**Approximation Bounds for Hierarchical Clustering: Average Linkage, Bisecting K-means, and Local Search**

**Benjamin Moseley** moseleyb@andrew.cmu.edu

Carnegie Mellon University Pittsburgh, PA 15213, USA

**Joshua R. Wang** joshuawang@google.com

Google Research

1600 Amphitheatre Parkway

Mountain View, CA 94043, USA

**Editor:** Vahab Mirrokni

**Abstract**

Hierarchical clustering is a data analysis method that has been used for decades. Despite its widespread use, the method has an underdeveloped analytical foundation. Having a well understood foundation would both support the currently used methods and help guide future improvements. The goal of this paper is to give an analytic framework to better understand observations seen in practice. This paper considers the dual of a problem framework for hierarchical clustering introduced by [Dasgupta](#bookmark2) [(2016)](#bookmark2). The main result is that one of the most popular algorithms used in practice, average linkage agglomerative clustering, has a small constant approximation ratio for this objective. To contrast, this paper establishes that using several other popular algorithms, including bisecting k-means divisive clustering, have a very poor lower bound on its approximation ratio for the same objective. However, we show that there are divisive algorithms that perform well with respect to this objective by giving two constant approximation algorithms. This paper is some of the ﬁrst work to establish guarantees on widely used hierarchical algorithms for a natural objective function. This objective and analysis give insight into what these popular algorithms are optimizing and when they will perform well.

**1. Introduction**

Hierarchical clustering is a widely used method to analyze data. See [Murtagh and Contreras](#bookmark3) [(2012);](#bookmark3) [Krishnamurthy et al.](#bookmark4) [(2012);](#bookmark4) [Heller and Ghahramani](#bookmark5) [(2005) for an overview and](#bookmark5) pointers to relevant work. In a typical hierarchical clustering problem, one is given a set of n data points and a notion of similarity between the points. The output is a hierarchy of clusters on the input. Speciﬁcally, adendrogram (tree) is constructed where the leaves correspond to the n input data points and the root corresponds to a cluster containing all data points. Each internal node of the tree corresponds to a cluster of the data points in its subtree. The clusters (internal nodes) become more reﬁned as we movedown the tree. The goal is to construct the tree so that these deeper clusters contain points that are relatively more similar.

©2023 Benjamin Moseley and Joshua R. Wang.

License: CC-BY 4.0, see [https://creativecommons.org/licenses/by/4.0/.](https://creativecommons.org/licenses/by/4.0/) Attribution requirements are provided at [http://jmlr.org/papers/v24/18-080.html.](http://jmlr.org/papers/v24/18-080.html)

Moseley and Wang

There are many reasons for the popularity of hierarchical clustering, including that the number of clusters is not predetermined and that the clusters produced induce taxonomies that give meaningful ways to interpret data.

Methods used to perform hierarchical clustering are divided into two classes: agglomerative and divisive. **Agglomerative** algorithms take a bottom-up approach and are more commonly [used than divisive approaches (Hastie et al.,](#bookmark6) [2009)](#bookmark6). In an agglomerative method, each of the n input data points starts as its own cluster. Then iteratively, pairs of similar clusters are merged according to some appropriate notion of similarity. Perhaps the most popular deﬁnition of similarity is **average linkage** where the similarity between two clusters is deﬁned as the average similarity between all pairs of data points in the two clusters. In average linkage agglomerative clustering the two clusters with the highest average similarity are merged at each step. Other variants are also popular. Related examples include: **single linkage**, where the similarity between two clusters is the maximum similarity between any pair of single data points indiﬀerent clusters, and **complete linkage**, where the distance is the minimum similarity between any pair of single data points indiﬀerent clusters.

**Divisive** algorithms take a top-down approach where initially all data points areplaced into a single cluster. They recursively perform splits, dividing a cluster into smaller clusters that will be further subdivided. The process continues until each cluster consists of a single data point. In each step of the algorithm, the data points are partitioned such that points within each cluster are more similar than points across clusters. There are several approaches to perform divisive clustering. One example is bisecting k-means where k-means is used at each step with k = 2. For details on bisecting k-means, see [Jain](#bookmark7) [(2010)](#bookmark7).

**Motivation:** Hierarchical clustering has been used and studied for decades. There has been some work on theoretically quantifying the quality of the solutions produced by algorithms, such as [Ackerman et al.](#bookmark8) [(2012);](#bookmark8) [Ackerman and Ben-David](#bookmark9) [(2016);](#bookmark9) [Zadeh and Ben-David](#bookmark10) [(2009);](#bookmark10) [Ben-David and Ackerman](#bookmark11) [(2008);](#bookmark11) [Dasgupta](#bookmark2) [(2016)](#bookmark2). Much of this work focuses on deriving the structure of solutions created by algorithms or analytically describing desirable properties of a clustering algorithm. Though the area has been well-studied, there is no widely accepted formal problem framework. Hierarchical clustering describes a class of algorithmic methods rather than a problem with an objective function. Studying a formal objective for the problem could lead to the ability to objectively compare diﬀerent methods; there is a desire for the community to investigate potential objectives, which would further support the use of current methods and guide the development of improvements.

This paper is concerned with investigating objectives for hierarchical clustering. It gives a natural objective and leverages its structural connection to average linkage agglomerative clustering to prove this algorithm obtains a constant approximation to the best possible clustering. In contrast to this positive result, single linkage, complete linkage, and bisecting k-means are shown to have superconstant (i.e. scaling with the number of data points) approximation ratios. This paper also provides some divisive algorithms that have comparable theoretical guarantees to average linkage.

**Problem Formulation:** Towards this paper’s goal, we begin by trying to establish a formal problem framework for hierarchical clustering. Recently, [Dasgupta](#bookmark2) [(2016) introduced a new](#bookmark2) problem framework for hierarchical clustering. This work justiﬁedits proposed objective by establishing that for several sample problem instances, the resulting solution corresponds to

Approximation Bounds for Hierarchical Clustering

what one might expect out of a desirable solution. This work spurred considerable interest [and there have been several follow up papers (Charikar and Chatziafratis,](#bookmark12) [2017;](#bookmark12) [Dasgupta,](#bookmark2) [2016;](#bookmark2) [Roy and Pokutta,](#bookmark13) [2016)](#bookmark13).

In the problem introduced by [Dasgupta](#bookmark2) [(2016), the input is a set of](#bookmark2) n data points as input along with, for every pair of points i and j, a (nonnegative) weight wi,j denoting their similarity. The higher the weight, the more similar the edge endpoints. This can be represented as a weighted complete graph G. The output is a (full) binary tree where the leaves of the tree correspond to the input data points. For each pair of points i and j, let T [i \_j] denote the subtree rooted at i and j’s least common ancestor. Let leaves(T [i \_j]) denote the set of leaves in the tree T [i \_ j]. The goal is to construct the tree such that the cost cost G(T) :=εi,j∈[n]wijj leaves(T [i \_j])j is minimized. Intuitively, this objective enforces that more similar points i and j should have a lower least common ancestor in the tree because the weight wi,j is large and so additions to j leaves(T [i \_j])j are severely penalized.

[For this objective, several approximation algorithms have been given (Charikar and](#bookmark12) [Chatziafratis,](#bookmark12) [2017;](#bookmark12) [Dasgupta,](#bookmark2) [2016;](#bookmark2) [Roy and Pokutta,](#bookmark13) [2016)](#bookmark13). There is a divisive clustering algorithm with an approximation ratio of O(√log n) [(Charikar and Chatziafratis,](#bookmark12) [2017)](#bookmark12). In particular, the algorithm gives an O(αn)-approximation where αn is the approximation ratio [of the sparsest cut subroutine (Charikar and Chatziafratis,](#bookmark12) [2017)](#bookmark12). Furthermore, assuming the Small-Set Expansion Hypothesis, every algorithm is a [ω(1)-approximation (Charikar](#bookmark12) [and Chatziafratis,](#bookmark12) [2017)](#bookmark12). The current best known bound on αn is O(√log n) [(Arora et al.,](#bookmark14) [2009)](#bookmark14). Unfortunately, this conclusion misses one of our key goals in trying to establish an objective function. While the algorithms and analysis are ingenious, none of the algorithms with theoretical guarantees are from the class of algorithms used in practice. Furthermore, in this work we establish that several commonly used algorithms (namely, single linkage, average linkage, complete linkage, and bisecting k-means) have bad negative bounds on their solution quality with respect to Dasgupta’s cost objective.

Hence this question still looms: are there strong theoretical guarantees for practical algorithms? It seems that Dasgupta’s cost objective may not be ideal for this task; is there another natural objective that admits solutions that are provably close to optimal?

**Results:** In this paper, we consider an objective function motivated by the objective introduced by [Dasgupta](#bookmark2) [(2016)](#bookmark2). For a given tree T let j leaves-outside (T [i \_ j])j be the total number of leaves that are not in the subtree rooted at the least common an- cestor of i and j. The Dasgupta objective considers constructing a tree T to minimize the cost function cost G(T) := εi,j∈[n]wijj leaves(T [i \_ j])j . This paper considers the dual problem where T is constructed to maximize the reward function rewardG (T) := εi,j∈[n]wijj leaves-outside (T [i \_ j])j = (nεi,j∈[n]wi,j) - cost G(T). It is important to observe that the optimal clustering is the same for both objectives. Due to this, all the examples given in [Dasgupta](#bookmark2) [(2016) motivating their objective by showing desirable structural](#bookmark2) properties of the optimal solution also apply to the objective considered in this paper. Our objective can be interpreted similarly to that in [Dasgupta](#bookmark2) [(2016)](#bookmark2). In particular, similar points i and j should be located lower in the tree as to maximize j leaves-outside(T [i \_j])j .

This paper gives a thorough investigation of this new problem framework by analyzing several algorithms under this objective. The main result is establishing that average linkage  . This result gives theoretical justiﬁcation for the use of

Moseley and Wang

average linkage clustering and, additionally, this shows that the objective considered is tractable since it admits Ω(1)-approximations. It also suggests that the objective captures a component of what average linkage optimizes for.

This paper then seeks to understand what other algorithms are good for this objective. We establish that single linkage, complete linkage, and bisecting k-means is no better than a polynomial-approximations. This establishes that these methods are very poor for the objective considered, and that these methods optimize for something distinct than what average linkage optimizes for.

Given these negative results, we question whether there are divisive algorithms that optimize for our objective. We answer this question aﬃrmatively by giving a local search 

 .

**Related Work (This Cost Function):** [Charikar et al.](#bookmark15) [(2019a) gave a tight lower bound](#bookmark15) construction for average linkage under this paper’s reward objective, which matches our upper bound. They same work also gives an SDP relaxation technique which has a better approximation ratio. [Charikar et al.](#bookmark16) [(2019b) study our objective for Euclidean data and](#bookmark16) shows a better bound for average linkage under this data model. [Menon et al.](#bookmark17) [(2019) consider](#bookmark17) our objective in the setting where the data points arrive one by one.

**Related Work (Other Cost Functions):** [Recently a contemporaneous paper (Cohen-](#bookmark18) [Addad et al.,](#bookmark18) [2017) done independently has been published](#bookmark18). This paper considers a class of objectives motivated by the work of [Dasgupta](#bookmark2) [(2016)](#bookmark2). For their objective, they also derive positive results for average linkage clustering and additionally give axiomatic properties that are desirable in an objective for hierarchical clustering. [Ma and Dhavala](#bookmark19) [(2018) consider](#bookmark19) combining Dasgupta’s objective function with prior knowledge about the data set. [Wang and](#bookmark20) [Wang](#bookmark20) [(2018) suggests an alternate objective with the goal of comparing the clusterability](#bookmark20) across diﬀerent input graphs, rather than just diﬀerent clusterings for a single input graph. Both [Chierchia and Perret](#bookmark21) [(2019) and](#bookmark21) [Monath et al.](#bookmark22) [(2019) propose continuous objective](#bookmark22) functions with the goal of applying gradient descent. [Lattanzi et al.](#bookmark23) [(2019) and](#bookmark23) [Abboud](#bookmark24) [et al.](#bookmark24) [(2019) both strive to make more eﬃcient (parallelized and raw runtime, respectively)](#bookmark24) versions of classical clustering algorithms, and judge the resulting quality via comparing iteration-by-iteration against what the classical algorithm would have done. [Wang and](#bookmark25) [Moseley](#bookmark25) [(2020) propose an objective function for which bisecting k-means achieves a constant](#bookmark25) approximation.

**2. Preliminaries**

In this section, we give preliminaries including a formal deﬁnition of the problem considered

and basic building blocks for later algorithm analysis.

In the **Reward Hierarchical Clustering Problem** there aren input data points given as a set V. There is a weight wi;j ≥ 0 between each pair of points i and j denoting their similarity, represented as a complete graph G. The output of the problem is a rooted tree T where the leaves correspond to the data points and the internal nodes of the tree correspond to clusters of the points in the subtree. We will use the indices 1, 2, . . . n to denote the leaves of the tree. For two leaves i and j, let T [i \_j] denote the subtree rooted at the least

Approximation Bounds for Hierarchical Clustering

common ancestor of i and j and let the set leaves-outside (T [i ∨ j]) denote the number of leaves in T that are not in T [i ∨ j]. The objective is to construct T to maximize the reward rewardG (T) =εi∈[n] εji∈[n]wi,j| leaves-outside (T [i ∨ j])| .

We make no assumptions on the structure of the optimal tree T. However, one optimal tree is a binary tree, so we may restrict the solution to binary trees without loss of generality. This can be seen as follows. It was shown in [Dasgupta](#bookmark2) [(2016) that the optimal solution](#bookmark2) for the cost G(T) is binary. Knowing that cost G(T) + rewardG (T) = nεi,j wi,j, so the optimal solution to minimizing cost G(T) is the same as the optimal solution to maximizing rewardG (T). Thus, there is an optimal solution for the rewardG (T) objective that is binary.

To see this, let leaves(T [i ∨ j]) be the set of leaves in T [i ∨ j] and cost G(T) := εi,jwij|leaves(T [i ∨ j])| . The objective considered in [Dasgupta](#bookmark2) [(2016) focuses on minimiz](#bookmark2)- ing cost G(T). We note than cost G(T) + rewardG (T) = nεi,j wi,j, so the optimal solution to minimizing cost G(T) is the same as the optimal solution to maximizing rewardG (T). In [Dasgupta](#bookmark2) [(2016) it was shown that the optimal solution for any input is a binary tree](#bookmark2).

As mentioned, there are two common types of algorithms for hierarchical clustering: ag- glomerative (bottom-up) algorithms and divisive (top-down) algorithms. In an agglomerative algorithm, each vertex v ∈ V begins in separate cluster, and each iteration of the algorithm chooses two clusters to merge into one. In a divisive algorithm, all vertices v ∈ V begin in a single cluster, and each iteration of the algorithm selects a cluster with more than one vertex and partitions it into two small clusters.

In this section, we present some basic techniques which we later use to analyze the eﬀect each iteration has on the reward. It will be convenient for us to think of the weight function as taking in two vertices instead of an edge, i.e. w : V × V → R≥0 . This is without loss of generality, because we can always set the weight of any non-edge to zero (e.g. wvv = 0 ∀v ∈ V).

To bound the performance of an algorithm itsuﬃces to bound rewardG (T) and cost G(T) since rewardG (T) + cost G(T) = nεi,j wi,j . Further, let T\* denote the optimal hierarchical clustering. Then its reward is at most rewardG (T\* ) ≤ (n — 2) εij wij . This is because any edge ij can have at most (n — 2) non-leaves for its subtree T [i ∨ j]; i and j are always leaves.

**2.1 Analyzing Agglomerative Algorithms**

In this section, we discuss a method for bounding the performance of an agglomerative algorithm. When an agglomerative algorithm merges two clusters A,B, this determines the least common ancestor for any pair of nodes i,j where i ∈ A and j ∈ B . Given this, we deﬁne the reward gain due to merging A and B as, merge-rewG (A, B) := (n — |A| — |B|) εa∈A,b∈B wab.

Notice that the ﬁnal reward rewardG (T) is exactly the sum over iterations of the reward gains, since each edge is counted exactly once: when its endpoints are merged into a single cluster. Hence, rewardG (T) =εmerges A,B merge-rewG (A, B).

We next deﬁne the cost of merging A and B as the following. This is the potential reward lost by merging A and B ; reward that can no longer be gained after A and B are

m|erged, but was initially possible. Deﬁne, merge-cost G(A, B) := |B| εa∈A,c∈[n]\(AUB) wac +

A| εb∈B,c∈[n]\(AUB) wbc.

Moseley and Wang

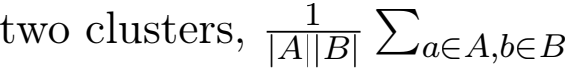
The total cost of the tree T , cost G(T), is exactly the sum over iterations of the cost increases, plus an additional 2 εij wij term that accounts for each edge being counted towards its own endpoints. We can see why this is true if we consider a pair of vertices i, j ∈ [n] in the ﬁnal hierarchical clustering T. If at some point a cluster containing i is merged with a third cluster before it gets merged with the cluster containing j, then the number of leaves in T [i ∨ j] goes up by the size of the third cluster. This is exactly the quantity captured by our cost increase deﬁnition. Aggregated overall pairs i,j this is the following, cost G(T) =εi,j∈[n]wij|leaves(T [i ∨ j])| = 2 εi,j∈[n]wij +εmerges A,B merge-cost G(A, B).

**2.2 Analyzing Divisive Algorithms**

Similar reasoning can be used for divisive algorithms. The following are reward gain and cost increase deﬁnitions for when a divisive algorithm partitions a cluster into two clusters A,B . Deﬁne, split-rewG (A, B) := |B| εa,a, ∈A waa, + |A| εb,b, ∈B wbb, and split-cost G(A, B) := (|A| + |B|) εa∈A,b∈B wab.

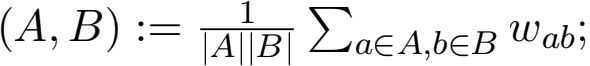
Consider the reward gain. For a, a/ ∈ A we are now guaranteed that when the nodes in B are split from A then every node in B will not be a leaf in T [a ∨ a/] (and asymmetric term for when they are both in B). On the cost side, the term counts the cost of any pairs a ∈ A and b ∈ B that are now separated since we now know their subtree T [i ∨ j] has exactly the nodes in A ∪ B as leaves.

**3. A Theoretical Guarantee for Average Linkage Agglomerative Clustering**

In this section, we present the main result, a theoretical guarantee on average linkage clustering. See [Murtagh and Contreras](#bookmark3) [(2012) for details and background on this widely](#bookmark3) used algorithm. The formal deﬁnition of the algorithm is given in the following pseudocode. The main idea is that initially all n input points are in their own cluster and the algorithm recursively merges clusters until there is one cluster. In each step, the algorithm merges the clusters A and B such that the pair maximizes the average distances of points between the  wab.

**Data:** Vertices V , weights w : E → R≥0 Initialize clusters C ← ∪v∈V{v};

**while** |C| ≥ 2 **do**

Choose A, B ∈ C to maximize w-  Set C ← C ∪ {A ∪ B} \ {A, B};

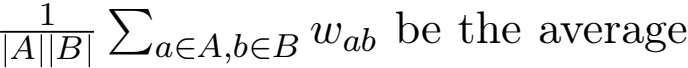
**end**

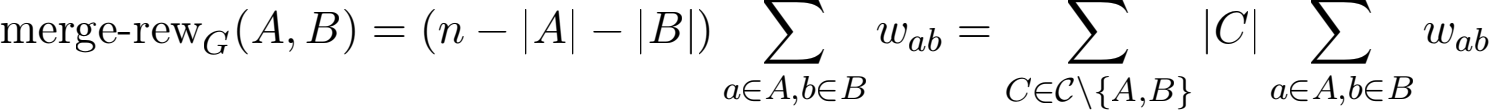
**Algorithm 1:** Average Linkage

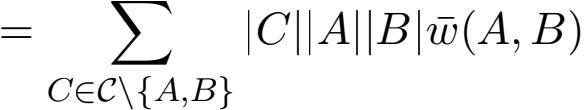
The following theorem establishes that this algorithm is only a small constant factor away from optimal.

**Theorem 1** Consider a graph G = (V, E) with nonnegative edge weights w : E → R≥0 . Let the hierarchical clustering T\* be an optimal solution maximizing of rewardG (·) and let T be the hierarchical clustering returned by Algorithm [1](#bookmark26). Then, rewardG (T) ≥ G (T\* ) .

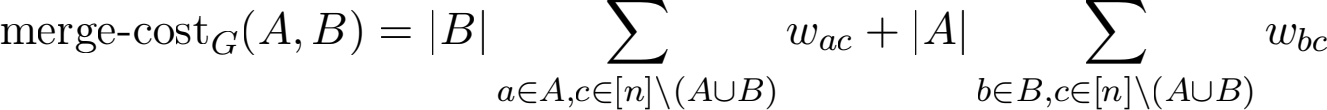
Approximation Bounds for Hierarchical Clustering

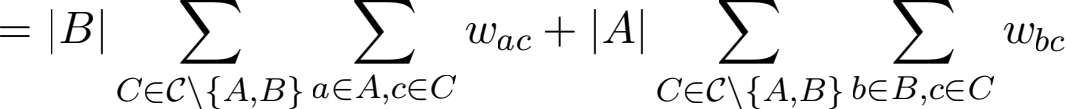
**Proof** Consider an iteration of Algorithm [1.](#bookmark26) Let the current clusters be in the set C , and the algorithm chooses to merge clusters A and B from C. When doing so, the algorithm attains a reward gain of the following. Let w- (A, B) =  weight of an edge between points in A and B .

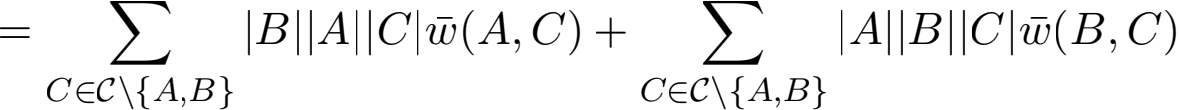


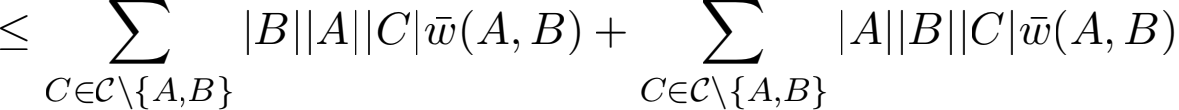


while at the sametime incurring a cost increase of:



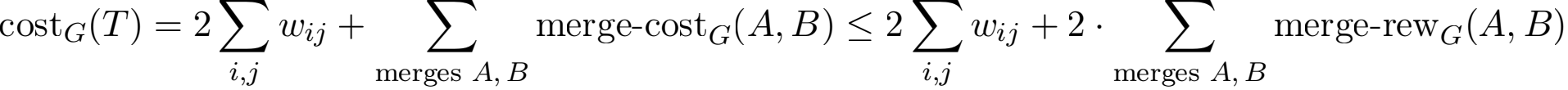


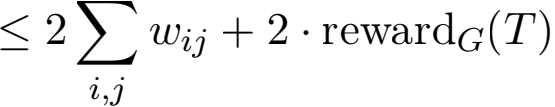




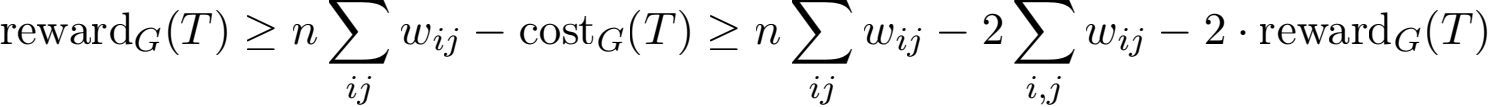
= 2 · merge-rewG (A, B)

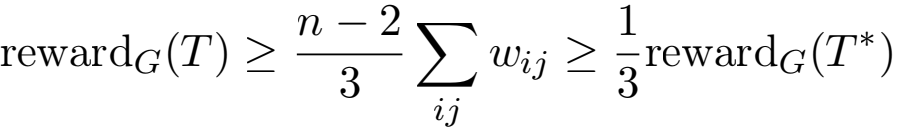
Intuitively, every time this algorithm loses two units of potential it cements the gain of  . Formally:





Now the reward can be bounded as follows.





where the last step follows from the fact that it is impossible to have more than n — 2 leaves-outside. 

As an aside, we note that it is not possible for all inequalities in the proof to be simultaneously tight, and hence average linkage is actually slightly better (inverse polynomial in n) than a 1/3 approximation. The reasoning is as follows.

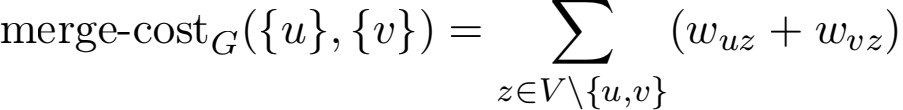
Consider the very ﬁrst merge made by average-linkage, where it picks the highest weight edge of our graph G. Suppose this edge (u, v) has weight W. The proof above relates

Moseley and Wang

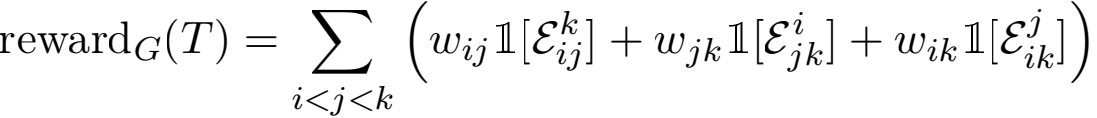
merge-cost G({u}, {v}) and merge-rewG ({u}, {v}). We plug known quantities into the deﬁni- tion of merge reward.

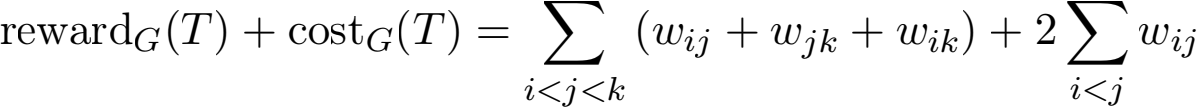
merge-rewG ({u}, {v}) = (n — 2)W

We can also unroll the deﬁnition of merge cost for this simple merge.



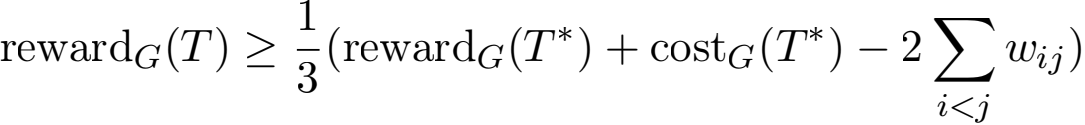
We know from our proof that the merge cost is between zero and 2(n — 2)W. Let’s express the cost as α(n — 2)W, where α ∈ [0, 2]. We handle the case where α is high as follows. The average triangle containing (u, v) has (α + 1)W total weight. Our reward objective has an alternate form in terms of triangles, as observed by [Charikar et al.](#bookmark16) [(2019b)](#bookmark16).

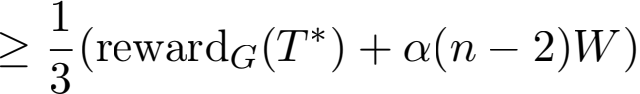




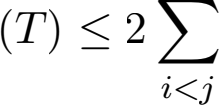
Following the notation in that paper, εk denotes the event that among the three nodes

{i,j, k}, i is split from the others ﬁrst. For the average triangle containing (u, v), this means that even the optimal hierarchical clustering can only score W out of a possible (α + 1)W total. We can combine this missing αW value for each of our (n — 2) triangles with the ﬁnal approximation statement to get:

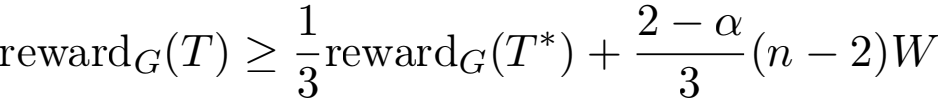


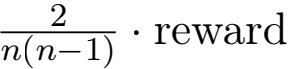


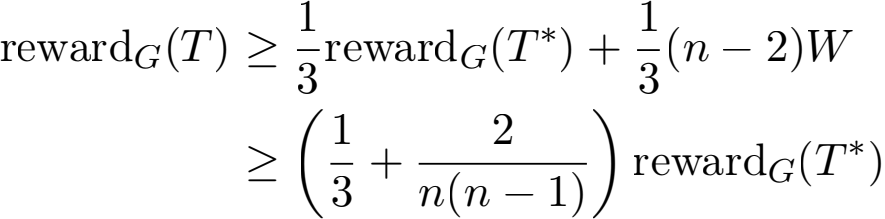
Now, we handle the case where α is low. In this case, we gain (n — 2)W merge reward while only incurring α(n — 2)W merge cost. This puts a little ahead in our telescoping

inequality: cost G wij + 2rewardG 

Hence we arrive at a slightly better conclusion:



We conclude by noting that our two guarantees are balanced when α = 1 and that (n — 2)W ≥ G (T\* ).



Approximation Bounds for Hierarchical Clustering

**4. Divisive Local-Search**

In this section, we develop a simple local search algorithm and bound its approximation ratio. The local search algorithm takes as input a cluster C and divides it into two clusters A and B to optimize a local objective: the split reward. In particular, initially A = B = ∅. Each node in C is added to A or B uniformly at random.

Local search is run by moving individual nodes between A and B . In a step, any point i ∈ A (resp. B ) is added to B (resp. A) if Σj,l∈A;j,liwj,l + (|A| — 1) Σj∈B wi,j >

A| Σj∈B,ji wi,j). This states that a point is moved to another set if the objective increases. The algorithm performs these local moves until there is no node that can be moved to improve the objective.

Σ|j,l∈B wj,l + |B| Σj∈A,ji wi,j (resp. Σj,l∈B;j,liwj,l + (|B| — 1) Σj∈A wi,j >Σj,l∈A wj,l +

**Data:** Vertices V , weights w : E → R≥0 Initialize clusters C ← {V };

**while** some cluster C ∈ C has more than one vertex **do**

Let A,B be a uniformly random 2-partition of C;

Run local search on A,B to maximize |B| Σa,a, ∈A waa, + |A| Σb,b, ∈B wbb, , considering just moving a single node;

Set C ← C ∪ {A, B} \ {C};

**end**

**Algorithm 2:** Divisive Local-Search

- mation.

**Theorem 2** Consider a graph G = (V, E) with nonnegative edge weights w : E → R≥0 . Let the hierarchical clustering T\* be the optimal solution of rewardG (·) and let T be the

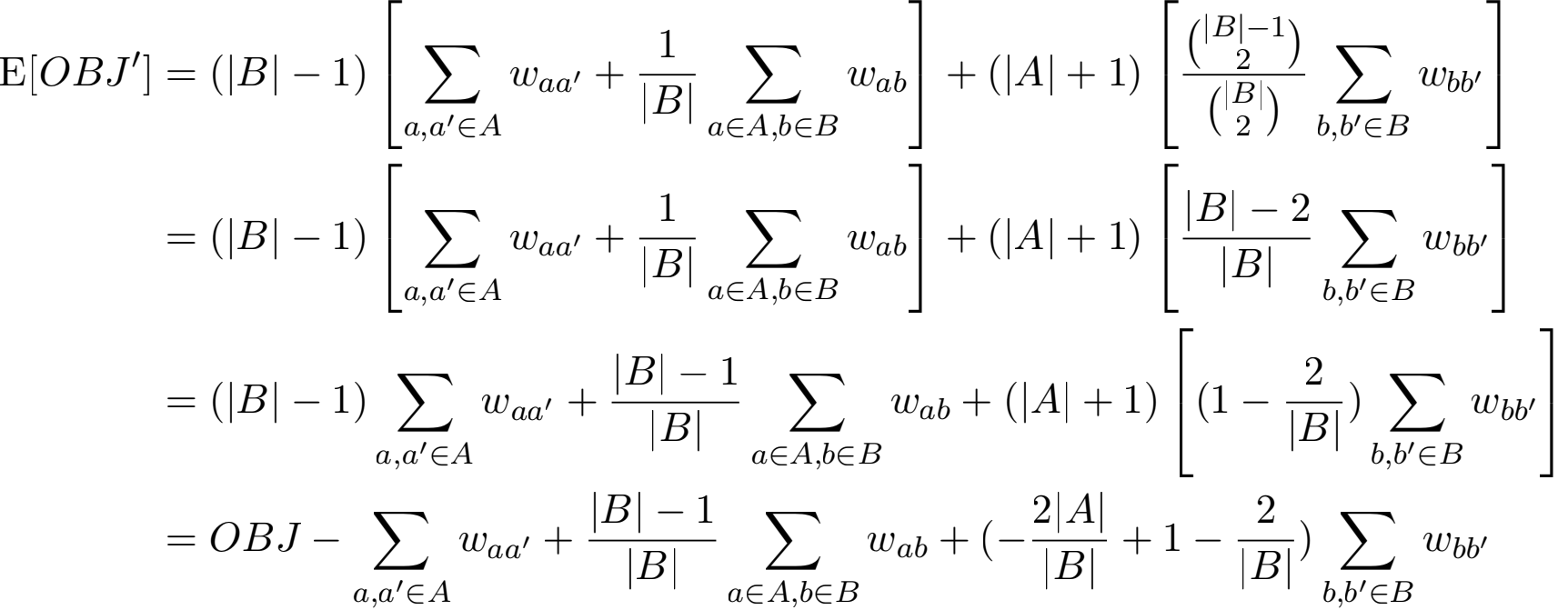
hierarchical clustering returned by Algorithm [2](#bookmark27). Then, rewardG (T) ≥  G (T\* ) .

**Proof** Since we know that rewardG (T\* ) ≤ (n — 2) Σij wij, it suﬃces to show that rewardG (T) ≥  wij . We do this by considering possible local moves at ev- ery step.

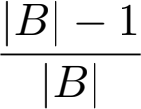
Consider any step of the algorithm and suppose the algorithm decides to partition a cluster into A,B . As stated in the algorithm, its local search objective value is OBJ = |B| Σa,a, ∈A waa, + |A| Σb,b, ∈B wbb, . Assume without loss of generality that |B| ≥ |A|, and consider the expected local search objective OBJ/ value for moving a random node from B to A. Note that the new local search objective value is at most what the algorithm obtained,

Moseley and Wang

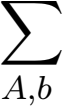
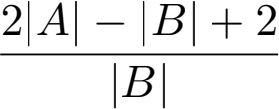
i.e. OBJI ≤ OBJ :



But since there are no improving moves we know the following.

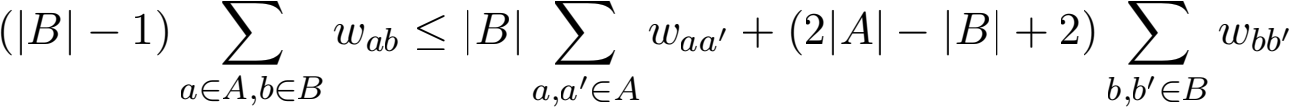
0 ≥ E[OBJI] - OBJ = - Σ waa, + 

a,a, ∈A

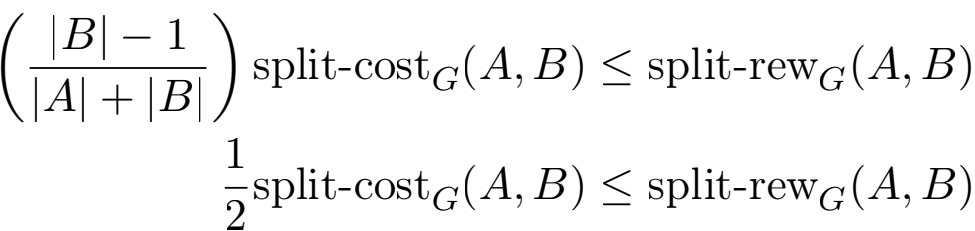
a∈∈B wab - 

Σ wbb, b,b, ∈B

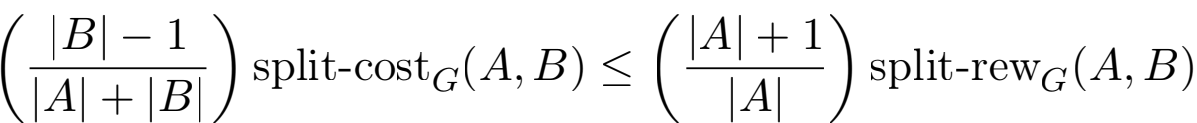
Rearranging terms and multiplying by jBj yields the following.

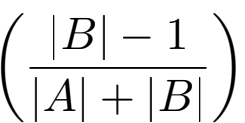
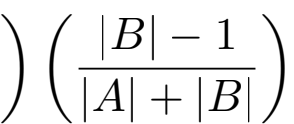


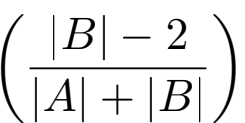
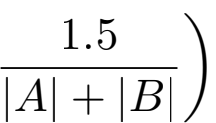
We now consider three cases. Either (i) jBj ≥ jAj + 2, (ii) jBj = jAj + 1, or (iii) jBj = jAj . Case (i) is straightforward:



In case (ii), we use the fact that (x + 2)(x - 2) ≤ (x + 1)(x - 1) to simplify:

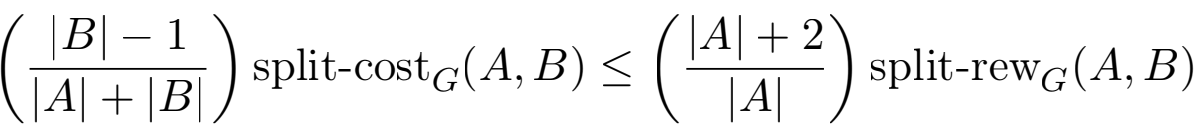


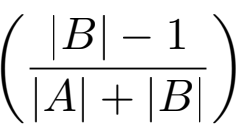
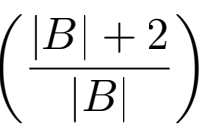
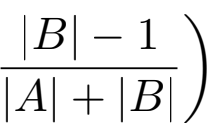
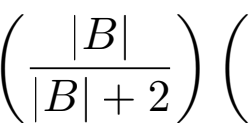
 split-cost G(A, B) ≤ ( ) split-rewG (A, B) (  split-cost G(A, B) ≤ split-rewG (A, B)

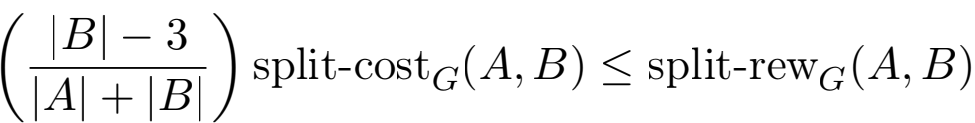
 split-cost G(A, B) ≤ split-rewG (A, B)  -  split-cost G(A, B) ≤ split-rewG (A, B)

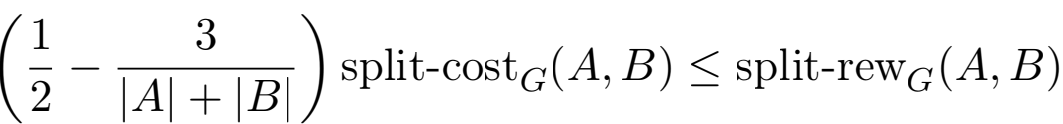
Approximation Bounds for Hierarchical Clustering

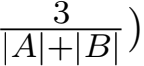
Case (iii) proceeds similarly; we now use the fact that (x + 2)(x — 3) ≤ (x)(x — 1) to simplify:

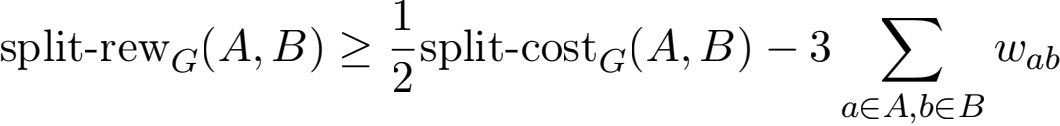


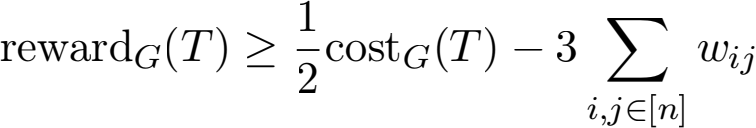
 split-cost G(A, B) ≤  split-rewG (A, B)  split-cost G(A, B) ≤ split-rewG (A, B)

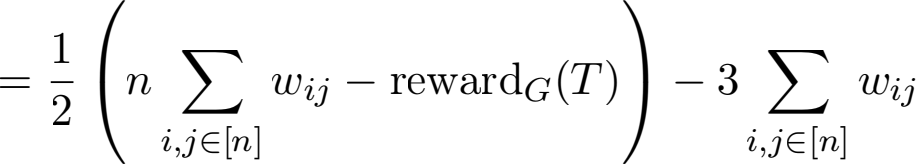


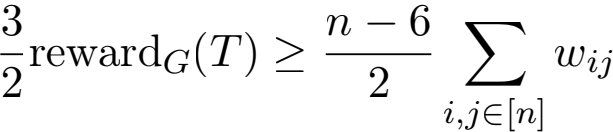


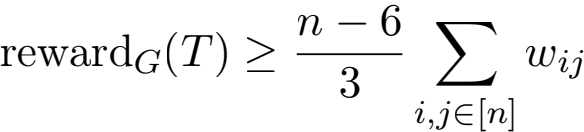
Hence we have shown that for each step of our algorithm, the split reward is at least  —  times the split cost. Werewrite this inequality and then sum over alliterations:











This is what we wanted to

prove.

We note that it is possible to improve the loss in terms of n to 

the local search objective (jBj — 1) Σa,a, ∈A waa, + (jAj — 1) Σb,b, ∈B wbb, .

**5. Random Hierarchical Clustering**

In this section, we bound the performance of a random divisive algorithm. In each step, the algorithm is given acluster and divides the points into two clusters A and B where a point is added in each step uniformly at random. We show that this algorithm is a  to our reward function and further this is tight.

**Theorem 3** Consider a graph G = (V, E) with nonnegative edge weights w : E → R≥0 . Let the hierarchical clustering T\* be a maximizer of rewardG (·) and let T be the hierarchical

Moseley and Wang

**Data:** Vertices V , weights w : E → R≥0 Initialize clusters C ← {V };

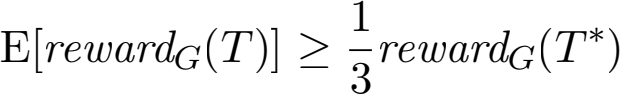
**while** some cluster C ∈ C has more than one vertex **do**

Let A,B be a uniformly random 2-partition of C; Set C ← C ∪ {A, B} \ {C};

**end**

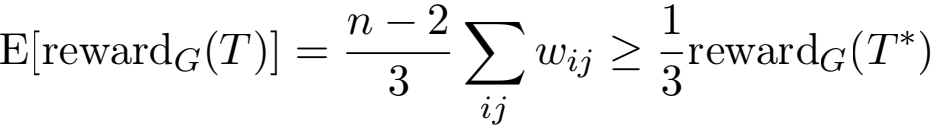
**Algorithm 3:** Random Hierarchical Clustering

clustering returned by Algorithm [3](#bookmark29). Then:



**Proof** We begin by pretending that A or B empty is a valid partition of C , and address this detail at the end of the proof. If so, we can generate A,B with the following random process: for each vertex v ∈ C , ﬂipa fair coin to decide if it goes into A or into B .

Now, consider an edge (i, j) ∈ E. The algorithm will score a reward of wij|leaves-outside (T [i ∨ j])| . Thus, we need to determine the expected value of | leaves-outside (T [i ∨ j])| . How often does one of the n — 2 other nodes besides i and j become a nonleaf of T [i ∨ j]? Fix all all coin ﬂips made for i and let k  i,j be a point. The point k will become a nonleaf if j matches more coin ﬂips than k does. The number of matched coin ﬂips is a geometric random variable with parameter 1/2. There is a 1/2 chance of matching for zero coin ﬂips, a 1/4 chance of matching for one coin ﬂip,and soon. Let h be the number of edges on the path from i to the root of the tree. The probability of equality is Σ  · · · = 1/3 for suﬃcientlylarge h. By symmetry, the remaining 2/3 probability is split bewteen j matching for more and k matching for more. Hence each of the other n — 2 nodes k has exactly a 1/3 chance of being a nonleaf. As a result,

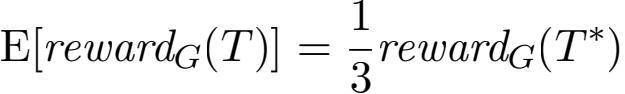


since it is impossible to have more than n — 2 nonleaves.

Finally, we address the possibility of A or B being empty. This is equivalent to anode in T having a single child. In this case, rewardG (T) is unchanged if we merge the node with that child, since this does not change leaves(T [i ∨ j]) for any edge (i, j). Hence if A or B is empty we can safely redraw. Hence our random process is equivalent to uniformly drawing overall partitions. This completes the proof. 

We now establish that this is tight.

**Lemma 4** There exists a graph G = (V, E) with nonnegative edge weights w : E → R≥0 , such that if the hierarchical clustering T\* is an optimal solution of rewardG (·) and T is the hierarchical clustering returned by Algorithm [3,](#bookmark29)

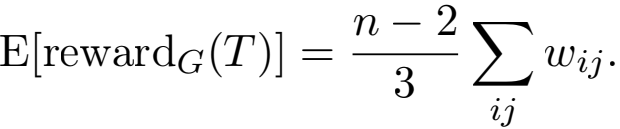


Approximation Bounds for Hierarchical Clustering

|  |  |  |
| --- | --- | --- |
| **Algorithm** | This paper’s Reward Objective | Dasgupta’s Cost Objective |
| Single Linkage | Flail:  [Thm. [10]](#bookmark30) | Line: Ω( ) [Thm. [11]](#bookmark31) |
| Average Linkage | Clique Star: (1/2 + δ) [Thm. [12]](#bookmark32) | Clique Star: Ω(n1/3) [Thm. [13]](#bookmark33) |
| Complete Linkage | Double Star:  [Thm. [14]](#bookmark34) | Line: Ω( ) [Thm. [15]](#bookmark35) |
| Bisecting k-Means | Double Star:  [Thm. [16]](#bookmark36) | Cycle Star: Ω( √n) [Thm. [17]](#bookmark37) |

Table 1: The base counterexample graphs we use and the approximation bounds they yield.

**Proof** In the proof of Lemma [3, we showed that](#bookmark28)



This naturally suggests a tight example: any graph where the optimal hierarchical clustering T\* can capture all edges (i, j) ∈ E with non-zero weight using only clusters of size 2. In other words, in any graph where the edges form a matching, the bound is tight. 

**6. Negative Results**

This section gives negative lower bound proofs for the algorithms single linkage, average linkage, complete linkage, and bisecting k-means. All of the proofs in this section follow a similar structure:

1. Construct a base counterexample graph.

2. Increase several edge weights in the base graph by multiples of a constant ∈ > 0 in order to ensure the algorithm breaks-ties in favor of the wrong decision.

3. Upper-bound the quality of the algorithm’s hierarchical clustering, T.

4. The performance of the optimal solution is lower-bounded by giving a hierarchical clustering solution T/ . This solution will be far superior to the algorithm’ssolution.

In steps (3) and (4), it will typically be the case that the analysis breaks edge weights into a base component and a tie-breaking component, which come from steps (1) and (2) respectively.

We begin by deﬁning all of the base counterexample graphs that will be used. Then the negative results for each of the algorithms are given in the following subsections.

**6.1 Base Counterexample Graphs**

This subsection gives deﬁnitions for the base counterexample graphs that will be used in the lower bound proofs.

**Deﬁnition 5 (Line Graph. See Figure** [**1.)**](#bookmark38)The line graph Pn, consists of n nodes: v1 , . . . , vn . The only edges are in the path (v1 , v2 , . . . , v n) . This graph is parameterized by n.

Moseley and Wang

v1

v2

v3

v4

v5

Figure 1: The unweighted line graph on 5 nodes.

v1

v4

v2

v5

v3

|  |
| --- |
| Line |

|  |
| --- |
| v6  v7  v8  v9  v10  v11  v12 |

|  |
| --- |
| Clique |

Figure 2: The unweighted (n = 12, k = 5)-ﬂail graph.

**Deﬁnition 6 (Flail Graph. See Figure** [**2)**](#bookmark39)The (n, k)-ﬂail graph consists of n nodes: v1 , . . . , vn . The ﬁrst k nodes, v1 , . . . , vk, form a clique. The only other edges are those in the path vk, v k+1 , . . . , vn . This graph is parameterized by n and k.

**Deﬁnition 7 (Double Star Graph. See Figure** [**3)**](#bookmark40)The k-double star graph consists of 2l nodes: v1 , . . . , v2k . There are edges from v1 to all of v2 , v3 , . . . , vl, forming a star. There are also edges from vl+1 to all of vl+2, vl+3 , . . . , v2l, forming another star. Finally, there is an edge from v1 to vl+1, connecting the two stars. This graph is parameterized by l, the size of each star.

**Deﬁnition 8 (Cycle Star Graph. See Figure** [**4)**](#bookmark41)The (k, l)-cycle star graph consists of k + kl nodes. Let U = {v1 , . . . , v k} and Ui = {vi;1, . . . , vi;l} for i ∈ [1, . . . , k] . The vertex

set V = U [ u=1 Ui . There are edges from vi to vi+1 and vk to v1 , forming a cycle. There

are also edges from vi to all of Ui, for each i ∈ [1, . . . , k], forming a star. This graph is parameterized by k, the nodes on the cycle, and l, the leaves in each of the k stars.

**Deﬁnition 9 (Clique Star Graph. See Figure** [**5)**](#bookmark42)The (k, l)-clique star graph consists of k + kl nodes. Let U = {v1 , . . . , v k} and Ui = {vi;1, . . . , vi;l} for i ∈ [1, . . . , k] . The vertex

set V = U [ u=1 Ui . The nodes in U form a clique. There are also edges from vi to all of

Ui, for each i ∈ [1, . . . , k], forming a star. This graph is parameterized by k, the nodes inthe clique, and l, the leaves in each of the k stars.

v7

v8

v9

v6

v10

|  |
| --- |
| Star |

|  |
| --- |
| Star |

|  |
| --- |
| v2  v3  v5  v4  v1 |

Figure 3: The unweighted (l = 5)-double star graph.

Approximation Bounds for Hierarchical Clustering

Star

v1;3

v1;2

v1;1

v2;1

v5;3

v2;2

v2;3

|  |
| --- |
| v1  v2 一  v5  v3 /  v4 |

v5;2

v3;1

v5;1

|  |
| --- |
| Cycle |

\

/

v3;2

v3;3

v4;3

v4;1

v4;2

Figure 4: The unweighted (k = 5; l = 3)-cycle star graph.

Star

v1;3

v1;2

v1;1

v2;1

v5;3

v2;2

v2;3

v1

v2

v5;2

v5

v3;1

v3

v5;1

v4

|  |
| --- |
| Clique |

\

/

v3;2

v3;3

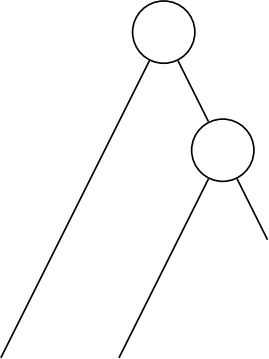
v4;3

v4;1

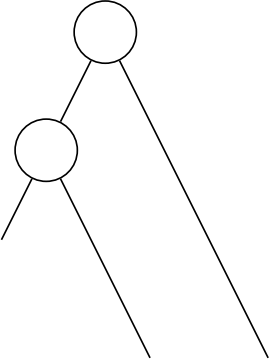
v4;2

Figure 5: The unweighted (k = 5; l = 3)-clique star graph.

Moseley and Wang

Tree T

v1 v2 v3 v4

Tree TI

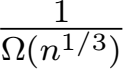
v1 v2 v3 v4

Figure 6: Hierarchical clusterings from the proof of Theorem [10, for](#bookmark30) n = 4. Single-linkage produces the left clustering, while the right clustering achieves a better objective value.

**6.2 Single Linkage and the Reward Objective**

This subsection gives a lower-bound proof showing that single linkage does not produce a good approximation to the reward objective introduced in this paper.

**Theorem 10** There exists a family of graphs Gn with nonnegative edge weights w : E → R≥0 such that if the hierarchical clustering T\* is a maximizer of rewardG (·) and T is the hierarchical

clustering returned by single linkage, rewardG (T) ≤  · rewardG (T\* ) .

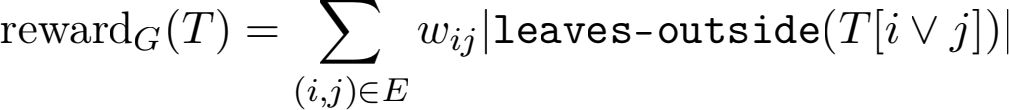
**Proof** For convenience, it is assumed that n isa perfect cube. If n is not, then the graph can be padded with unconnected vertices.

The base counterexample graph family is the ﬂail graph; see Figure [2](#bookmark39) for an example. For the actual counterexample graph, several weights for tie-breaking purposes will be increased. Let ∈ > 0 be a constant that will be ﬁxed later. Choose Gn to be the (n, n2/3)-ﬂail graph. The weights w are as follows. All edges are unit weight, except edges between consecutive vertices: the edge (vi, vi+1) has weight (1 + i∈). In other words, the most valuable edge is (v n — 1 , v n), followed by (v n —2 , v n — 1 ), and so on all the way down to (v1 , v2 ). The least valuable edges are the clique edges between nonconsecutive vertices.

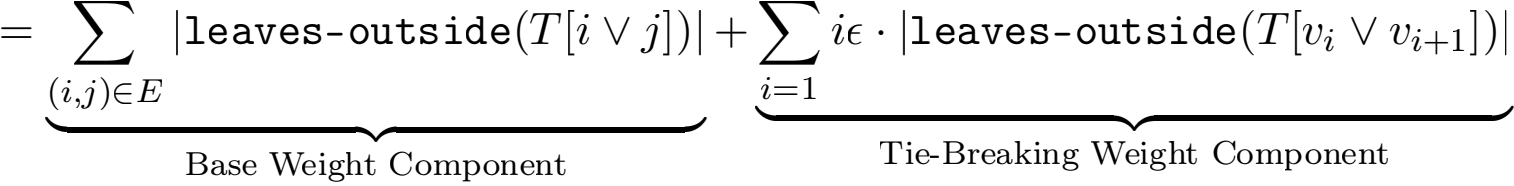
Consider the clustering created by the single linkage algorithm on Gn. Recall that single linkage merges clusters in each step based on the most valuable edge between two clusters. By construction, it must take the cluster {v n} is merged with v n — 1 . The resulting cluster is merged with v n —2 , and so on, until the cluster merges with v1 . This clustering is depicted in Figure [6.](#bookmark43) Let T be the tree corresponding to this hierarchical clustering. The reward given in this tree is bounded as follows. To compute the reward, we ﬁrst split each edge into a unit weight component and a tie-breaking component. Then the contribution from clique edges

Approximation Bounds for Hierarchical Clustering

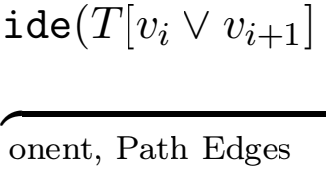
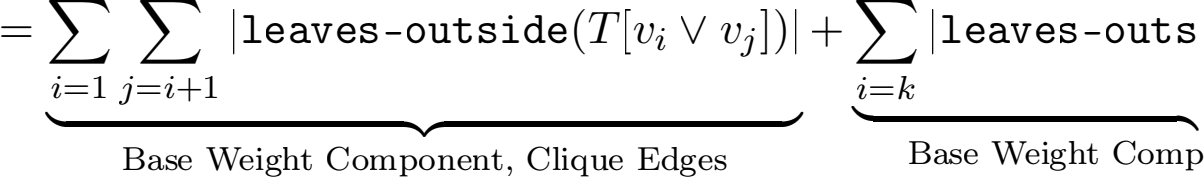
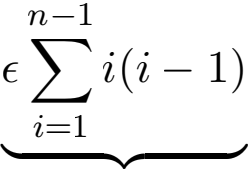
and path edge are accounted for separately.



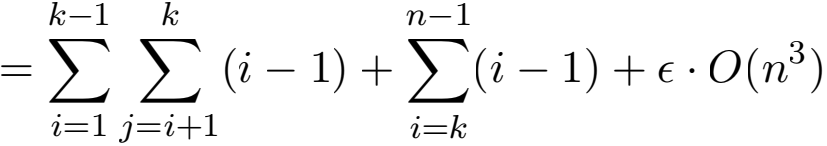
n-1



k-1 k n-1

)–j + 

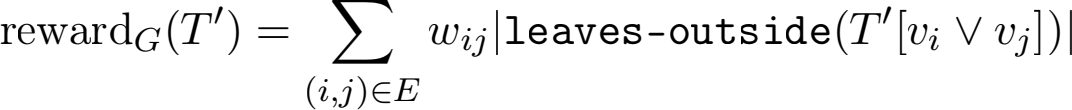




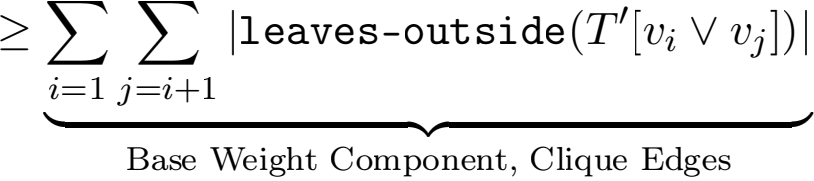
= O(k3 ) + O(n2 ) + ∈ · O(n3 )

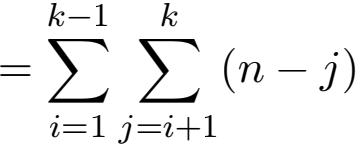
Fix ∈ to be 1/n. Since k = n2/3, this entire expression is O(n2 ).

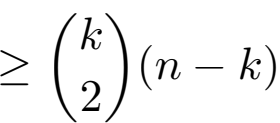
To ﬁnish the proof, it suﬃces to show that some hierarchical clustering obtains at least Ω(n7/3) reward on this graph. Consider the hierarchical clustering which does things in the opposite direction; T/ takes the cluster {v1 } and merges in v2 , then v3 , and so on, until it merges in vn. This clustering is also depicted in Figure [6.](#bookmark43) How much reward does T/ accumulate? A similar calculation as before bounds this as follows:



k-1 k



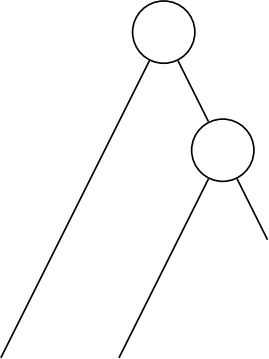




Since k = n2/3, this is at least Ω((n2/3)2 · n) = Ω(n7/3), as desired. This establishes that some hierarchical clustering, and hence also the optimal hierarchical clustering, beats single

linkage in terms of reward by a factor of at least Ω(n1/3). This completes the proof. 

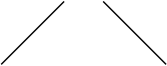
Moseley and Wang

Tree T

v1 v2 v3 v4

Tree TI









v1 v2 v3 v4

Figure 7: Hierarchical clusterings from the proof of Theorem [11, for](#bookmark31) n = 4. Single-linkage produces the left clustering, while the right clustering achieves a better objective value.

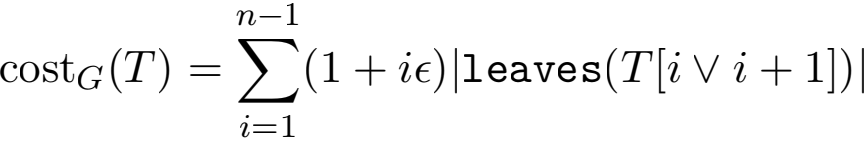
**6.3 Single Linkage and the Cost Objective**

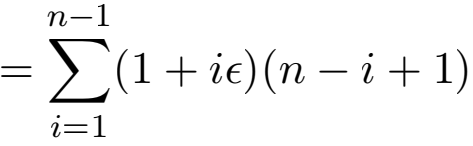
In this subsection, it is shown that the algorithm single linkage does not produce a good approximation to Dasgupta’s cost objective.

**Theorem 11** There exists a family of graphs Gn with nonnegative edge weights w : E → R≥0 such that if the hierarchical clustering T\* is a minimizer of costG (·) and T is the hierarchical clustering returned by single linkage, costG (T) ≥ Ω() · costG (T\* ) .

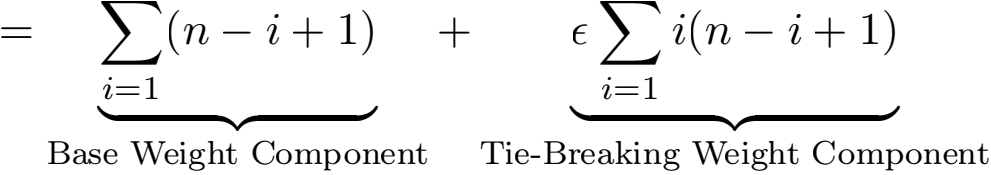
**Proof** The proof uses the line graph as the base counterexample graph family; see Figure [1](#bookmark38) for an example. For the actual counterexample graph, several weights in the line graph are increased for tie-breaking purposes. Let ∈ > 0 be a constant that will be set later. Choose Gn to be Pn. The weights of Gn areas follows. The edge (vi, vi+1) has weight (1 + i∈). In other words, the most valuable edge is (v n — 1 , v n), followed by (v n —2 , v n — 1 ), and soon all the way down to (v1 , v2 ).

Consider what the single linkage algorithm does on Gn. Recall that single linkage only considers the most valuable edge between two clusters. By construction, it must take the cluster {v n} and merge in v n — 1 , then v n —2 , and so on, until it merges in v1 . This clustering is depicted in Figure [7.](#bookmark44) Given that this is how the algorithm behaves, the question is how much cost does this hierarchical clustering T accumulate. This is bounded as follows.





n—1 n—1

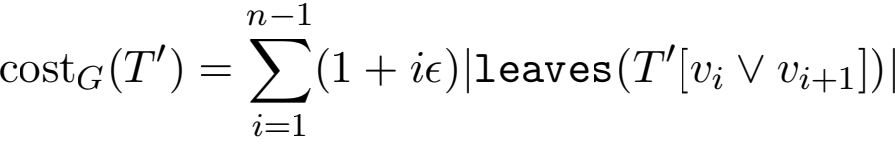


≥ Ω(n2 )

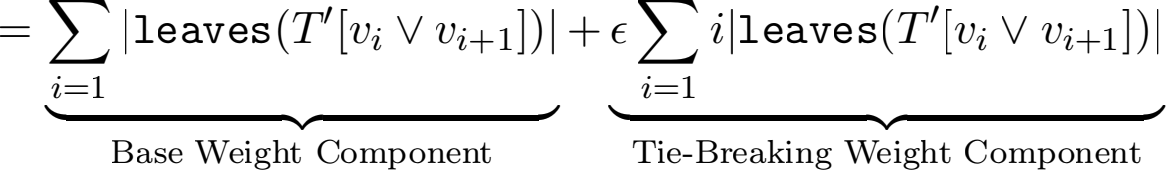
Approximation Bounds for Hierarchical Clustering

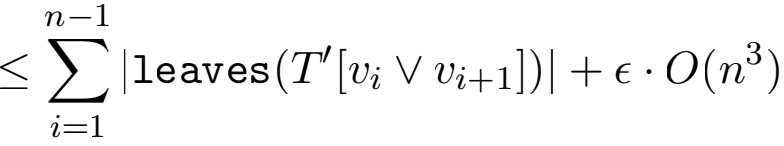
To ﬁnish the proof, it suﬃces to show that some hierarchical clustering obtains at most O(nlog n) cost on this graph. This was essentially shown in Dasgupta’s original paper, but this is presented here as well for the sake of completeness [Dasgupta](#bookmark2) [(2016)](#bookmark2).

Consider the following hierarchical clustering TI . The construction begins with each node in its own cluster: C1 = {v1 }, C2 = {v2 }, . . . , Cn = {v n}. Then perform a merge step by pairing up each odd numbered cluster with the next largest numbered cluster. This yields new clusters C1 = {v1 , v2 }, C2 = {v3 , v4 }, . . .. The edges between the resulting clusters from a line graph. The process is repeated until there is a single cluster left. This clustering is also depicted in Figure [7.](#bookmark44) The total cost of this clustering TI is bounded as follows.



n—1 n—1





Choose ∈ = 1/n2 so that the second term is O(n). To bound the ﬁrst term, notice that every time a merge step is performed, at most n cost is incurred; there is a single edge between every pair of merged clusters and all the clusters being merged are disjoint. There are at most O(log n) merge steps performed, since each step reduces the number of clusters by a factor of 2. Hence cost G(TI ) ≤ O(nlog n), as desired. Together this shows that some hierarchical clustering, and hence also the optimal hierarchical clustering, beats single linkage in terms of cost by a factor of at least Ω(n/ log n). This completes the proof. 

**6.4 Average Linkage and the Reward Objective**

In this subsection, it is shown that the average linkage algorithm is at best a constant- approximation to the reward objective.

**Theorem 12** Let δ > 0 be any constant. There exists a family of graphs Gn with nonnegative edge weights w : E → R≥0 such that if the hierarchical clustering T\* is a maximizer of rewardG (·) and T is the hierarchical clustering returned by average linkage, rewardG (T) ≤  · rewardG (T\* ), as long as n is suﬃciently large.

**Proof** Let k be a constant integer (that depends only on the constant δ) that we will select later. For convenience, assume that the number of nodes n is a multiple of k. If n is not, then the graph can be padded with unconnected vertices.

The base counterexample graph family is the clique star graph; see Figure [5](#bookmark42) for an example. For the actual counterexample graph, several weights will be increased for tie- breaking purposes. Let ∈ > 0 be a constant that will be ﬁxed later. Choose Gn to be the

Moseley and Wang

tree T (clique ﬁrst)

v k

v k;l

v1;1 · · ·

v1 · · ·

tree TI (stars ﬁrst)

· · ·

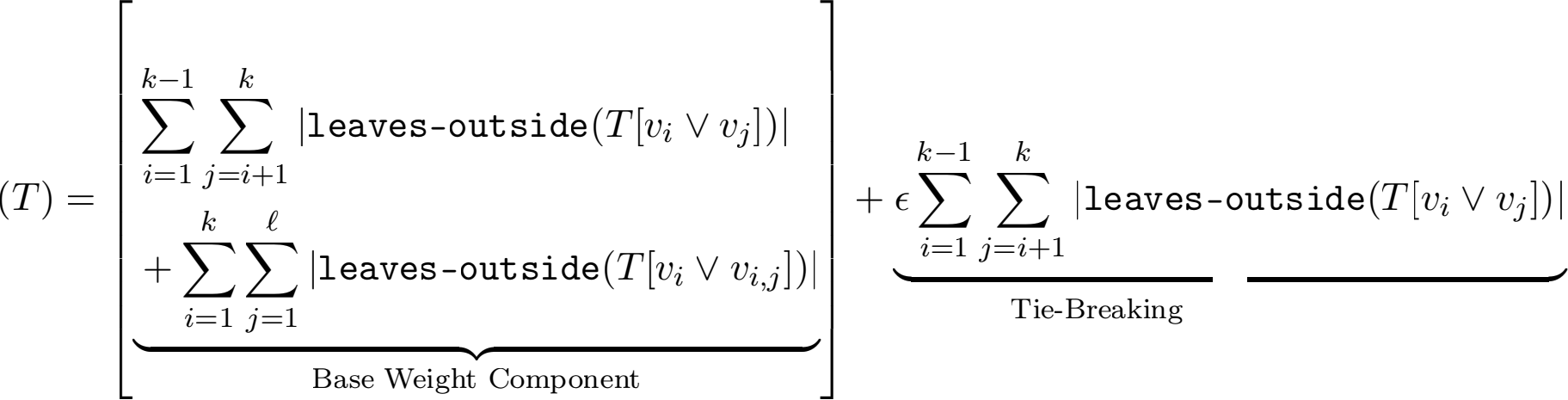
v1 v1;1 · · · v1;l · · · vk v k;1 · · · v k;l

Figure 8: Hierarchical clusterings from the proofs of Theorem [12](#bookmark32) and Theorem [13.](#bookmark33) Average- linkage produces the top clustering, while the bottom clustering achieves a better objective value. Solid edges denote direct children, while dotted edges denote indirect children.

Approximation Bounds for Hierarchical Clustering

(k, (n/k) — 1)-clique star graph. The weights w are as follows. All edges are unit weight, except the clique edges which have weight (1 + ∈) instead.

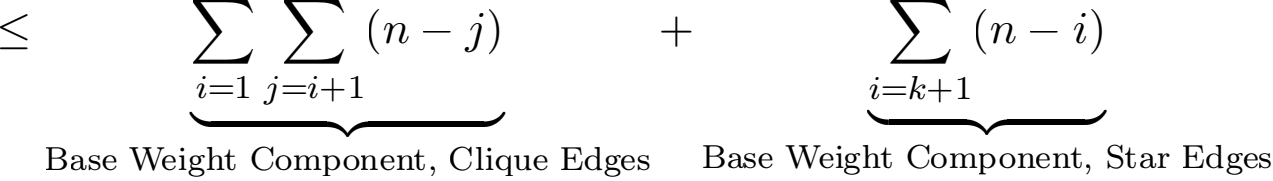
Consider what average linkage does on Gn. Average linkage begins by merging together the clique of weight (1 + ∈). After this, the remaining graph is a large star, so average linkage ﬁnishes by merging in one node at a time into the center. This clustering is depicted in Figure [8.](#bookmark45) How much reward does this hierarchical clustering T accumulate?

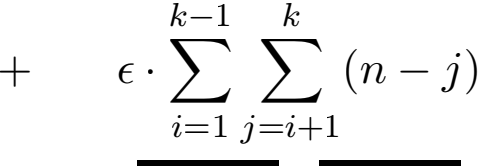
rewardG 

、◆

Weight Component

k—1 k n

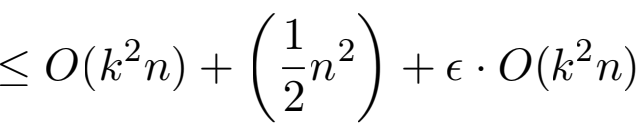




、 –

、◆

Tie-Breaking Weight Component

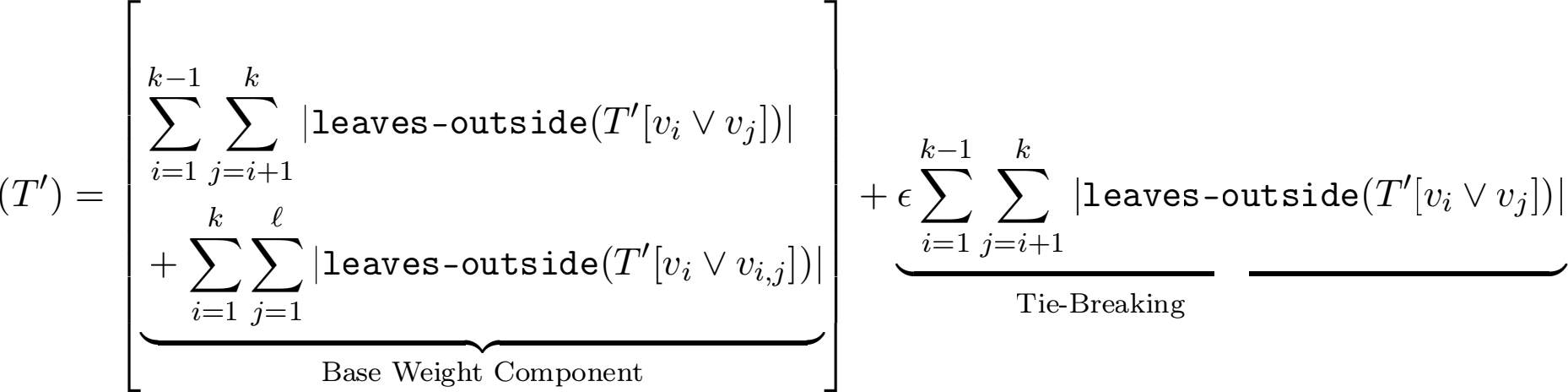


We now choose ∈ = 1 so that the RHS expression is 2 + O(k2n). To ﬁnish the proof, it will suﬃce to show that some hierarchical clustering obtains nearly n2 reward on this graph.

Consider the hierarchical clustering which ﬁrst handles the small stars (by merging in one node at a time into the center), and then merges together the remaining clique. This

Moseley and Wang

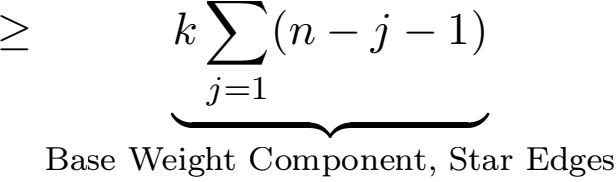
clustering is also depicted in Figure [8.](#bookmark45) How much reward does T, accumulate?

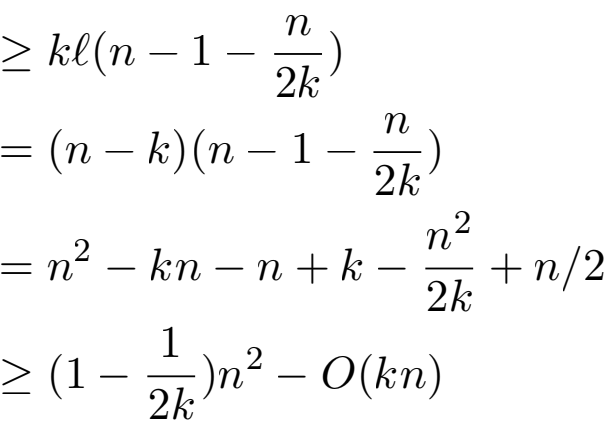
rewardG 

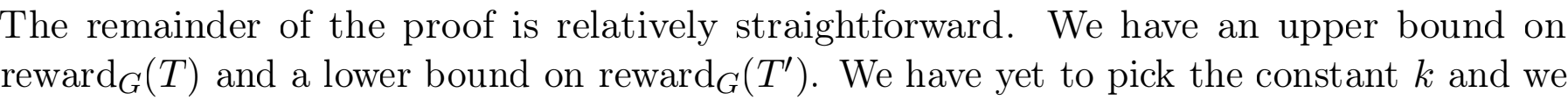
、◆

Weight Component

l

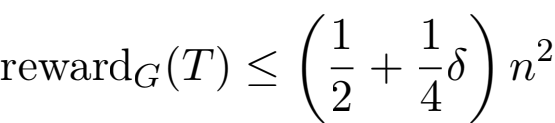


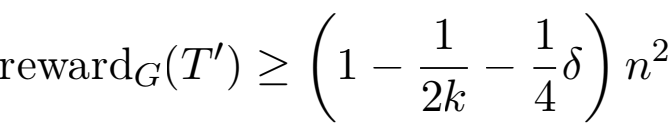




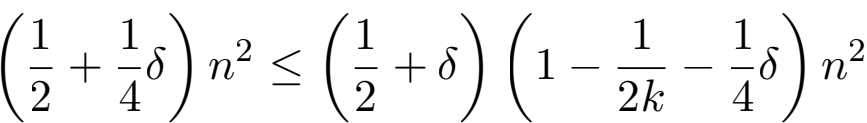
have yet to deﬁne what it means for n to be suﬃciently large. We want to use our remaining choices to guarantee a  gap between rewardG (T) and rewardG (T, ).

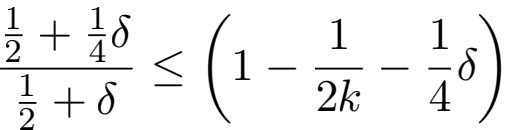
Our ﬁrst action is to let n be suﬃciently large so that O(k2n) and O(kn) are both at most 2 . Our bounds simplify to the following:

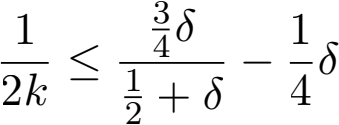




We know the ﬁnal gap we want, so our choice of k is now forced. We would like:







Note that for δ ≥  . However, if    and sothe RHS is positive. Hence it is possible to pick an integer k large enough

Approximation Bounds for Hierarchical Clustering

to create the ﬁnal gap we want. In particular, choosing k to be 「δ-1l suﬃces. Thus, some hierarchical clustering, and hence also the optimal clustering, beats average linkage in terms of cost by a factor of at least  . This completes the proof.



**6.5 Average Linkage and the Cost Objective**

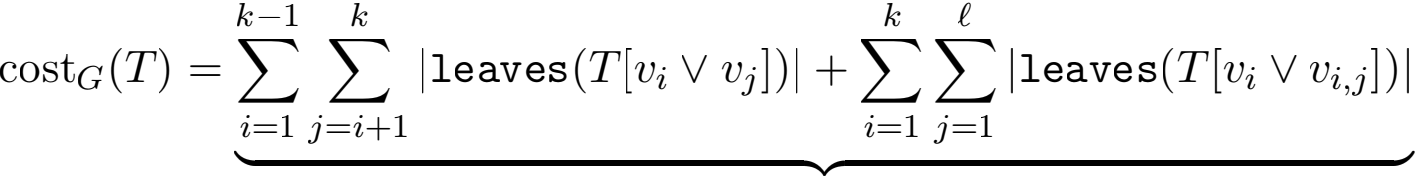
This subsection shows that the average linkage algorithm does not produce a good approxi- mation to Dasgupta’s cost objective.

**Theorem 13** There exists a family of graphs Gn with nonnegative edge weights w : E → R≥0 such that if the hierarchical clustering T\* is a minimizer of costG (·) and T is the hierarchical clustering returned by average linkage, costG (T) ≥ Ω(n1/3) · costG (T\* ) .

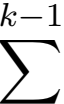
**Proof** For convenience, assume that n isa perfect cube. If n is not, then the graph can be padded with unconnected vertices.

The base counterexample graph family is the clique star graph; see Figure [5](#bookmark42) for an example. For the actual counterexample graph, several weights will be increased for tie- breaking purposes. Let ∈ > 0 be a constant that will be ﬁxed later. Choose Gn to be the (n1/3, n2/3 — 1)-clique star graph. The weights w are as follows. All edges are unit weight, except the clique edges which have weight (1 + ∈) instead.

Consider what average linkage does on Gn. Average linkage begins by merging together the clique of weight (1 + ∈). After this, the remaining graph is a large star, so average linkage ﬁnishes by merging in one node at a time into the center. This clustering is depicted in Figure [8.](#bookmark45) How much cost does this hierarchical clustering T accumulate?



Base Weight Component

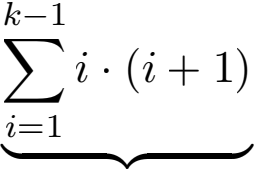
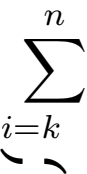
+ ∈  j leaves(T [vi V vj])j

i=1 j =i+1

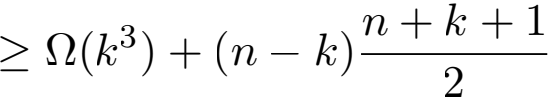
、 –

、◆

Tie-Breaking Weight Component

≥  + 1

Base Weight Component, Clique Edges Base Weight Component, Star Edges

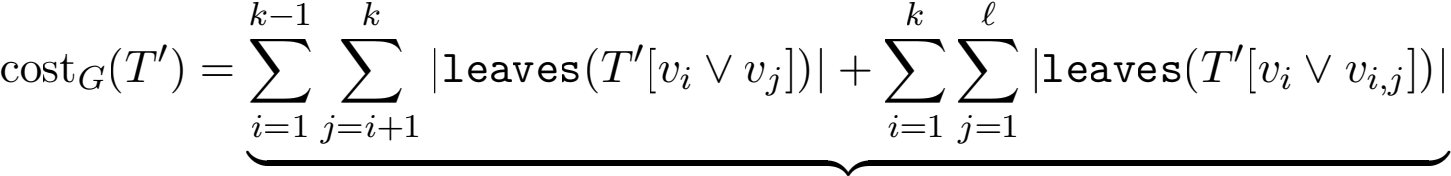


Since k = n1/3, this entire expression is Ω(n2 ). To ﬁnish the proof, itsuﬃces to show that some hierarchical clustering obtains at most O(n5/3) cost on this graph.

Consider the hierarchical clustering which ﬁrst handles the small stars (by merging in one node at a time into the center), and then merges together the remaining clique. This

Moseley and Wang

clustering is also depicted in Figure [8.](#bookmark45) How much cost does TI accumulate?



Base Weight Component

k—1 k

+ ∈Σ Σ | leaves(TI [vi ∨ vj])|

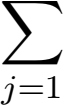
i=1 j =i+1

、 –

、◆

Tie-Breaking Weight Component

k—1 l

≤ Σi=1 i · (i + 1) · (l + 1) + k (j + 1) +∈ · O(k3l)

、、◆– 、、◆–

Base Weight Component, Clique Edges Base Weight Component, Star Edges

≤ O(k3l) + O(kl2 ) + ∈ · O(k3l)

Since k = n1/3 and l = n2/3 — 1, and we can choose ∈ = 1, this entire expression is O(n5/3). Thus, some hierarchical clustering, and hence also the optimal clustering, beats average linkage in terms of cost by a factor of at least Ω(n1/3). This completes the proof.



**6.6 Complete Linkage and the Reward Objective**

In this subsection, it is shown that the complete linkage algorithm does not produce a good approximation to the reward objective.

**Theorem 14** There exists a family of graphs Gn with nonnegative edge weights w : E → R≥0 such that if the hierarchical clustering T\* is a maximizer of rewardG (·) and T is the hierarchical clustering returned by complete linkage, rewardG (T) ≤  · rewardG (T\* ) .

**Proof** For convenience, assume that n is even. If n is not, then the graph can be padded with unconnected vertices.

The base counterexample graph family is the double star graph; see Figure [3](#bookmark40)for an example. Here n denotes the number of nodes and l = (n — 2)/2. For the actual counterexample graph, several weights will be increased for tie-breaking purposes. Let ∈ > 0 be a constant that will be ﬁxed later. Choose Gn to be the n/2-double star graph, although our tie-breaking additions will cause some additional edges to appear. The edge (v1 , vl+1) has weight (1 + ∈), while all other edges originally in the base graph have unit weight. Let S = [n] \ {1, l + 1}. For all pairs (i, j) ∈ S × S where i < j, add an edge of weight ∈ between vi and vj (that is, add a clique of weight ∈ on S).

Consider what complete linkage does on Gn. Complete linkage begins by merging the edge of weight (1 + ∈), creating clusters {v1 , vl+1}, {v2 }, . . . , {v n}. At this point, the {v1 , vl+1} cluster has zero complete linkage with every other cluster, but the complete linkage between any other pair of clusters is ∈ . Hence complete linkage will merge all other clusters (in some

Approximation Bounds for Hierarchical Clustering

tree T (breaks stars)

v2·l

v1

v2 · · ·

vl+1

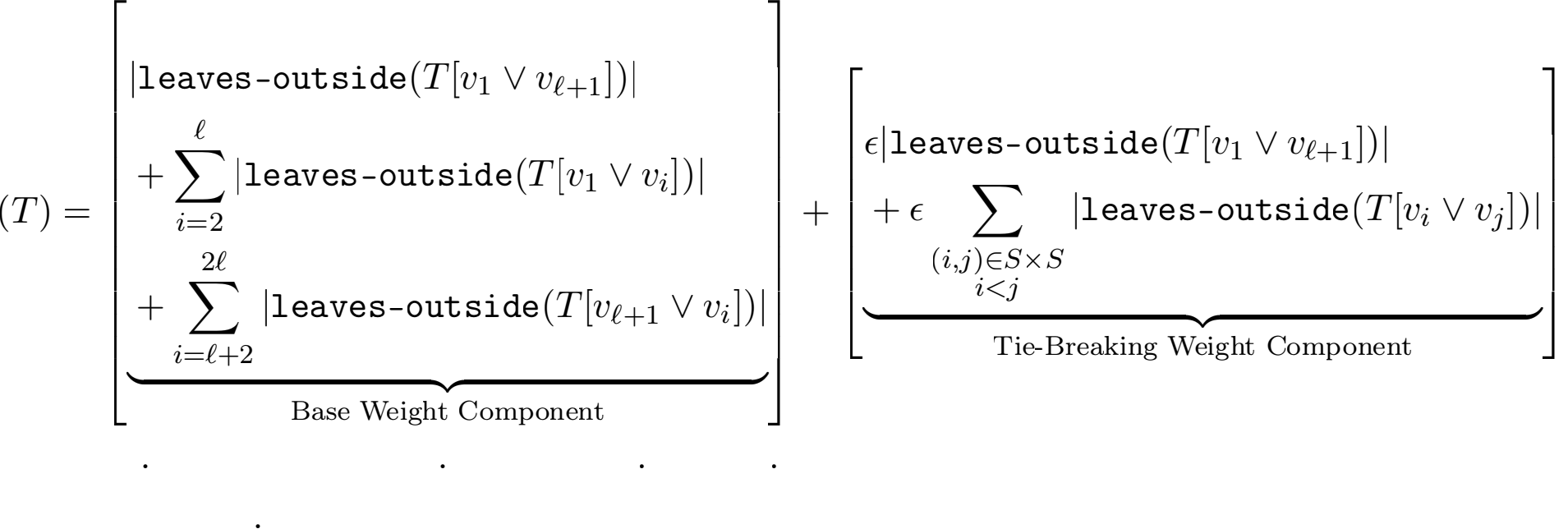
tree T/ (stars ﬁrst)

v1 v2 · · · vl vl+1 vl+2 · · · v2·l

Figure 9: Hierarchical clusterings from the proofs of Theorem [14](#bookmark34) and Theorem [16.](#bookmark36) Complete- linkage and Bisecting k-Means produce the top clustering, while the bottom clustering achieves a better objective value. Solid edges denote direct children, while dotted edges denote indirect children.

Moseley and Wang

order) and only then merge in {v1 , vl+1}. This clustering is depicted in Figure [9.](#bookmark46) How much reward does this hierarchical clustering T accumulate?

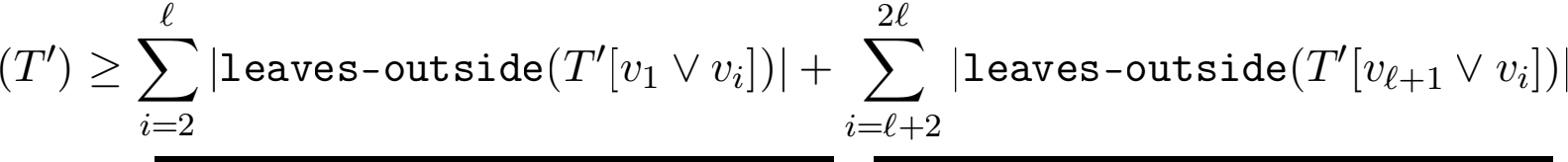
rewardG 

= [1 (n — 2) + (l — 1) 0 + (l — 1) 0] + [∈ O(n3 )]

= O(n) + ∈ O(n3 )

Fix ∈ to be 1/n2 , so this entire expression is O(n). To ﬁnish the proof, itsuﬃces to show that some hierarchical clustering obtains at least Ω(n2 ) reward on this graph.

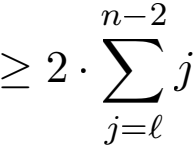
Consider the following hierarchical clustering TI . Cluster the stars separately; {v1 } is merged with v2 , then v3 , and so on, until vl is merged. At the sametime, {v k+1} is merged with vl+2, then v k+3, and so on, until v2l is merged. This clustering is also depicted in Figure [9.](#bookmark46) How much reward does TI accumulate?

rewardG 

、 –

、◆

Base Weight Component, Star Edges



≥ Ω(n2 )

We have shown that some hierarchical clustering, and hence also the optimal hierarchical clustering, beats complete linkage in terms of reward by a factor of at least Ω(n). This completes the proof. 

**6.7 Complete Linkage and the Cost Objective**

In this subsection, it is shown that the complete linkage algorithm does not produce a good approximation to Dasgupta’s cost objective.

**Theorem 15** There exists a family of graphs Gn with nonnegative edge weights w : E → R≥0 such that if the hierarchical clustering T\* is a minimizer of costG (·) and T is the hierarchical clustering returned by complete linkage, costG (T) ≥ Ω() · costG (T\* ) .

**Proof** For convenience,it will be assumed that n is even. If n is odd, then the graph can be padded with unconnected vertices.

Approximation Bounds for Hierarchical Clustering

tree T (S12 and S03 ﬁrst)

v13

v14

v1

v3

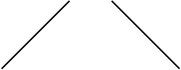
v4

v16

· · · v15

v2 · · ·

tree T, (balanced binary)







v13 v14 v15 v16

v1 v2 v3 v4

· · ·

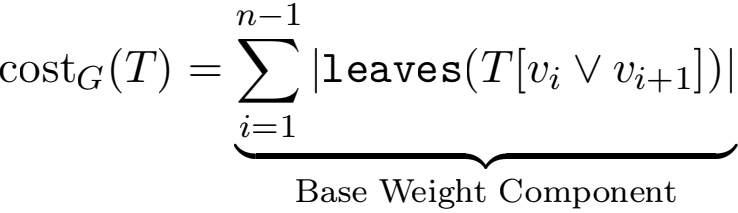
Figure 10: Hierarchical clusterings from the proof of Theorem [15, for](#bookmark35) n = 16. Complete- linkage produces the top clustering, while the bottom clustering achieves a better objective value. Solid edges denote direct children, while dotted edges denote indirect children.

Like with single linkage, the base counterexample graph family is the line graph; see Figure [1](#bookmark38) for an example. For the actual counterexample graph, several weights are increased for tie-breaking purposes. Let ∈ > 0 be a constant that will be ﬁxed later. Choose Gn to be Pn, although the tie-breaking additions will cause some additional edges to appear. Let S12 = {i ∈ [n] j i 三 1 (mod 4) or i 三 2 (mod 4)} and S03 = {i ∈ [n] j i 三 0 (mod 4) or i 三 3 (mod 4)}. For all pairs (i, j) ∈ S12 × S12 where i < j, add ∈ to the weight of edge (vi, vj). If such an edge did not exist before, pretend it existed as a zero-weight edge, which now has ∈-weight (that is,we add an clique of weight ∈ on S12). This process is repeated for all pairs (i, j) ∈ S03 × S03 where i < j (that is, we also add an clique of weight ∈ on S03).

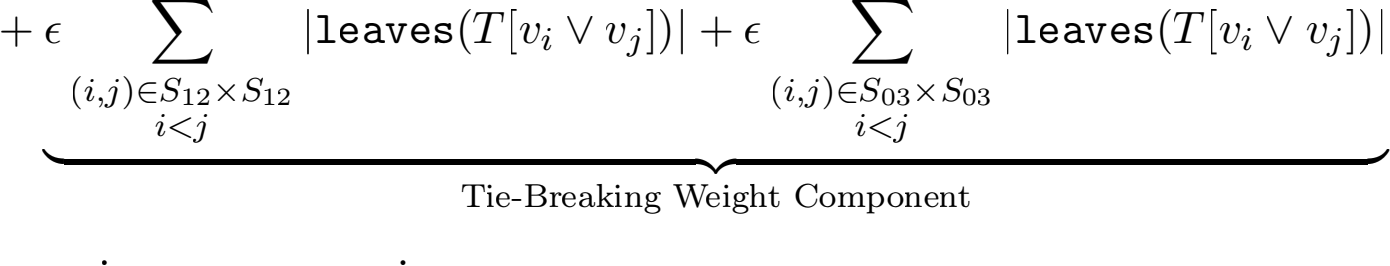
Consider what the complete linkage algorithm does on Gn. Complete link- age begins by merging all the edges of weight (1 + ∈), resulting with the clusters {v1 , v2 }, {v3 , v4 }, . . . , {vn-1 , v n}. At this point, there are two types of clusters. A clus-

Moseley and Wang

ter either (i) only includes vi where i is congruent to one or two modulo four, or (ii) only includes vi where i is congruent to zero or three modulo four. Furthermore, the complete linkage between two clusters of the former type is ∈, the complete linkage between two clusters of the latter type is ∈, and the complete linkage between two clusters of diﬀerent types is zero. Hence, complete linkage will merge all clusters of the former type (in some order) and all clusters of the latter type (in some order). Finally, it will merge these two large clusters together. This clustering is depicted in Figure [10.](#bookmark47) The cost this hierarchical clustering T accumulates is as follows.



≥ (n/2 2 + (n/2 — 1) n) ≥ Ω(n2 )



Note that n/2 of the base edges are involved immediately in a merge into a cluster of size two, while the remainder are involved in the ﬁnal merge into a cluster of size n. To ﬁnish the proof, itsuﬃces to show that some hierarchical clustering obtains at most O(nlog n) cost on this graph. The proof of this is identical to that in Theorem [11, except that the tie-breaking](#bookmark31) term is slightly diﬀerent. This clustering is also depicted in Figure [10.](#bookmark47) Choosing ∈ = 1/n2 still suﬃces to make the O(nlog n) term dominate, since for the current graph added at most O(n2 ) edges, each of which involves at most O(n) leaves.

Together this shows that some hierarchical clustering, and hence also the optimal cluster- ing, beats complete linkage in terms of cost by a factor of at least Ω(nlog n). This completes

the proof.

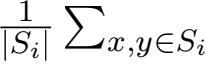


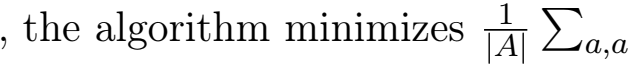
**6.8 Bisecting k-Means and the Reward Objective**

In this subsection, we consider the divisive algorithm which uses the k-means objective (with k = 2) when choosing how to split clusters. Normally, the k-means objective con-

cerns the distances between points and their cluster center: min Σ=1 Σx∈Si jjx — μijj 2 .

However, it is known that this can be rewritten as a sum over intra-cluster distances:

min Σ=1  jjx — yjj 2 [Awasthi et al.](#bookmark48) [(2015)](#bookmark48). In other words, when splitting a cluster

into two sets A and B , ∈A jja — aI jj 2 , ∈B jjb — bI jj 2 .

At ﬁrst glance,this appears to almost capture split-rewG (A, B); the key diﬀerence is that the summation has been scaled down by a factor of jAjjBj . Of course, it also involves minimization over squared distances instead of maximization over similarity weights. We show that the divisive algorithm which splits clusters by the natural k-means similarity

Approximation Bounds for Hierarchical Clustering

v6

A\*

v3

v4

v5

v7

v8

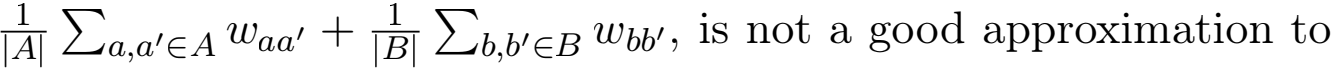
v9

v10

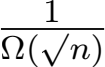
v1

B\* v2

Figure 11: The initial split (A\* , B\* ) of bisecting 2-means from the proof of Theorem [16, for](#bookmark36) l = 5.

objective, namely max  the optimal hierarchical clustering.

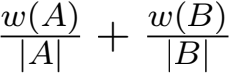
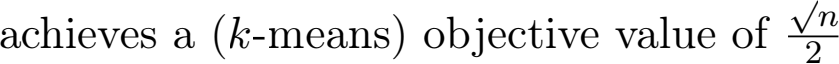
**Theorem 16** There exists a family of graphs Gn with nonnegative edge weights w : E → R≥0

such that if the hierarchical clustering T\* is a minimizer of costG (·) and T is the hierarchical clustering returned by bisecting k-Means, rewardG (T) ≤  · rewardG (T\* ) .

**Proof** For convenience, we will assume that n is even and a perfect square. If n is not, then we can always pad the graph with unconnected vertices.

The base counterexample graph family is the double star graph; see Figure [3](#bookmark40) for an example. In the actual counterexample graph, several weights are increased. Choose Gn to be the (l = n/2)-cycle star graph. The weights w are as follows. All edges are unit weight, except for the (v1 , vl+1) edge (which connects the two stars),which has √n weight instead.

Consider what bisecting k-means does on Gn. We want to show that the ﬁrst step of bisect- ing k-means results in the clusters A\* = {v1 , vl+1}, B\* = {v2 , v3, . . . , vl, vl+2, vl+3, . . . , v2l}. This split is depicted in Figure [11.](#bookmark49)

Recall that the k-means  objective is max  . Note that   . Can any other choice of clusters (A, B) match

this objective value? We begin by observing that Gn is a tree, so the graphs induced by A and B must be forests. A forest always has more vertices than edges, so the contribution of a cluster to the objective value is at most its average edge weight. Hence it cannot be the case that (v1 , vl+1) does not lie completely inside A or B ; this would result in objective value at most two.

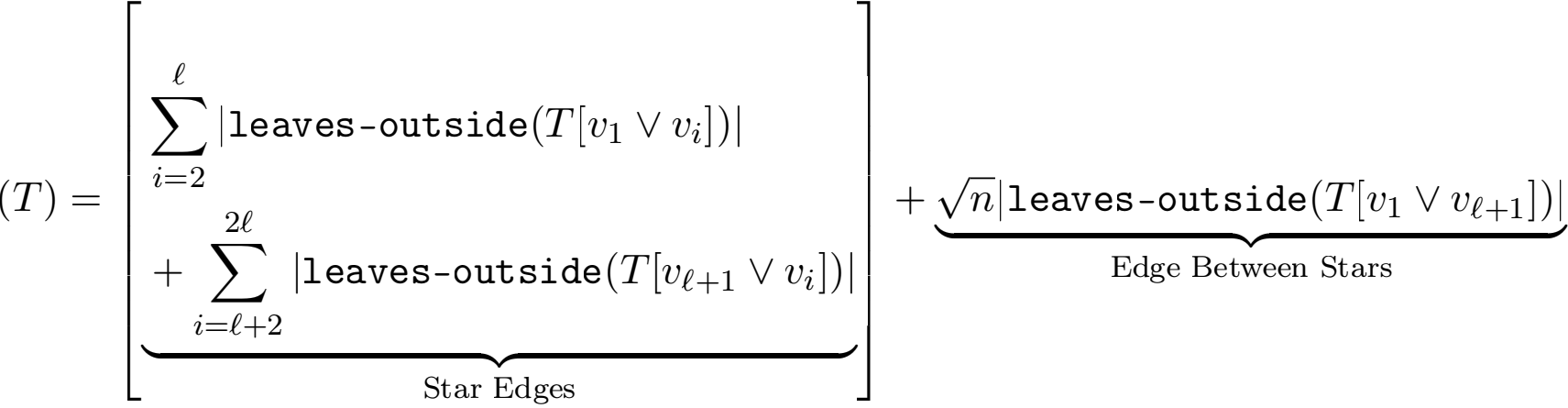
Without loss of generality, then, we assume that v1 , vl+1 ∈ A. At this point, B cannot contain edges since all edges have v1 or vl+1 as one of their endpoints. Furthermore, adding any other vertices to A only decreases its contribution, since any other vertex contributes

one to w  and one to |A| , which only serves to drag down the average  . Hence we

have shown that k-means begins by choosing clusters (A\* , B\* ). The upshot of this is that

Moseley and Wang

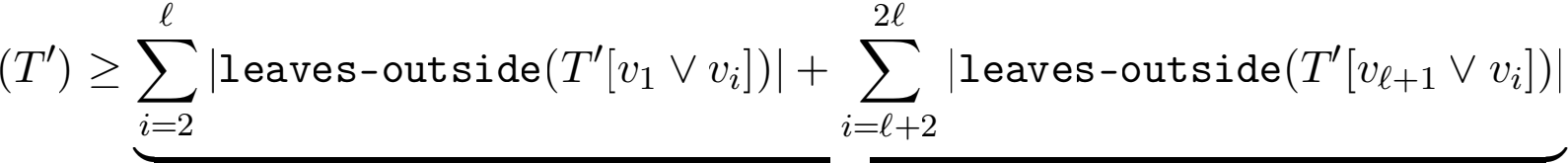
the initial split into (A\* , B\* ) cuts all n — 2 star edges, which greatly reduces the reward of T:

rewardG 



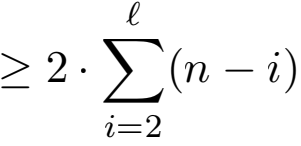
To ﬁnish the proof, it suﬃces to show that some hierarchical clustering obtains at least Ω(n2 ) reward on this graph.

Consider the following hierarchical clustering T/ . Cluster the stars separately: {v1 } is merged with v2 , thenv3 , and so on, until we merge in vl. At the sametime, {vl+1} is merged with vl+2, then vl+3, and so on, until we merge in v2l. Finally, merge these two clusters together. This clustering is also depicted in Figure [9.](#bookmark46) How much reward does T/ accumulate?

rewardG 

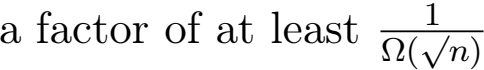
、◆

Star Edges



≥ Ω(ln)

Since l = n/2, this is Ω(n2 ), as desired. This shows that some hierarchical clustering, and hence also the optimal hierarchical clustering, beats bisecting k-means in terms of reward by

 . This completes the proof. 

**6.9 Bisecting k-Means and the Cost Objective**

This subsection gives a lower bound showing that bisecting k-means does not produce a good approximation to Dasgupta’s cost objective.

**Theorem 17** There exists a family of graphs Gn with nonnegative edge weights w : E → R≥0 such that if the hierarchical clustering T\* is a minimizer of costG (·) and T is the hierarchical clustering returned by bisecting k-Means, costG (T) ≥ Ω( √n) · costG (T\* ) .

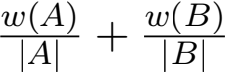
**Proof** For convenience, we will assume that n isa perfect square. If n is not, then we can always pad the graph with unconnected vertices.

The base counterexample graph family is the cycle star graph; see Figure [4](#bookmark41)for an example. In the actual counterexample graph, several weights are increased. Choose Gn to be the

Approximation Bounds for Hierarchical Clustering

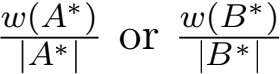
(n1/2, n1/2 — 1)-cycle star graph. The weights w are as follows. All edges are unit weight, except the clique edges which have weight √n instead.

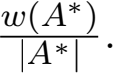
Consider what bisecting k-means does on Gn. The ﬁrst step of bisecting k-means is the most important, and the proof requires some analysis to prove this step is an irrevocable mistake.

Recall that the k-means  objective is max  . One choice of initial

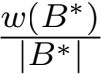
clusters is A = U and B = u=1 Ui. This choice has a (k-means similarity) objective score

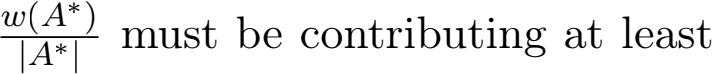
of √n. The optimal choice of initial clusters (A\* , B\* ) must hence score at least √n. This

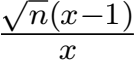
implies that either  must beat least  Without loss of generality, assume

t is  Since the total weight in the graph is at most 2n, this implies that A\* is small:

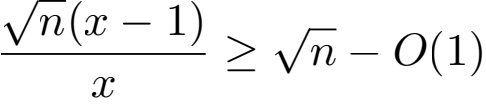
A\* | ≤ 4√n. B\* must be correspondingly large: |B\* | ≥ n — 4√n. This means that (assuming

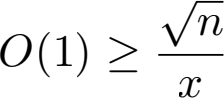
n is large enough)  is a constant!

We have come to the conclusion that  √n — O(1).

Notice that if A\* contains x < √ nodes from U , its contribution can beat most 

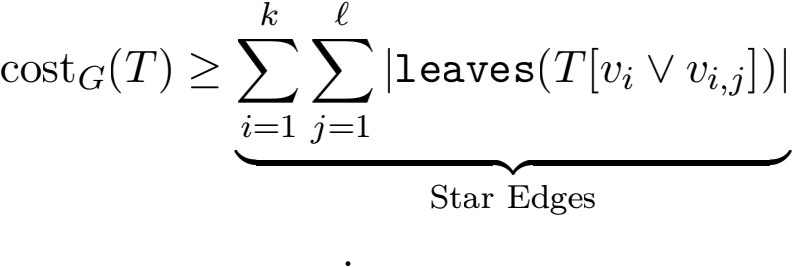
(nodes in V \ U only make matters worse, since they contribute one to the numerator and denominator at best). This implies that A\* must contain at least a constant fraction of the nodes in U :





x ≥ Ω(√n)

However, these nodes in A\* ∩ U are the centers of several stars. Together, these stars represent Ω(n) nodes, of which A\* cannot even cover a constant fraction, since we established that A\* is too small. These facts about the initial split are depicted in Figure [12.](#bookmark50) The upshot of this is that this initial split into (A\* , B\* ) cuts Ω(n) star edges, which will contribute a large factor to the cost of T :



≥ Ω(n) n = Ω(n2 )

To ﬁnish the proof, it suﬃces to show that some hierarchical clustering obtains at most O(n1.5 ) cost on this graph.

Consider the following hierarchical clustering T/ . Cluster the stars separately: {vi} is merged with vi,1, then vi,2, and so on, until we merge in vi,l. This leaves us with a cycle of k clusters,which we handle as we have handled line graphs; pair up each odd cluster with the next largest cluster, repeatedly, until we are left with a single cluster. This clustering is

Moseley and Wang

v1;4

v1;3

v1;2

v1;1

v2;1

v2;2

v2;3

v1

v2

v2;4

v5;4

v5

v5;3

v3;1

v3

|  |
| --- |
| A★ |

v4

v5;2

v5;1

v3;2

v4;4

v3;4

v4;1

v4;3

v3;3

v4;2

Figure 12: A depiction of the initial split (A★ , B★ ) of bisecting 2-means from the proof of Theorem [17, for](#bookmark37) k = 5, l = 4. Our analysis shows that A★ must contain a constant fraction of the central cycle nodes. Additionally, since it is of O(√n) size it cannot contain many star nodes.

tree T/ (stars ﬁrst)

· · ·

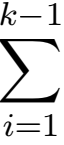
v1 v1;1 · · · v1;l · · ·

vk v k;1 · · · v k;l

Figure 13: Hierarchical clustering from the proof of Theorem [17.](#bookmark37) The depicted clustering achieves a better objective value than bisecting 2-means (which is illustrated in Figure [12)](#bookmark50). Solid edges denote direct children, while dotted edges denote indirect children.

Approximation Bounds for Hierarchical Clustering

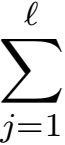
depicted in Figure [13.How much cost does](#bookmark51) T, accumulate?

cost G(T,) = √nj leaves(T,[vk V v1])j + √n  j leaves(T,[vi V vi+1])j

、 –

、◆

Cycle Edges

+Σi1  j leaves(T,[vi V vi,j])j

、 –

、◆

Star Edges ≤ (nlog n) + (kl2 )

Since k = √n and l = √n — 1, this is at most O(n1.5 ), as desired. This shows that some hierarchical clustering, and hence also the optimal hierarchical clustering, beats bi- secting k-means in terms of reward by a factor of at least Ω(√n). This completes the proof. 

**7. Conclusion**

One motive for developing an analytic framework is that it may help clarify and explain our observations from practice. In this case, we have shown that average linkage is a  [1), and the analysis that](#bookmark26) does so helps to explain what average linkage is optimizing. We have also shown that  [12)1 .](#bookmark32) One open problem is to devise new algorithms and determine the best approximation ratio possible for the problem. The current state of the art is a (≈ 0.336)-approximation based on semi-deﬁnite [programming (Charikar et al.,](#bookmark15) [2019a)](#bookmark15). Can this be improved further? Another open problem is to ﬁnda characterization of graphs that excludes some of the worst-case ones used to prove negative results. Is there a formal way to restrict inputs that allows for better objective guarantees?

We mention that similar results to ours for average-linkage have been shown by [Cohen-](#bookmark18) [Addad et al.](#bookmark18) [(2017)](#bookmark18).  related objective function when there are dissimilarity scores between the points.

Another open direction is the possibility of other objective functions. What are single linkage, complete linkage, and bisecting k-means optimizing for? One quirk shared by both Dasgupta’s cost objective and our reward objective is that the optimal tree is always binary. This is not appropriate for all applications; for example, in the classical application of biological taxonomy, groups typically contain much more than two subgroups. Can we devise an objective function which incentivizes non-binary trees?

**8. Acknowledgments**

Benjamin Moseley was supported in part by a Google Research Award, a Yahoo Research Award and NSF Grants CCF-1617724, CCF-1733873, CCF-1725661, CCF-1824303, CCF-

1.  ε > 0 [(Charikar et al.,](#bookmark15) [2019a)](#bookmark15).

Moseley and Wang

1845146 and CMMI-1938909. This work was partially done while the author was working at Washington University in St. Louis.

Joshua R. Wang was supported in part by NSF Grant CCF-1524062. This work was partially done while the author was at Stanford University.

**References**

Amir Abboud, Vincent Cohen-Addad, and Hussein Houdrougé . Subquadratic high- dimensional hierarchical clustering. In Advances in Neural Information Processing Systems, pages 11576–11586, 2019.

Margareta Ackerman and Shai Ben-David. A characterization of linkage-based hierarchical clustering. Journal of Machine Learning Research, 17:232:1–232:17, 2016.

Margareta Ackerman, Shai Ben-David, Simina Brânzei, and David Loker. Weighted clustering. In Proceedings of the Twenty-Sixth AAAI Conference on Artiﬁcial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada., 2012.

Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander ﬂows, geometric embeddings and graph partitioning. J. ACM, 56(2):5:1–5:37, 2009.

Pranjal Awasthi, Afonso S Bandeira, Moses Charikar, Ravishankar Krishnaswamy, Soledad Villar, and Rachel Ward. Relax, no need to round: Integrality of clustering formulations. In Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science , pages 191–200. ACM, 2015.

Shai Ben-David and Margareta Ackerman. Measures of clustering quality: A work- ing set of axioms for clustering. In Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, De- cember 8-11, 2008, pages 121–128, 2008. URL [http://papers.nips.cc/paper/](http://papers.nips.cc/paper/3491-measures-of-clustering-quality-a-working-set-of-axioms-for-clustering) [3491-measures-of-clustering-quality-a-working-set-of-axioms-for-clustering.](http://papers.nips.cc/paper/3491-measures-of-clustering-quality-a-working-set-of-axioms-for-clustering)

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.

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

Moses Charikar, Vaggos Chatziafratis, Rad Niazadeh, and Grigory Yaroslavtsev. Hierarchical clustering for euclidean data. In The 22nd International Conference on Artiﬁcial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan, pages 2721–2730, 2019b.

Approximation Bounds for Hierarchical Clustering

Giovanni Chierchia and Benjamin Perret. Ultrametric ﬁtting by gradient descent. In Advances in neural information processing systems, pages 3175–3186, 2019.

Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. Hierarchical clustering: Objective functions and algorithms. CoRR, abs/1704.02147, 2017.

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

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised Learning, pages

485–585. Springer New York, New York, NY, 2009.

Katherine A. Heller and Zoubin Ghahramani. Bayesian hierarchical clustering. In Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005, pages 297–304, 2005.

Anil K. Jain. Data clustering: 50 years beyond k-means. Pattern Recognition Letters, 31(8): 651 – 666, 2010. ISSN 0167-8655. doi: <https://doi.org/10.1016/j.patrec.2009.09.011>. URL [http://www.sciencedirect.com/science/article/pii/S0167865509002323.](http://www.sciencedirect.com/science/article/pii/S0167865509002323)

Akshay Krishnamurthy, Sivaraman Balakrishnan, Min Xu, and Aarti Singh. Eﬃcient active algorithms for hierarchical clustering. In Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012, 2012.

Silvio Lattanzi, Thomas Lavastida, Kefu Lu, and Benjamin Moseley. A framework for parallelizing hierarchical clustering methods. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 73–89. Springer, 2019.

Xiaofei Ma and Satya Dhavala. Hierarchical clustering with prior knowledge. arXiv preprint arXiv:1806.03432, 2018.

Aditya Krishna Menon, Anand Rajagopalan, Baris Sumengen, Gui Citovsky, Qin Cao, and Sanjiv Kumar. Online hierarchical clustering approximations. arXiv preprint arXiv:1909.09667, 2019.

Nicholas Monath, Manzil Zaheer, Daniel Silva, Andrew McCallum, and Amr Ahmed. Gradient- based hierarchical clustering using continuous representations of trees in hyperbolic space. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 714–722, 2019.

Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery, 2(1):86–97, 2012.

Aurko Roy and Sebastian Pokutta. Hierarchical clustering via spreading metrics. In Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, pages 2316–2324, 2016.

Dingkang Wang and Yusu Wang. An improved cost function for hierarchical cluster trees. arXiv preprint arXiv:1812.02715, 2018.

Moseley and Wang

Yuyan Wang and Ben Moseley. An objective for hierarchical clustering in euclidean space and its connection tobisecting k-means. In Proceedings of the 34th Conference on Artiﬁcial Intelligence (AAAI 2020), 2020.

Reza Zadeh and Shai Ben-David. A uniqueness theorem for clustering. In UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artiﬁcial Intelligence, Montreal, QC, Canada, June 18-21, 2009, pages 639–646, 2009.