A Cost Function for Similarity-Based Hierarchical Clustering

Sanjoy Dasgupta
University of California, San Diego, USA
dasgupta@cs.ucsd.edu

ABSTRACT

The development of algorithms for hierarchical clustering has been hampered by a shortage of precise objective functions. To help address this situation, we introduce a simple cost function on hierarchies over a set of points, given pairwise similarities between those points. We show that this criterion behaves sensibly in canonical instances and that it admits a top-down construction procedure with a provably good approximation ratio.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

Keywords

Hierarchical clustering, approximation algorithm

1. INTRODUCTION

A *hierarchical clustering* is a recursive partitioning of a data set into successively smaller clusters. It is represented by a rooted tree whose leaves correspond to the data points, and each of whose internal nodes represents the cluster of its descendant leaves.

A hierarchy of this sort has several advantages over a *flat clustering*, which is a partition of the data into a fixed number of clusters. First, there is no need to specify the number of clusters in advance. Second, the output captures cluster structure at all levels of granularity, simultaneously.

There are several well-established methods for hierarchical clustering, the most prominent among which are probably the bottom-up agglomerative methods: single linkage, average linkage, and complete linkage (see, for instance, Chapter 14 of [9]). These are widely used and are part of standard packages for data analysis. Despite this, there remains an aura of mystery about the kinds of clusters that they find. In part, this is because they are specified procedurally rather than in terms of the objective functions they are trying to optimize. For many hierarchical clustering algorithms, it is hard to imagine what the objective function might be.

This is unfortunate, because the use of objective functions has greatly streamlined the development of other kinds of data analysis, such as classification methods. Once a cost function is specified, the problem definition becomes precise, and it becomes possible to study computational complexity and to compare the efficacy of different proposed algorithms. Also, in practice prior information and other requirements often need to be accommodated, and there is now a well-weathered machinery for incorporating these within cost functions, as explicit constraints or as regularization terms.

In this work we introduce a simple cost function for hierarchical clustering: a function that, given pairwise similarities between data points, assigns a score to any possible tree on those points. Through a series of lemmas (Section 2), we build up intuitions about the kinds of hierarchies that this function favors. Its behavior on some canonical examples—points on a line, complete graphs, and planted partitions—can be characterized readily (Section 3), and corresponds to what one would intuitively want in these cases. The cost function turns out to be NP-hard to optimize (Section 4), a fate it shares with all the common cost functions for flat clustering. However, it has a provably-good approximation algorithm (Section 5): a simple top-down heuristic, variants of which are used in practice for graph partitioning.

1.1 Related Work

Much of the development of hierarchical clustering has taken place within the context of phylogenetics [19, 10, 7]. Several of the methods originating in this field, such as average linkage, have subsequently been adapted for more general-purpose data analysis. This literature has also seen a progression of cost functions over taxonomies. One of the earliest of these is *parsimony*: given a collection of vectors $x_1, \ldots, x_n \in \{0,1\}^p$, the idea is to find a tree whose leaves correspond to the x_i's and whose internal nodes are also marked with vectors from the same space, so that total change along the branches of the tree (measured by Hamming distance, say) is minimized. Later work has moved towards probabilistic modeling, in which a tree defines a probability space over observable data and the goal is to find the maximum likelihood tree. This is similar in flavor to parsimony, but facilitates the addition of latent variables such as differential rates of mutation along different branches. Finally, there is the fully Bayesian setting in which a prior is placed over all possible trees and the object is to sample from the posterior distribution given the observations. Although some of the probabilistic models have been used for

STOC'16, June 19–21, 2016, Cambridge, MA, USA
ACM. 978-1-4503-4132-5/16/06...$15.00
http://dx.doi.org/10.1145/2897518.2897527

more general data (for instance, [17]), they tend to be fairly complex and often implicitly embody constraints—such as a "molecular clock", in which all leaves are expected to be roughly equidistant from the root—that are questionable outside the biological context. A more basic and interesting question is to find variants of parsimony that behave reasonably in general settings and that admit good algorithms. This is not the agenda of the present paper, however. For instance, our cost function does not assume that the data lie in any particular space, but is based purely on pairwise similarities between points.

The literature on hierarchical clustering also spans many other disciplines, and there are at least two other prominent strains of work addressing some of the same concerns that have motivated our search for a cost function.

One of these has to do with the statistical consistency of hierarchical clustering methods, as pioneered by Hartigan [8]. The aim here is to establish that if data is sampled from a fixed underlying distribution, then the tree structure obtained from the data converges in some suitable sense as the sample size grows to infinity. Only a few methods have been shown to be consistent—single linkage in one dimension, and some variants of it in higher dimension [4, 6]—and it is an open question to assess the convergence properties, if any, of other common methods such as average linkage.

Finally, there has been an effort to evaluate hierarchies in terms of cost functions for flat clustering—by evaluating the k-means cost, for example, of a hierarchy truncated to k clusters. Results of this kind are available for several standard clustering objectives, including k-center, k-median, and k-means [3, 5, 18, 14]. In the present paper, we take a different approach, treating the hierarchies themselves as first-class objects to be optimized.

On the algorithmic front, the method we ultimately present (Section 5) uses a recursive splitting strategy that is standard in multiway graph partitioning: divide the data into two clusters, then recurse [12]. The specific splitting criterion is to find an approximate sparsest cut (by spectral methods, for instance), which is somewhat surprising because our cost function includes no ratios of any sort. Recursive spectral partitioning has previously been justified for flat clustering by appealing directly to the conductance of individual clusters [11], whereas in our setting, the same method is seen to produce a hierarchy that approximately optimizes a natural cost function over trees.

2. THE COST FUNCTION AND ITS BASIC PROPERTIES

We assume that the input to the clustering procedure consists of pairwise similarities between n data points. Although it would perhaps be most natural to receive this information in matrix form, we will imagine that it is presented as a weighted graph, as this will facilitate the discussion. The input, then, is an undirected graph $G = (V, E, w)$, with one node for each point, edges between pairs of similar points, and positive edge weights $w(e)$ that capture the degree of similarity. We will sometimes omit w, in which case all edges are taken to have unit weight.

We would like to hierarchically cluster the n points in a way that is mindful of the given similarity structure. To this end, we define a cost function on possible hierarchies. This requires a little terminology. Let T be any rooted, not necessarily binary, tree whose leaves are in one-to-one correspondence with V. For any node u of T, let $T[u]$ be the subtree rooted at u, and let $\texttt{leaves}(T[u]) \subseteq V$ denote the leaves of this subtree. For leaves $i, j \in V$, the expression $i \vee j$ denotes their lowest common ancestor in T. Equivalently, $T[i \vee j]$ is the smallest subtree whose leaves include both i and j (Figure 1).

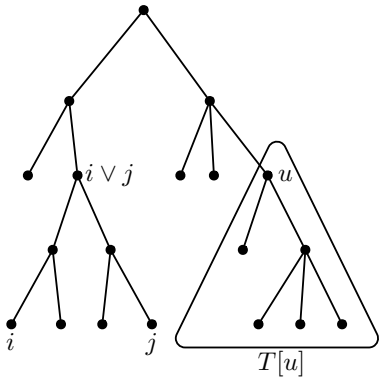


Figure 1: Induced subtrees and least common ancestors.

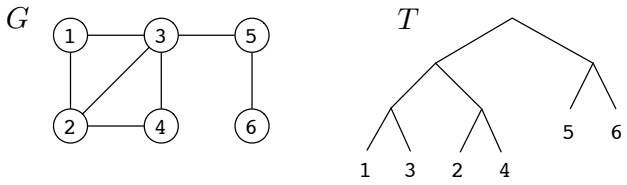


Figure 2: A small graph G and a candidate hierarchical clustering.

The edges $\{i, j\}$ in G, and their strengths w_{ij}, reflect locality. When clustering, we would like to avoid cutting too many edges. But in a hierarchical clustering, all edges do eventually get cut. All we can ask, therefore, is that edges be cut *as far down the tree as possible*. Accordingly, we define the cost of T to be

$$\text{cost}_G(T) = \sum_{\{i,j\} \in E} w_{ij} \, |\texttt{leaves}(T[i \vee j])|.$$

(Often, we will omit the subscript G.) If an edge $\{i, j\}$ of unit weight is cut all the way at the top of the tree, it incurs a penalty of n. If it is cut further down, in a subtree that contains α fraction of the data, then it incurs a smaller penalty of αn.

Figure 2 shows a toy example. Edge $\{3, 5\}$ is cut at the root of T and incurs a cost of 6. Edges $\{1, 2\}, \{2, 3\}, \{3, 4\}$ each incur a cost of 4, and the remaining three edges cost 1 apiece. Thus $\text{cost}_G(T) = 6 + 3 \times 4 + 3 \times 1 = 21$.

2.1 Two Interpretations of the Cost Function

A natural interpretation of the cost function is in terms of cuts. Each internal node of the tree corresponds to a *split* in which some subset of nodes $S \subseteq V$ is partitioned into two or more pieces. For a binary split $S \to (S_1, S_2)$, the splitting cost is $|S| \, w(S_1, S_2)$, where $w(S_1, S_2)$ is the weight of the

cut,

$$w(S_1, S_2) \;=\; \sum_{\{i,j\}\in E:\ i\in S_1, j\in S_2} w_{ij}.$$

In the example of Figure 2, the top split,

$$\{1,2,3,4,5,6\} \to (\{1,2,3,4\},\{5,6\})$$

costs 6, while the subsequent split

$$\{1,2,3,4\} \to (\{1,3\},\{2,4\})$$

costs 12.

This extends in the obvious way to k-ary splits $S \to (S_1, S_2, \ldots, S_k)$, whose cost is $|S|w(S_1, \ldots, S_k)$, with

$$w(S_1, \ldots, S_k) \;=\; \sum_{1\le i<j\le k} w(S_i, S_j).$$

The cost of a tree T is then the sum, over all internal nodes, of the splitting costs at those nodes:

$$\text{cost}(T) = \sum_{\text{splits } S\to(S_1,\ldots,S_k) \text{ in } T} |S|w(S_1,\ldots,S_k). \quad (1)$$

We would like to find the hierarchy T that minimizes this cost.

Alternatively, we can interpret a tree T as defining a distance function on points V as follows:

$$d_T(i,j) = |\texttt{leaves}(T[i \vee j])| - 1.$$

This is an ultrametric, that is,

$$d_T(i,j) \le \max(d_T(i,k), d_T(j,k)) \quad \text{for all } i,j,k \in V.$$

The cost function can then be expressed as

$$\text{cost}(T) = \sum_{\{i,j\}\in E} w_{ij} d_T(i,j) \quad + \text{constant}.$$

Thus, we seek a tree that minimizes the average distance between similar points.

2.2 Modularity of Cost

An operation we will frequently perform in the course of analysis is to replace a subtree of a hierarchical clustering T by some other subtree. The overall change in cost has a simple expression that follows immediately from the sum-of-splits formulation of equation (1).

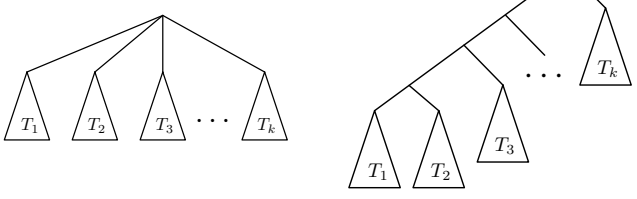
LEMMA 1. *Within any tree T, pick an internal node u. Suppose we obtain a different tree T' by replacing $T[u]$ by some other subtree T'_u with the same set of leaves. Then*

$$cost(T') = cost(T) - cost(T[u]) + cost(T'_u),$$

where the costs of $T[u]$ and T'_u are with respect to the original graph restricted to $\texttt{leaves}(T[u])$.

2.3 The Optimal Tree is Binary

In the figure below, the tree on the left is at least as costly as that on the right.



It follows that there must always exist an optimal tree that is binary—and thus we henceforth limit attention to the binary case.

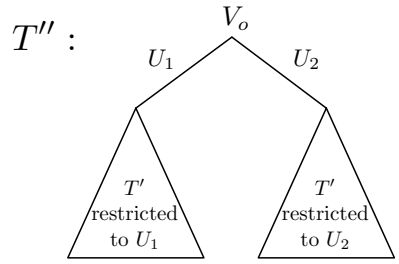
2.4 Different Connected Components Must First Be Split Apart

If G has several connected components, we would expect a hierarchical clustering to begin by pulling these apart. This natural property holds under the cost function we are considering.

LEMMA 2. *Suppose an optimal binary tree T contains a subtree T' whose leaves induce a subgraph of G that is not connected. Let the topmost split in T' be* $\texttt{leaves}(T') \to (V_1, V_2)$. *Then $w(V_1, V_2) = 0$.*

PROOF. By way of contradiction, suppose $w(V_1, V_2) > 0$. We'll construct a subtree T'' with the same leaves as T' but with lower cost—implying, by modularity of the cost function (Lemma 1), that T cannot be optimal.

For convenience, write $V_o = \texttt{leaves}(T')$, and let U_1, U_2 be a partition of V_o into two sets that are disconnected from each other (in the subgraph of G induced by V_o). T'' starts by splitting V_o into these two sets, and then copies T' on each side of the split.



The top level of T'' has cost zero. Let's compare level $\ell+1$ of T'' to level ℓ of T'. Suppose this level of T' contains some split $S \to (S_1, S_2)$. The cost of this split—its contribution to the cost of T'—is

$$\text{cost of split in } T' = |S|w(S_1, S_2).$$

Now, T'' has two corresponding splits, namely $S \cap U_1 \to (S_1 \cap U_1, S_2 \cap U_1)$ and $S \cap U_2 \to (S_1 \cap U_2, S_2 \cap U_2)$. Their combined cost is

$$\begin{aligned}
&\text{cost of splits in } T'' \\
&= |S \cap U_1| w(S_1 \cap U_1, S_2 \cap U_1) + |S \cap U_2| w(S_1 \cap U_2, S_2 \cap U_2) \\
&\le |S| w(S_1 \cap U_1, S_2 \cap U_1) + |S| w(S_1 \cap U_2, S_2 \cap U_2) \\
&= |S| w(S_1, S_2) \;=\; \text{cost of split in } T',
\end{aligned}$$

with strict inequality if $S \cap U_1$ and $S \cap U_2$ are each nonempty and $w(S_1, S_2) > 0$. The latter conditions hold for the very first split of T', namely $V_o \to (V_1, V_2)$.

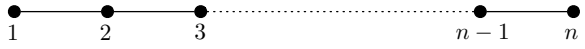
Thus, every level of T' is at least as expensive as the corresponding level of T'' and the first level is strictly more expensive, implying $\text{cost}(T'') < \text{cost}(T')$, as claimed. ∎

3. ILLUSTRATIVE EXAMPLES

To get a little more insight into our new cost function, let's see what it does in three simple and canonical situations: the line graph, the complete graph, and random graphs with planted partitions.

3.1 The Line

Consider a line on n nodes, in which every edge has unit weight:



A hierarchical clustering need only employ splits consisting of a single edge. This is because any split that involves removing multiple edges $\{e_1, e_2, \ldots\}$ can be replaced by a series of single-edge splits (first e_1, then e_2, and so on), with an accompanying reduction in cost.

Let $C(n)$ denote the cost of the best tree for the line on n nodes. The removal of an edge incurs a cost of n and creates two smaller versions of the same problem. Thus, for $n > 1$,

$$C(n) = n + \min_{1 \le j \le n-1} C(j) + C(n-j).$$

The best strategy is to split as evenly as possible, into sets of size $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$, yielding an overall cost $C(n) = n \log_2 n + O(n)$.

On the other hand, the worst splitting strategy is to choose an extremal edge, which when applied successively results in a total cost of

$$n + (n-1) + (n-2) + \cdots + 2 = \frac{n(n+1)}{2} - 1.$$

3.2 The Complete Graph

What is the best hierarchical clustering for the complete graph on n nodes, with unit edge weights? It turns out that all trees have exactly the same cost, which is intuitively pleasing—and will be a crucial tool in our later analysis.

THEOREM 3. *Let G be the complete graph on n nodes, with unit edge weights. For any binary tree T with n leaves,*

$$cost_G(T) = \frac{1}{3}(n^3 - n).$$

PROOF. For a clique on n nodes, a crude upper bound on the best achievable cost is $n \cdot \binom{n}{2}$, assessing a charge of n for every edge. We can round this up even further, to n^3.

Pick any tree T, and uncover it one split at a time, starting at the top. At each stage, we will keep track of the actual cumulative cost of the splits seen so far, along with a crude upper bound Φ on the cost yet-to-be-incurred. If, for instance, we have uncovered the tree to the point where the bottom nodes correspond to sets of size $n_1, n_2, \ldots$ (summing to n), then the bound on the cost-to-come is $\Phi = n_1^3 + n_2^3 + \cdots$.

Initially, we are at the root, have paid nothing yet, and our crude bound on the cost of what lies below is $\Phi_0 = n^3$. Let's say the t^{th} split in the tree, $1 \le t \le n-1$, breaks a set of size m into sets of size k and $m-k$. For this split, the immediate cost incurred is $c_t = mk(m-k)$ and the crude upper bound on the cost yet-to-be-incurred shrinks by

$$\Phi_{t-1} - \Phi_t = m^3 - (k^3 + (m-k)^3) = 3mk(m-k) = 3c_t.$$

When we finally reach the leaves, the remaining cost is zero, but our crude upper bound on it is $\Phi_{n-1} = n$. Thus the total cost incurred is

$$\begin{aligned} \text{cost}(T) &= c_1 + c_2 + \cdots + c_{n-1} \\ &= \frac{\Phi_0 - \Phi_1}{3} + \frac{\Phi_1 - \Phi_2}{3} + \cdots + \frac{\Phi_{n-2} - \Phi_{n-1}}{3} \\ &= \frac{\Phi_0 - \Phi_{n-1}}{3} = \frac{n^3 - n}{3}, \end{aligned}$$

as claimed. ∎

3.3 Planted Partitions

The behavior of clustering algorithms on general inputs is usually quite difficult to characterize. As a result, a common form of analysis is to focus upon instances with "planted" clusters.

The simplest such model, which we shall call the (n, p, q)-*planted partition* (for an even integer n and $0 \le q < p \le 1$), assumes that there are two clusters, each containing $n/2$ points, and that the neighborhood graph $G = (V = [n], E)$ is constructed according to a generative process. For any pair of distinct nodes i, j, an edge is placed between them with probability

$$\begin{array}{ll} p & \text{if } i,j \text{ are in the same cluster} \\ q & \text{otherwise} \end{array}$$

and these $\binom{n}{2}$ decisions are made independently. All edges in E are then assigned unit weight. It has been found, for instance, that spectral methods can recover such partitions for suitable settings of p and q [2, 16].

We will also consider what we call the *general planted partition model*, which allows for an arbitrary number of clusters $C_1, C_2, \ldots, C_k$, of possibly different sizes, and has a less constrained generative process. For any pair i, j, the probability that there is an edge between i and j is

$$\begin{array}{ll} > q & \text{if } i,j \text{ are in the same cluster} \\ q & \text{otherwise} \end{array}$$

and these decisions need not be independent. Once again, the edges have unit weight.

3.3.1 The General Planted Partition Model

Fix any set of points $V = [n]$, with unknown underlying clusters $C_1, \ldots, C_k$. Because the neighborhood graph G is stochastic, we will look at the expected cost function,

$$\begin{aligned} &\mathbb{E}[\text{cost}_G(T)] \\ &= \sum_{\{i,j\}} \mathbb{E}_G[w_{ij}] \, |\texttt{leaves}(T[i \vee j])| \\ &= \sum_{\{i,j\}} \Pr(\text{edge between } i \text{ and } j \text{ in } G) \, |\texttt{leaves}(T[i \vee j])|. \end{aligned}$$

LEMMA 4. *For a general planted partition model with clusters $C_1, \ldots, C_k$, define graph $H = (V, F, w)$ to have edges between points in the same cluster, with weights*

$$w(i,j) = Pr(\text{edge between } i \text{ and } j) - q > 0.$$

In particular, H has k connected components corresponding to the clusters.

Now fix any binary tree T with leaves V. For G generated stochastically according to the partition model,

$$\mathbb{E}[cost_G(T)] = \frac{1}{3}q(n^3 - n) + cost_H(T).$$

PROOF. Letting K_n denote the complete graph on n nodes (with unit edge weights), we have

$$\begin{aligned}
&\mathbb{E}[\text{cost}_G(T)] \\
&= \sum_{\{i,j\}} \Pr(\text{edge between } i \text{ and } j \text{ in } G)\, |\texttt{leaves}(T[i \vee j])| \\
&= q \sum_{\{i,j\}} |\texttt{leaves}(T[i \vee j])| + \\
&\sum_{\{i,j\}} (\Pr(\text{edge between } i \text{ and } j \text{ in } G) - q)\, |\texttt{leaves}(T[i \vee j])| \\
&= q \sum_{\{i,j\}} |\texttt{leaves}(T[i \vee j])| + \sum_{\{i,j\} \in F} w_{ij}\, |\texttt{leaves}(T[i \vee j])| \\
&= q\, \text{cost}_{K_n}(T) + \text{cost}_H(T).
\end{aligned}$$

We then invoke Theorem 3. ∎

Recall that the graph H of Lemma 4 has exactly k connected components corresponding to the underlying clusters $C_1, \ldots, C_k$. By Lemma 2, any tree that minimizes $\text{cost}_H(T)$, and thus $\mathbb{E}[\text{cost}_G(T)]$, must first pull these components apart.

COROLLARY 5. *Pick any binary tree T with leaves V. Under the general planted partition model, T is a minimizer of $\mathbb{E}[cost_G(\cdot)]$ if and only if, for each $1 \leq i \leq k$, T cuts all edges between cluster C_i and $V \setminus C_i$ before cutting any edges within C_i.*

Thus, in expectation, the cost function behaves in a desirable manner for data from a planted partition. It is also interesting to consider what happens when sampling error is taken into account. This requires somewhat more intricate calculations, for which we restrict attention to simple planted partitions.

3.3.2 High Probability Bounds for the Simple Planted Partition Model

Suppose graph $G = (V = [n], E)$ is drawn according to an (n, p, q)-planted partition model, as defined at the beginning of this section. We have seen that a tree T with minimum *expected* cost $\mathbb{E}[\text{cost}_G(T)]$ is one that begins with a split into the two planted clusters (call them L and R).

Moving from expectation to reality, we hope that the optimal tree for G will contain an almost-perfect split into L and R, say with at most an ϵ fraction of the points misplaced. To this end, for any $\epsilon > 0$, define a tree T (with leaves V) to be *ϵ-good* if it contains a split $S \to (S_1, S_2)$ satisfying the following two properties.

- $|S \cap L| \geq (1-\epsilon)n/2$ and $|S \cap R| \geq (1-\epsilon)n/2$. That is, S contains almost all of L and of R.
- $|S_1 \cap L| \leq \epsilon n/2$ and $|S_2 \cap R| \leq \epsilon n/2$ (or with S_1, S_2 switched). That is, $S_1 \approx R$ and $S_2 \approx L$, or the other way round.

As we will see, a tree that is not ϵ-good has expected cost quite a bit larger than that of optimal trees and, for $\epsilon \approx ((\log n)/n)^{1/2}$, this difference overwhelms fluctuations due to variance in G.

To begin with, recall from Lemma 4 that when G is drawn from the planted partition model, the expected cost of a tree T can be written as

$$\mathbb{E}[\text{cost}_G(T)] \;=\; \frac{1}{3}q(n^3 - n) + (p-q)\text{cost}_H(T), \qquad (2)$$

where H is a graph consisting of two disjoint cliques of size $n/2$, corresponding to L and R, with edges of unit weight.

We will also need to consider subgraphs of H. Therefore, define $H(\ell, r)$ to be a graph with $\ell + r$ nodes consisting of two disjoint cliques of size ℓ and r. We have already seen (Lemma 2 and Theorem 3) that the optimal cost for such a graph is

$$C(\ell, r) \;\stackrel{\text{def}}{=}\; \frac{1}{3}(\ell^3 - \ell) + \frac{1}{3}(r^3 - r),$$

and is achieved when the top split is into $H(\ell, 0)$ and $H(0, r)$. Such a split can be described as *laminar* because it conforms to the natural clusters. We will also consider splits of $H(\ell, 0)$ or of $H(0, r)$ to be laminar, since these do not cut across different clusters. As we shall now see, any non-laminar split results in increased cost.

LEMMA 6. *Consider any hierarchical clustering T of the graph $H(\ell, r)$ whose top split is into $H(\ell_1, r_1)$, $H(\ell_2, r_2)$. Then*

$$cost_{H(\ell,r)}(T) \geq C(\ell, r) + \ell_1\ell_2 r + r_1 r_2 \ell.$$

PROOF. The top split cuts $\ell_1\ell_2 + r_1 r_2$ edges. The cost of the two resulting subtrees is at least $C(\ell_1, r_1) + C(\ell_2, r_2)$. Adding these together,

$$\begin{aligned}
&\text{cost}(T) \\
&\geq (\ell_1\ell_2 + r_1 r_2)(\ell + r) + C(\ell_1, r_1) + C(\ell_2, r_2) \\
&= (\ell_1\ell_2 + r_1 r_2)(\ell + r) \\
&\quad + \frac{1}{3}\left(\ell_1^3 + \ell_2^3 - \ell_1 - \ell_2 + r_1^3 + r_2^3 - r_1 - r_2\right) \\
&= C(\ell, r) + \ell_1\ell_2 r + r_1 r_2 \ell,
\end{aligned}$$

as claimed. ∎

Now let's return to the planted partition model.

LEMMA 7. *Suppose graph $G = (V = [n], E)$ is generated from the (n, p, q)-planted partition model with $p > q$. Let T^* be a tree whose topmost split divides V into the planted clusters L, R. Meanwhile, let T be a binary tree that is not ϵ-good, for some $0 < \epsilon < 1/6$. Then*

$$\mathbb{E}[cost_G(T)] > \mathbb{E}[cost_G(T^*)] + \frac{1}{16}\epsilon(p-q)n^3.$$

PROOF. Invoking equation (2), we have

$$\begin{aligned}
&\mathbb{E}[\text{cost}_G(T)] - \mathbb{E}[\text{cost}_G(T^*)] \\
&= (p-q)\left(\text{cost}_{H(n/2,n/2)}(T) - \text{cost}_{H(n/2,n/2)}(T^*)\right) \\
&= (p-q)\left(\text{cost}_{H(n/2,n/2)}(T) - C(n/2, n/2)\right).
\end{aligned}$$

Thus our goal is to lower-bound the excess cost of T, above the optimal value of $C(n/2, n/2)$, due to the non-laminarity of its splits. Lemma 6 gives us the excess cost due to any particular split in T, and we will, in general, need to sum this over several splits.

Consider the uppermost node in T whose leaves include at least a $(1-\epsilon)$ fraction of L as well as of R. Call these points $L_o \subseteq L$ and $R_o \subseteq R$, and denote the split at this node by

$$(L_o \cup R_o) \to (L_1 \cup R_1, L_2 \cup R_2),$$

where L_1, L_2 is a partition of L_o and R_1, R_2 is a partition of R_o. It helps to think of L_o has having been divided into a smaller part and a larger part; and likewise with R_o. Since T is not ϵ-good, it must either be the case that one of the smaller parts is not small enough, or that the two smaller parts have landed up on the same side of the split instead of on opposite sides. Formally, one of the two following situations must hold:

1. $\min(|L_1|, |L_2|) > \epsilon n/2$ or $\min(|R_1|, |R_2|) > \epsilon n/2$
2. ($|L_1| \leq \epsilon n/2$ and $|R_1| \leq \epsilon n/2$) or ($|L_2| \leq \epsilon n/2$ and $|R_2| \leq \epsilon n/2$)

In the first case, assume without loss of generality that $|L_1|, |L_2| > \epsilon n/2$. Then the excess cost due to the non-laminarity of the split is, from Lemma 6, at least

$$|L_1| \cdot |L_2| \cdot |R| > \frac{\epsilon n}{2} \cdot \frac{(1-2\epsilon)n}{2} \cdot \frac{(1-\epsilon)n}{2}$$

by construction.

In the second case, suppose without loss of generality that $|L_1|, |R_1| \leq \epsilon n/2$. Since we have chosen the uppermost node whose leaves include $(1-\epsilon)$ of both L and R, we may also assume $|L_2| < (1-\epsilon)n/2$. The excess cost of this non-laminar split and of all other splits $S \to (S_1, S_2)$ on the path up to the root is, again by Lemma 6, at least

$$\begin{aligned}
&\sum_{S \to (S_1,S_2)} |L \cap S_1| \cdot |L \cap S_2| \cdot |R \cap S| \\
&\geq \sum_{S \to (S_1,S_2)} \min(|L \cap S_1|, |L \cap S_2|) \cdot \frac{(1-2\epsilon)n}{2} \cdot \frac{(1-\epsilon)n}{2} \\
&> \frac{\epsilon n}{2} \cdot \frac{(1-2\epsilon)n}{2} \cdot \frac{(1-\epsilon)n}{2},
\end{aligned}$$

where the last inequality comes from observing that the smaller portions of each split of L down from the root have a combined size of exactly $|L \setminus L_2| > \epsilon n/2$.

In either case, we have

$$\begin{aligned}
\text{cost}_{H(n/2,n/2)}(T) &> C(n/2, n/2) + \epsilon(1-\epsilon)(1-2\epsilon)\frac{n^3}{8} \\
&\geq C(n/2, n/2) + \frac{\epsilon n^3}{16},
\end{aligned}$$

which we substitute into our earlier characterization of expected cost. ∎

Thus there is a significant difference in expected cost between an optimal tree and one that is not ϵ-good for large enough ϵ. We now study the discrepancy between $\text{cost}(T)$ and its expectation.

LEMMA 8. *Let $G = (V = [n], E)$ be a random draw from the (n, p, q)-planted partition model. Then for any $0 < \delta < 1$, with probability at least $1 - \delta$,*

$$\max_T |cost_G(T) - \mathbb{E}[cost_G(T)]| \leq \frac{n^2}{2}\sqrt{2n \ln 2n + \ln \frac{2}{\delta}},$$

where the maximum is over all binary trees with leaves V.

PROOF. Fix any tree T. Then $\text{cost}_G(T)$ is a function of $\binom{n}{2}$ independent random variables (the presence or absence of each edge); moreover, any one variable can change it by at most n. By McDiarmid's inequality [15], for any $t > 0$,

$$\begin{aligned}
\Pr(|\text{cost}_G(T) - \mathbb{E}\text{cost}_G(T)| > tn^2) &\leq 2\exp\left(-\frac{2t^2n^4}{\binom{n}{2}n^2}\right) \\
&\leq 2e^{-4t^2}.
\end{aligned}$$

Next, we need an upper bound on the number of possible hierarchical clusterings on n points. Imagine reconstructing any given tree T in a bottom-up manner, in a sequence of $n-1$ steps, each of which involves merging two existing nodes and assigning them a parent. The initial set of points can be numbered $1, \ldots, n$ and new internal nodes can be assigned numbers $n+1, \ldots, 2n-1$ as they arise. Each merger chooses from at most $(2n)^2$ possibilities, so there are at most $(2n)^{2n}$ possible trees overall.

Taking a union bound over these trees, and setting $4t^2 = \ln(2(2n)^{2n}/\delta)$, yields the result. ∎

Putting these pieces together gives a characterization of the tree that minimizes the cost function, for data from a simple planted partition.

THEOREM 9. *Suppose $G = (V = [n], E)$ is generated at random from an (n, p, q)-planted partition model. Let T be the binary tree that minimizes $cost_G(\cdot)$. Then for any $\delta > 0$, with probability at least $1 - \delta$, T is ϵ-good for*

$$\epsilon = \frac{16}{p-q}\sqrt{\frac{2\ln 2n}{n} + \frac{1}{n^2}\ln\frac{2}{\delta}},$$

provided this quantity is at most $1/6$.

PROOF. Let T^* be a tree whose top split divides V into its true clusters, and let T be the minimizer of $\text{cost}_G(\cdot)$. Since $\text{cost}_G(T) \leq \text{cost}_G(T^*)$, it follows from Lemma 8 that

$$\mathbb{E}[\text{cost}_G(T)] \leq \mathbb{E}[\text{cost}_G(T^*)] + n^2\sqrt{2n\ln 2n + \ln\frac{2}{\delta}}.$$

Now, if T fails to be ϵ-good for some $0 < \epsilon \leq 1/6$, then by Lemma 7,

$$\epsilon < \frac{16}{(p-q)n^3}\left(\mathbb{E}[\text{cost}_G(T)] - \mathbb{E}[\text{cost}_G(T^*)]\right).$$

Combining these inequalities yields the theorem. ∎

In summary, when data is generated from a simple partition model, the tree that minimizes $\text{cost}(T)$ almost perfectly respects the planted clusters, misplacing at most an $O(\sqrt{(\ln n)/n})$ fraction of the data.

4. HARDNESS OF FINDING THE OPTIMAL CLUSTERING

In this section, we will see that finding the tree that minimizes the cost function is NP-hard. We begin by considering a related problem.

4.1 Maximizing the Cost Function

Interestingly, the problem of *maximizing* $\text{cost}(T)$ is equivalent to the problem of minimizing it. To see this, let $G = (V, E)$ be any graph with unit edge weights, and let $G^c = (V, E^c)$ denote its complement. Pick any tree T and

assess its suitability as a hierarchical clustering of G and also of G^c:

$$\begin{aligned}
&\text{cost}_G(T) + \text{cost}_{G^c}(T) \\
&= \sum_{\{i,j\}\in E} |\texttt{leaves}(T[i \vee j])| + \sum_{\{i,j\}\in E^c} |\texttt{leaves}(T[i \vee j])| \\
&= \sum_{\{i,j\}} |\texttt{leaves}(T[i \vee j])| \\
&= \text{cost}_{K_n}(T) \;=\; \frac{n^3 - n}{3}
\end{aligned}$$

where K_n is the complete graph on n nodes, and the last equality is from Theorem 3. In short: a tree that maximizes cost(T) on G also minimizes cost(T) on G^c, and vice versa. A similar relation holds in the presence of edge weights, where the complementary weights w^c are chosen so that $w(i,j) + w^c(i,j)$ is constant across all i, j.

In proving hardness, it will be convenient to work with the maximization version.

4.2 A Variant of Not-all-equal SAT

We will reduce from a special case of not-all-equal SAT that we call NAESAT*:

> NAESAT*
>
> *Input:* A Boolean formula in CNF such that each clause has either two or three literals, and each variable appears exactly three times: once in a 3-clause and twice, with opposite polarities, in 2-clauses.
>
> *Question:* Is there an assignment to the variables under which every clause has at least one satisfied literal and one unsatisfied literal?

In a commonly-used version of NAESAT, which we will take to be standard, there are exactly three literals per clause but each variable can occur arbitrarily often. Given an instance $\phi(x_1, \ldots, x_n)$ of this type, we can create an instance ϕ' of NAESAT*as follows:

- Suppose variable x_i occurs k times in ϕ. Without loss of generality, $k > 1$ (otherwise we can discard the containing clause with impunity). Create k new variables $x_{i1}, \ldots, x_{ik}$ to replace these occurrences.
- Enforce agreement amongst $x_{i1}, \ldots, x_{ik}$ by adding implications

  $$(\overline{x}_{i1} \vee x_{i2}),\ (\overline{x}_{i2} \vee x_{i3}), \ldots, (\overline{x}_{ik} \vee x_{i1}).$$

  These k clauses are satisfied $\iff$ they are not-all-equal satisfied $\iff x_{i1} = x_{i2} = \cdots = x_{ik}$.

Then ϕ' has the required form, and moreover is not-all-equal satisfiable if and only if ϕ is not-all-equal satisfiable.

4.3 Hardness of Maximizing the Cost

We now show that it is NP-hard to find a binary tree that maximizes cost(T).

THEOREM 10. *Given an instance ϕ of NAESAT*, we can in polynomial time specify a weighted graph $G = (V, E, w)$ and integer M such that*

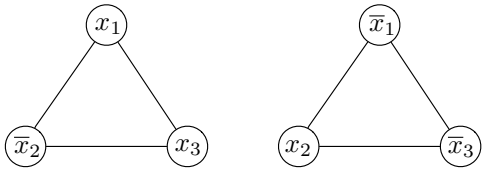
- $\max_{e\in E} w(e)$ *is polynomial in the size of ϕ, and*
- *ϕ is not-all-equal satisfiable if and only if there exists a binary tree T with $\text{cost}_G(T) \geq M$.*

PROOF. Suppose instance $\phi(x_1, \ldots, x_n)$ has m clauses of size three and m' clauses of size two. We will assume that certain redundancies are removed from ϕ:

- If there exists a 2-clause C whose literals also appear in a 3-clause C', then C' is redundant and can be removed.
- The same holds if the 3-clause C' contains the two literals of C with polarity reversed. This is because $(x \vee y)$ is not-all-equal satisfied if and only if $(\overline{x} \vee \overline{y})$ is not-all-equal satisfied.
- Likewise, if there is a 2-clause C whose literals also appear in a 2-clause C', but with reversed polarity, then C' can be removed.
- Any clause that contains both a literal and its negation can be removed.

Based on ϕ, construct a weighted graph $G = (V, E, w)$ with $2n$ vertices, one per literal (x_i and $\overline{x}_i$). The edges E fall into three categories:

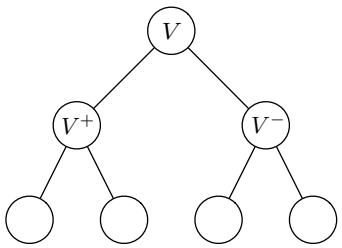
1. For each 3-clause, add six edges to E: one edge joining each pair of literals, and one edge joining the negations of those literals. These form two triangles. For instance, the clause $(x_1 \vee \overline{x}_2 \vee x_3)$ becomes:

   These edges have weight 1.
2. Similarly, for each 2-clause, add two edges of unit weight: one joining the two literals, one joining their negations.
3. Finally, add n edges $\{x_i, \overline{x}_i\}$ of weight $W = 2nm + 1$.

Under the definition of NAESAT*, and with the removal of redundancies, these $6m + 2m' + n$ edges are all distinct.

Now suppose ϕ is not-all-equal satisfiable. Let $V^+ \subset V$ be the positive literals under the satisfying assignment, and V^- the negative literals. Consider a two-level tree T:



The top split cuts all edges of G, except for one edge per triangle (there are $2m$ of these). The cost of this split is

$$|V| \cdot w(V^+, V^-) = 2n(4m + 2m' + nW).$$

By the symmetry of the construction, the remaining $2m$ edges are evenly split between V^+ and V^-. The m edges that lie entirely within V^+ contain at most one instance of each variable. Therefore, they are node-disjoint, and a single split suffices to cut all of them. The same holds for V^-. The total cost of the second level of splits is thus $|V^+|\cdot m+|V^-|\cdot m = 2nm$. In sum,

$$\begin{aligned}\text{cost}_G(T) &= 2n(4m + 2m' + nW) + 2nm \\ &= 10nm + 4nm' + 2n^2W.\end{aligned}$$

Call this M.

Conversely, suppose there exists a tree T of cost $\geq M$. The top split of T must cut all of the edges $\{x_i, \overline{x}_i\}$; if not, cost(T) would be at most $2n(6m + 2m' + nW) - W < M$. Therefore, the split divides V into two sets of size n, call them V^+ and V^-, such that each variable appears with one polarity in V^+ and the opposite polarity in V^-.

This top split $V \to (V^+, V^-)$ necessarily leaves at least one edge per triangle untouched. However, it must cut all other edges, otherwise the total cost would again fall below M. It follows that V^+ is a not-all-equal satisfying assignment for ϕ. ∎

5. AN APPROXIMATION ALGORITHM

Given a graph G with weighted edges, we have seen that it is NP-hard to find a tree that minimizes cost(T). We now consider top-down heuristics that begin by choosing a split $V \to (S, V \setminus S)$ according to some criterion, and then recurse on each half. What a suitable split criterion?

The cost of a split $(S, V \setminus S)$ is $|V| \cdot w(S, V \setminus S)$. Ideally, we would like to shrink the node set substantially in the process, since this reduces the multiplier on subsequent splits. For a node chosen at random, the expected amount by which its cluster shrinks as a result of the split is

$$\frac{|S|}{|V|} \cdot |V \setminus S| + \frac{|V \setminus S|}{|V|} \cdot |S| = \frac{2|S||V \setminus S|}{|V|}.$$

A natural greedy criterion would therefore be to choose the split that yields the maximum shrinkage per unit cost, or equivalently, the minimum ratio

$$\frac{w(S, V \setminus S)}{|S| \cdot |V \setminus S|}.$$

This is known as the *sparsest cut* and has been studied intensively in a wide range of contexts. Although it is NP-hard to find an optimal cut, a variety of good approximation algorithms have been developed [12, 13, 1, 20]. We will assume simply that we have a heuristic whose approximation ratio, on graphs of n nodes, is at most α_n times optimal, for some positive nondecreasing sequence (α_n). For instance, the Leighton-Rao algorithm [13] has $\alpha_n = O(\log n)$.

The resulting hierarchical clustering algorithm is shown in Figure 3. It is interesting that even though the cost function over hierarchies does not explicitly ask for sparse cuts (cost$_G(T)$ contains no ratios), the sparsest cut objective emerges organically when one adopts a greedy top-down approach to constructing the tree.

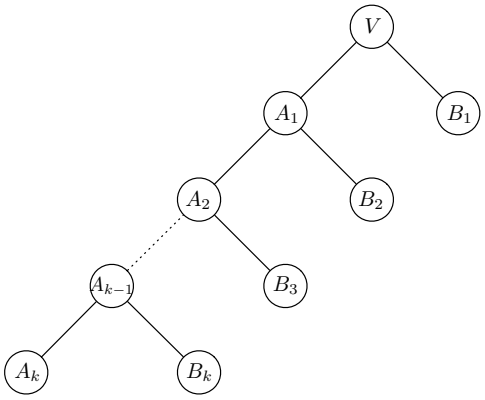
We will now see that this algorithm returns a tree of cost at most $O(\alpha_n \log n)$ times optimal. The first step is to show that if there exists a low cost tree, there must be a correspondingly sparse cut of V.

LEMMA 11. *Pick any binary tree T on V. There exists a partition A, B of V such that*

$$\frac{w(A,B)}{|A| \cdot |B|} < \frac{27}{4|V|^3}\, cost(T),$$

and for which $|V|/3 \leq |A|, |B| \leq 2|V|/3$.

PROOF. The diagram below shows a portion of T, drawn in a particular way. Each node u is labeled with the leaf set of its induced subtree $T[u]$. We always depict the smaller half of each cut (A_i, B_i) on the right, so that $|A_i| \geq |B_i|$.



Let $n = |V|$. Pick the smallest k so that $|B_1|+\cdots+|B_k| \geq n/3$. This also means $|A_{k-1}| > 2n/3$ (to deal with cases when $k = 1$, let $A_o = V$) and $|A_k| > n/3$. Define $A = A_k$ and $B = B_1 \cup \cdots \cup B_k$.

Now,

$$\begin{aligned}&\text{cost}(T) \\ &\geq w(A_1, B_1) \cdot n + w(A_2, B_2) \cdot |A_1| + \cdots + w(A_k, B_k) \cdot |A_{k-1}| \\ &> \frac{2n}{3}\left(w(A_1, B_1) + \cdots + w(A_k, B_k)\right) \\ &\geq \frac{2n}{3}\, w(A, B)\end{aligned}$$

since removing all the edges in the cuts $(A_1, B_1), \ldots, (A_k, B_k)$ disconnects A_k from $B_1 \cup \cdots \cup B_k$. Therefore,

$$\frac{w(A,B)}{|A| \cdot |B|} \;<\; \frac{3}{2n} \cdot \frac{\text{cost}(T)}{(2n/3) \cdot (n/3)} \;=\; \frac{27}{4n^3}\,\text{cost}(T),$$

as claimed. ∎

THEOREM 12. *Pick any graph G on n vertices, with positive edge weights $w : E \to \mathbb{R}^+$. Let tree T^* be a minimizer of cost$_G(\cdot)$ and let T be the tree returned by the top-down algorithm of Figure 3. Then*

$$cost_G(T) \;\leq\; (c_n \ln n)\, cost_G(T^*),$$

for $c_n = 27\alpha_n/4$.

PROOF. We'll use induction on n. The case $n = 1$ is trivial since any tree with one node has zero cost.

Assume the statement holds for graphs of up to $n - 1$ nodes. Now pick a weighted graph $G = (V, E, w)$ with n nodes, and let T^* be an optimal tree for it. By Lemma 11, the sparsest cut of G has ratio at most

$$\frac{27}{4n^3}\text{cost}(T^*).$$

```
function MakeTree(V)
If |V| = 1:  return leaf containing the singleton element in V
Let (S, V \ S) be an α_n-approximation to the sparsest cut of V
LeftTree = MakeTree(S)
RightTree = MakeTree(V \ S)
Return [LeftTree, RightTree]
```

Figure 3: A top-down heuristic for finding a hierarchical clustering that approximately minimizes cost(T).

The top-down algorithm identifies a partition (A, B) of V that is an α_n-approximation to this cut, so that

$$\frac{w(A,B)}{|A|\cdot|B|} \;\le\; \frac{27\alpha_n}{4n^3}\,\mathrm{cost}(T^*) \;=\; \frac{c_n}{n^3}\mathrm{cost}(T^*),$$

and then recurses on A and B.

We now obtain an upper bound on the cost of the best tree for A (and likewise B). Start with T^*, restrict all cuts to edges within A, and disregard subtrees that are disjoint from A. The resulting tree, call it T_A^*, has the same overall structure as T^*. Construct T_B^* similarly. Now, for any split $S \to (S_1, S_2)$ in T^* there are corresponding (possibly empty) splits in T_A^* and T_B^*. Moreover, the cut edges in T_A^* and T_B^* are disjoint subsets of those in T^*; hence the cost of this particular split in T_A^* and T_B^* combined is at most the cost in T^*. Formally,

$$\begin{aligned}
&(\text{cost of split in } T_A^*) + (\text{cost of split in } T_B^*)\\
&= |S\cap A|\cdot w(S_1\cap A, S_2\cap A) + |S\cap B|\cdot w(S_1\cap B, S_2\cap B)\\
&\le |S|\cdot w(S_1\cap A, S_2\cap A) + |S|\cdot w(S_1\cap B, S_2\cap B)\\
&\le |S|\cdot w(S_1,S_2) \;=\; \text{cost of split in } T^*.
\end{aligned}$$

Summing over all splits, we have

$$\mathrm{cost}(T_A^*) + \mathrm{cost}(T_B^*) \le \mathrm{cost}(T^*).$$

Here the costs on the left-hand side are with respect to the subgraphs of G induced by A and B, respectively.

Without loss of generality, $|A| = pn$ and $|B| = (1-p)n$, for some $0 < p < 1/2$. Recall that our algorithm recursively constructs trees, say T_A and T_B, for subsets A and B. Applying the inductive hypothesis to these trees, we have

$$\begin{aligned}
\mathrm{cost}(T_A) &\le c_n \mathrm{cost}(T_A^*)\ln pn\\
\mathrm{cost}(T_B) &\le c_n \mathrm{cost}(T_B^*)\ln(1-p)n
\end{aligned}$$

where we have used the monotonicity of (α_n), and hence

$$\begin{aligned}
&\mathrm{cost}(T)\\
&\le n\cdot w(A,B) + \mathrm{cost}(T_A) + \mathrm{cost}(T_B)\\
&\le n\cdot|A|\cdot|B|\cdot\frac{c_n}{n^3}\cdot\mathrm{cost}(T^*) + c_n\mathrm{cost}(T_A^*)\ln pn\\
&\quad + c_n\mathrm{cost}(T_B^*)\ln(1-p)n\\
&\le c_n p(1-p)\cdot\mathrm{cost}(T^*) + c_n\mathrm{cost}(T_A^*)\ln(1-p)n\\
&\quad + c_n\mathrm{cost}(T_B^*)\ln(1-p)n\\
&\le c_n p\cdot\mathrm{cost}(T^*) + c_n\mathrm{cost}(T^*)\ln(1-p)n\\
&= c_n\mathrm{cost}(T^*)\,(p + \ln(1-p) + \ln n)\\
&\le c_n\mathrm{cost}(T^*)\ln n,
\end{aligned}$$

as claimed. ∎

Is this approximation guarantee non-trivial? Recall the example of a graph composed of a single line of vertices (Section 3.1): there, we found that a good hierarchy has cost $O(n\log n)$ while a bad hierarchy has cost $\Omega(n^2)$. For sparse graphs of this kind, a polylogarithmic approximation guarantee is useful. For dense graphs, however, all clusterings have cost $\Omega(n^3)$, and thus the guarantee ceases to be meaningful.

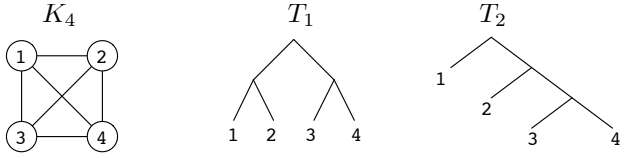
6. A GENERALIZED COST FUNCTION

A more general objective function for hierarchical clustering is

$$\mathrm{cost}_G(T) = \sum_{\{i,j\}\in E} w_{ij}\, f(|\texttt{leaves}(T[i\vee j])|),$$

where f is defined on the nonnegative reals, is strictly increasing, and has $f(0) = 0$. For instance, we could take $f(x) = \ln(1+x)$ or $f(x) = x^2$.

Under this generalized cost function, all the properties of Section 2 continue to hold, substituting $|S|$ by $f(|S|)$ as needed, for $S \subseteq V$. However, it is no longer the case that for the complete graph, all trees have equal cost. When G is the clique on four nodes, for instance, the two trees shown below (T_1 and T_2) need not have the same cost.



The tree costs, for any f, are:

$$\begin{aligned}
\mathrm{cost}(T_1) &= 4f(4) + 2f(2)\\
\mathrm{cost}(T_2) &= 3f(4) + 2f(3) + f(2)
\end{aligned}$$

Thus T_1 is preferable if f is concave (in which case $f(2) + f(4) \le 2f(3)$), while the tables are turned if f is convex.

Nevertheless, the greedy top-down heuristic continues to yield a provably good approximation. It needs to be modified slightly: the split chosen is (an α_n-approximation to) the minimizer $S \subseteq V$ of

$$\frac{w(S, V\setminus S)}{\min(f(|S|), f(|V\setminus S|))} \;\text{ subject to }\; \frac{1}{3}|V| \le |S| \le \frac{2}{3}|V|.$$

Lemma 11 and Theorem 12 need slight revision.

LEMMA 13. *Pick any binary tree T on V. There exists a partition A, B of V such that*

$$\frac{w(A,B)}{\min(f(|A|), f(|B|))} \le \frac{cost(T)}{f(\lfloor 2n/3\rfloor)f(\lceil n/3\rceil)}.$$

and for which $|V|/3 \le |A|, |B| \le 2|V|/3$.

PROOF. Sets A, B are constructed exactly as in the proof of Lemma 11. Then, referring to the diagram in that proof,

$$\begin{aligned}
&\text{cost}(T)\\
&\geq w(A_1, B_1) \cdot f(n) + w(A_2, B_2) \cdot f(|A_1|) + \cdots\\
&\quad + w(A_k, B_k) \cdot f(|A_{k-1}|)\\
&\geq f(\lfloor 2n/3 \rfloor)\,(w(A_1, B_1) + \cdots + w(A_k, B_k))\\
&\geq f(\lfloor 2n/3 \rfloor)\, w(A, B),
\end{aligned}$$

whereupon

$$\frac{w(A, B)}{\min(f(|A|), f(|B|))} \;\leq\; \frac{\text{cost}(T)}{f(\lfloor 2n/3 \rfloor) f(\lceil n/3 \rceil)}.$$

∎

THEOREM 14. *Pick any graph G on n vertices, with positive edge weights $w : E \to \mathbb{R}^+$. Let tree T^* be a minimizer of cost$(\cdot)$ and let T be the tree returned by the modified top-down algorithm. Then*

$$cost_G(T) \leq (c_n \ln n)\, cost_G(T^*),$$

where

$$c_n = 3\alpha_n \cdot \max_{1 \leq n' \leq n} \frac{f(n')}{f(\lceil n'/3 \rceil)}.$$

PROOF. The proof outline is as before. We assume the statement holds for graphs of up to $n-1$ nodes. Now pick a weighted graph $G = (V, E, w)$ with n nodes, and let T^* be an optimal tree for it.

Let T denote the tree returned by the top-down procedure. By Lemma 13, the top split $V \to (A, B)$ satisfies $n/3 \leq |A|, |B| \leq 2n/3$ and

$$\begin{aligned}
w(A, B) &\leq \alpha_n \min(f(|A|), f(|B|))\, \frac{\text{cost}(T^*)}{f(\lfloor 2n/3 \rfloor) f(\lceil n/3 \rceil)}\\
&\leq \frac{\alpha_n}{f(\lceil n/3 \rceil)} \text{cost}(T^*),
\end{aligned}$$

where the last inequality uses the monotonicity of f.

As before, the optimal tree T^* can be used to construct trees T_A^* and T_B^* for point sets A and B, respectively, such that $\text{cost}(T_A^*) + \text{cost}(T_B^*) \leq \text{cost}(T^*)$. Meanwhile, the top-down algorithm finds trees T_A and T_B which, by the inductive hypothesis, satisfy

$$\begin{aligned}
\text{cost}(T_A) &\leq (c_n \ln |A|)\text{cost}(T_A^*)\\
\text{cost}(T_B) &\leq (c_n \ln |B|)\text{cost}(T_B^*)
\end{aligned}$$

Let's say, without loss of generality, that $|A| \leq |B|$. Then

$$\begin{aligned}
&\text{cost}(T)\\
&\leq f(n) \cdot w(A, B) + \text{cost}(T_A) + \text{cost}(T_B)\\
&\leq f(n) \cdot w(A, B) + (c_n \ln |B|)(\text{cost}(T_A^*) + \text{cost}(T_B^*))\\
&\leq \frac{\alpha_n f(n)}{f(\lceil n/3 \rceil)} \text{cost}(T^*) + (c_n \ln n + c_n \ln(2/3))\text{cost}(T^*)\\
&\leq \frac{c_n}{3} \text{cost}(T^*) + (c_n \ln n - c_n/3)\text{cost}(T^*)\\
&= (c_n \ln n)\text{cost}(T^*),
\end{aligned}$$

as claimed. ∎

7. ACKNOWLEDGEMENTS

The author is grateful to the National Science Foundation for support under grant IIS-1162581.

8. REFERENCES

[1] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the Association for Computing Machinery*, 56(2), 2009.

[2] R. Boppana. Eigenvalues and graph bisection: an average-case analysis. In *IEEE Symposium on Foundations of Computer Science*, 1985.

[3] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. *SIAM Journal on Computing*, 33(6):1417–1440, 2004.

[4] K. Chaudhuri, S. Dasgupta, S. Kpotufe, and U. von Luxburg. Consistent procedures for cluster tree estimation and pruning. *IEEE Transactions on Information Theory*, 60(12):7900–7912, 2014.

[5] S. Dasgupta and P. Long. Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences*, 70(4):555–569, 2005.

[6] J. Eldridge, M. Belkin, and Y. Wang. Beyond Hartigan consistency: merge distortion metric for hierarchical clustering. In *28th Annual Conference on Learning Theory*, 2015.

[7] J. Felsenstein. *Inferring Phylogenies*. Sinauer, 2004.

[8] J. Hartigan. Statistical theory in clustering. *Journal of Classification*, 2:63–76, 1985.

[9] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, 2009.

[10] N. Jardine and R. Sibson. *Mathematical Taxonomy*. John Wiley, 1971.

[11] R. Kannan, S. Vempala, and A. Vetta. On clusterings: good, bad, and spectral. *Journal of the ACM*, 51(3):497–515, 2004.

[12] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, 1970.

[13] F. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the Association for Computing Machinery*, 46(6):787–832, 1999.

[14] G. Lin, C. Nagarajan, R. Rajaraman, and D. Williamson. A general approach for incremental approximation and hierarchical clustering. *SIAM Journal on Computing*, 39:3633–3669, 2010.

[15] C. McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, 141:148–188, 1989.

[16] F. McSherry. Spectral partitioning of random graphs. In *IEEE Symposium on Foundations of Computer Science*, pages 529–537, 2001.

[17] R. Neal. Density modeling and clustering using Dirichlet diffusion trees. In J. Bernardo et al., editors, *Bayesian Statistics 7*, pages 619–629. Oxford University Press, 2003.

[18] C. Plaxton. Approximation algorithms for hierarchical location problems. *Journal of Computer and System Sciences*, 72:425–443, 2006.

[19] R. Sokal and P. Sneath. *Numerical Taxonomy*. W.H. Freeman, 1963.

[20] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.