22] Justin Eldridge, Mikhail Belkin, and Yusu Wang. Graphons, mergeons, and so on! In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2307–2315, 2016.

[23] Joseph Felsenstein and Joseph Felenstein. *Inferring phylogenies*, volume 2. Sinauer Associates Sunderland, 2004.

[24] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[25] N. Jardine and R. Sibson. *Mathematical Taxonomy*. Wiley series in probability and mathematical statistiscs. John Wiley & Sons, 1972.

[26] Jon Kleinberg. An impossibility theorem for clustering. In *NIPS*, volume 15, pages 463–470, 2002.

[27] Amit Kumar and Ravindran Kannan. Clustering with spectral norm and the k-means algorithm. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 299–308, 2010.

[28] Guolong Lin, Chandrashekhar Nagarajan, Rajmohan Rajaraman, and David P Williamson. A general approach for incremental approximation and hierarchical clustering. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1147–1156. Society for Industrial and Applied Mathematics, 2006.

[29] Vince Lyzinski, Minh Tang, Avanti Athreya, Youngser Park, and Carey E Priebe. Community detection and classification in hierarchical stochastic blockmodels. *IEEE Transactions on Network Science and Engineering*, 4(1):13–26, 2017.

[30] Frank McSherry. Spectral partitioning of random graphs. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 529–537, 2001.

[31] Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Chaitanya Swamy. The effectiveness of Lloyd-type methods for the k-means problem. *J. ACM*, 59(6):28, 2012.

[32] C Greg Plaxton. Approximation algorithms for hierarchical location problems. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 40–49. ACM, 2003.

[33] Aurko Roy and Sebastian Pokutta. Hierarchical clustering via spreading metrics. In *Advances In Neural Information Processing Systems*, pages 2316–2324, 2016.

[34] Peter HA Sneath and Robert R Sokal. Numerical taxonomy. *Nature*, 193(4818):855–860, 1962.

[35] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *In KDD Workshop on Text Mining*, 2000.

[36] Patrick J Wolfe and Sofia C Olhede. Nonparametric graphon estimation. *arXiv preprint arXiv:1309.5936*, 2013.

[37] Reza Bosagh Zadeh and Shai Ben-David. A uniqueness theorem for clustering. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 639–646. AUAI Press, 2009.