# Approximating min-sum k-Clustering in Metric Spaces

Yair Bartal*
Computer Science Dept.
Hebrew University, Israel
yair@cs.huji.ac.il

Moses Charikar†
Google, Inc.
Mountain View, CA
moses@google.com

Danny Raz‡
Computer Science Dept.
Technion, Israel
danny@cs.technion.ac.il

## ABSTRACT

The *min-sum k-clustering* problem in a metric space is to find a partition of the space into $k$ clusters as to minimize the total sum of distances between pairs of points assigned to the same cluster. We give the first polynomial time non-trivial approximation algorithm for this problem. The algorithm provides an $O(\frac{1}{\epsilon}\log^{1+\epsilon} n)$ approximation to the min-sum $k$-clustering problem in general metric spaces, with running time $n^{O(1/\epsilon)}$. The result is based on embedding of metric spaces into hierarchically separated trees. We also provide a bicriteria approximation result that provides a constant approximation factor solution with only a constant factor increase in the number of clusters. This result is obtained by modifying and drawing ideas from recently developed primal dual approximation algorithms for facility location.

## 1. INTRODUCTION

Clustering is a fundamental problem that arises in many applications in different fields such as data mining, image processing, and bioinformatics.

In this paper we consider the following basic clustering problem (MIN-SUM $k$-CLUSTERING): given a weighted graph $G$ over $n$ vertices, partition the graph into into $k$ clusters $C_1, \ldots C_k$ as to minimize

$$\sum_{i=1}^{k} \sum_{u,v \in C_i} c_{u,v},$$

where $c$ is the weight function. This problem was first formulated by Sahni and Gonzalez [9]. The problem is dual to the MAX $k$-CUT problem where the goal is to maximize the weight of edges that do not belong to any cluster. Sahni and Gonzalez showed that these problems are NP-complete. They have also observed that they differ in hardness of approximation in that the min-sum $k$-clustering problem is harder to approximate. Kann et al. [7] show the $k$-clustering problem with $k \geq 3$ is hard to approximate to within a factor of $n^{2-\epsilon}$.

A natural question is to study the complexity of the problem for metric spaces. Our paper gives the first non-trivial approximation algorithm for this problem for arbitrary values of $k$. Guttman-Beck and Hassin [4] give a 2 approximation algorithm for the problem with running time $n^{O(k)}$. Indyk [5] gives a PTAS for $k = 2$. Schulman [10] gives an algorithm for points in $R^d$, with distance function $\ell_2^2, \ell_1$ or $\ell_2$. Schulman's algorithm either computes a solution within $1+\epsilon$ of the optimal cost or one that disagrees with the optimal solution on at most an $\epsilon$ fraction of the vertices (but may have much larger cost). Its running time is linear for $d = o(\log n/\log\log n)$ and $n^{O(\log\log n)}$ in the general case.

We give two result for arbitrary $k$, and arbitrary metric spaces. The first is an $O(\frac{1}{\epsilon}\log^{1+\epsilon} n)$ approximation for the min-sum $k$-clustering problem in metric spaces, with running time $n^{O(1/\epsilon)}$.

The result is based on the work of Bartal [1, 2] on embeddings of arbitrary metric spaces into hierarchically separated trees (HSTs). Informally, a $\mu$-HST is a weighted rooted tree of diameter $\mu$ times the diameter of each of its subtrees which are also $\mu$-HSTs. Bartal [2] showed that if there is an $\alpha$ approximation algorithm for $\mu$-HSTs then there is a randomized $O(\alpha\mu\log_\mu n \log\log n)$ approximation algorithm for any metric space. This result can be derandomized as shown in Charikar et al. [3] who show that it is enough to compute the solution for the problem in $O(n\log n)$ HSTs.

We obtain the first result by devising a constant approximation algorithm for the min-sum $k$-clustering problem in $\Omega(\log^\epsilon n)$-HSTs, with running time $n^{O(1/\epsilon)}$.

While reducing the problem to HSTs serves as a major simplification, providing a solution for HSTs remains a non trivial task. In particular it is not enough to assume it is an arbitrary weighted tree and we heavily rely on the hierarchical structure of the HST. This structure enables us to use a dynamic programming approach.

To do that we use the following problem formulation (we call *balanced k-median*): given a weighted graph $G$, partition it into $k$ clusters $C_1, \ldots, C_k$ with centers $c_1, \ldots, c_k$ as

*Part of this work was done while the author was at Bell Labs, Lucent Technologies.

†Most of this work was done while the author was at the Computer Science Dept., Stanford University, with research supported by the Pierre and Christine Lamond Fellowship, ARO MURI Grant DAAH04-96-1-0007 and NSF Grant IIS-9811904.

‡Part of this work was done while the author was at Bell Labs, Lucent Technologies.


*STOC'01,* July 6-8, 2001, Hersonissos, Crete, Greece.

to minimize

$$\sum_i |C_i| \sum_{v \in C_i} d(v, c_i).$$

The solution to this problem is within a factor of 2 of that of the original $k$-clustering problem. We note that the earlier algorithm of Guttman-Beck and Hassin[4] for constant $k$ also makes use of this problem formulation. However, their algorithm is based on enumerating all possible combinations of cluster sizes and solving the above problem when cluster sizes are fixed. Since we require the running time to be independent of $k$, such an algorithm is not acceptable.

The above problem formulation leads to a dynamic programming solution on HSTs which has running time $n^{O(\log n)}$. To achieve a polynomial time algorithm we impose restrictions on the set of solutions as well as using a $\mu$-HST with $\mu = \Omega(\log^\epsilon n)$ so that the number of levels in the HST can be restricted to $O(\frac{1}{\epsilon} \log n / \log \log n)$. This results in an algorithm with running time $(\log n)^{O(\frac{1}{\epsilon} \log n / \log \log n)}$ which is polynomial in $n$.

The second result we present is a bi-criteria approximation algorithm that produces a solution whose cost is within a constant factor of the optimal cost; however the number of clusters used is a constant factor more than $k$. Additionally, one can trade off the tightness of the approximation guarantee for the number of clusters versus the approximation guarantee on the cost. To obtain this result, we formulate the problem as a capacitated $k$-median problem on an auxiliary bipartite graph. In fact the distance function we define on this graph does not satisfy the triangle inequality. Instead of solving this capacitated $k$-median problem directly, we solve a facility location type problem and obtain a bicriteria approximation (in the style of Marathe *et al.* [8]).

To do so, we use the primal dual algorithm of Jain and Vazirani [6]. We need to modify the algorithm to deal with capacities and the fact that the distances do not satisfy triangle inequality.

All together we are able to obtain a $(1 + \delta)$ times the number of clusters and a $19.3(1+\delta)/\delta^2$ approximation factor for the cost, e.g. we get 6.03-approximation using at most $5k$ clusters.

## 2. PRELIMINARIES

DEFINITION 1 (MIN-SUM $k$-CLUSTERING). *Given a weighted graph $G$ over $n$ vertices, partition the graph into $k$ clusters $C_1, \ldots C_k$ as to minimize $\sum_{i=1}^{k} \sum_{u,v \in C_i} c_{u,v}$, where $c$ is the distance function.*

Given a specific partition $P$ of $G$ into $k$ clusters, we define the cost of the partition to be $\mathbf{cost}(P) = \sum_{i=1}^{p} \sum_{u,v \in C_i} c_{u,v}$. The MIN-SUM $k$-CLUSTERING problem is then to find a clustering that minimizes the cost. We are going to define a different closely related optimization problem.

DEFINITION 2 (BALANCED $k$-MEDIAN). *Given a weighted graph $G$ over $n$ vertices, choose $k$ vertices as centers $c_1, \ldots c_k$, and assign each vertex $v$ to a center $c_i$ as to minimize $\sum_{i=1}^{k} |C_i| \sum_{v \in C_i} c_{v,c_i}$, where $C_i$ is the set of vertices assigned to center $c_i$.*

CLAIM 1. *A given weighted graph $G$ has a partition into $k$ clusters with cost $C$ (for the min-sum $k$-clustering objective) iff it has a partition into $k$ clusters $C_1, \ldots C_k$ with centers $c_1, \ldots c_k$, such that*

$$\frac{1}{2} \sum_{i=1}^{k} |C_i| \sum_{v \in C_i} c_{v,c_i} \leq C \leq \sum_{i=1}^{k} |C_i| \sum_{v \in C_i} c_{v,c_i}.$$

PROOF. Given a partition into clusters with centers, we have from the triangle inequality, $c_{u,v} \leq c_{u,c_i} + c_{v,c_i}$. Summing up over all pairs $u, v \in C_i$, we get,

$$\sum_{u,v \in C_i} c_{u,v} \leq |C_i| \sum_{v \in c_i} c_{v,c_i}$$

Also, given a partition into $k$ clusters $C_1, \ldots C_k$, define the center of cluster $i$ to be

$$c_i = \text{argmin}_{v \in C_i} \{ \sum_{u \in C_i} c_{u,v} \}.$$

From the definition of $c_i$ we get,

$$\begin{aligned}
\sum_{v \in C_i} c_{u,v} &\geq \sum_{v \in C_i} c_{v,c_i} \\
\sum_{u \in C_i} \sum_{v \in C_i} c_{u,v} &\geq |C_i| \sum_{v \in C_i} c_{v,c_i} \\
2 \sum_{u,v \in C_i} c_{u,v} &\geq |C_i| \sum_{v \in C_i} c_{v,c_i}
\end{aligned}$$

Hence, we get that

$$\frac{1}{2} |C_i| \sum_{v \in C_i} c_{v,c_i} \leq \sum_{u,v \in C_i} c_{u,v} \leq |C_i| \sum_{v \in C_i} c_{v,c_i}.$$

Summing this over all clusters $i$ proves the claim. $\square$

Claim 1 yields the following lemma:

LEMMA 1. *Given a weighted graph $G$ and a $c$-approximation for the balanced $k$-median problem on $G$ there exist a $2c$-approximation algorithm for the min-sum $k$-clustering problem on $G$.*

Lemma 1 allows us to focus attention on approximating the balanced $k$-median problem. Also note that the proof of Claim 1 implies we may assume w.l.o.g. that each center $c_i$ belongs to its center $C_i$.

Our near logarithmic approximation algorithms is based on approximation of arbitrary metric spaces by "hierarchically well-separated trees". We start by reviewing the notion of approximating metric spaces.

Let $V$ be a set of $n$ points. We study the set of metric spaces defined over $V$. Given such a metric space $M$, the distance between $u$ and $v$ in $V$ is denoted $c^M_{u,v}$.

DEFINITION 3. *A metric spaces $N$ over $V$, dominates a metric space $M$ over $V$, if for every $u, v \in V$, $c^N_{u,v} \geq c^M_{u,v}$.*

DEFINITION 4. *A metric spaces $N$ over $V$, $\alpha$-approximates a metric space $M$ over $V$, if it dominates $M$ and for every $u, v \in V$, $c^N_{u,v} \leq \alpha \cdot c^M_{u,v}$.*

DEFINITION 5 ([1]). *A set of metric spaces $\mathcal{S}$ over $V$, $\alpha$-probabilistically-approximates a metric space $M$ over $V$, if every metric space in $\mathcal{S}$ dominates $M$ and there exists a probability distribution over metric spaces $N \in \mathcal{S}$ such that for every $u, v \in V$, $\mathbf{E}(c^N_{u,v}) \leq \alpha \cdot c^M_{u,v}$.*

The following definition of "hierarchically well-separated trees" is from [1]:

DEFINITION 6 ($\mu$-HST). *For $\mu > 1$, a $\mu$-HST is a rooted weighted tree with the following properties: the weights from a node to each of its children are equal, and on every path from the root to a leaf weights are dropping by a factor of $\mu$ at each step.*

It will be more convenient for us to use a somewhat modified definition which has more regular structure as follows:

DEFINITION 7 ($\mu$-SHST). *For $\mu > 1$, a $\mu$*-hierarchical well-separated tree in special form *($\mu$-SHST) is a metric space defined on the leaves of a weighted rooted tree $T$. Let the level of an internal node in the tree be the number of edges on the path to the root. Let $\Delta$ denote the diameter of the resulting metric space. For a vertex $u \in T$ let $\Delta(u)$ denote the diameter of the subtree rooted at $u$. Then the tree has the following properties:*

- *All edges at a particular level are of the same weight.*
- *All leaves are at the same level.*
- *For any internal node $u$, $\Delta(u) = \Delta\mu^{-i}$*

Note that a $\mu$-SHST in special form defines a tree metric. Further, for $\mu > 2$, this is a $(\mu - 1)$-HST since edge weights in level $i$ are $\Delta(\mu^{-i} - \mu^{-i-1})/2$ (and edge weights in the last level $\ell$ are $\Delta\mu^{-\ell}/2$). Also the distance between two leaves $x, y \in T$ is equal to $\Delta(\mathrm{lca}(x, y))$ where lca() is the least common ancestor in $T$.

The following lemma explains how we may construct a $\mu$-SHST from the standard HST:

LEMMA 2. *Let $\mu = 2^h$, $h \geq 1$. Any 2-HST $T$, can be $O(\mu/\log\mu)$-probabilistically-approximated by a set of $\mu$-SHSTs. Moreover the size of this set is $h$.*

PROOF. First, construct $T'$ by rounding all edge weights in $T$ to powers of 2. Then for each path in $T'$ from the root to a leaf start going over the edges starting at the root. If the edge length drops by more than a factor of 2, add internal vertices to complete all missing powers of 2. The last step is to add another edge of length 1 at the leaves. It is easy to verify that $T'$ is a 2-SHST that 2-approximates $T$. Next we tranform it into a $\mu$-SHST. If $h = 1$ we are done. For $h > 1$ it will be convient to first define $T''$ by creating a new root attached by a path of $h$ edges to the root of $T'$, where the $j$th edge from the root of $T'$ is assigned weight $2^j\Delta$. Now choose at random $i$ between 1 and $h$, then remove from the tree all internal vertices in $T''$ not in levels $i + th$ for some intergal $t$. Define a tree $S$ by connecting every vertex directly to its ancestor in $T''$. It follows from the construction that for a vertex $v$ with parent $u$ in $S$ we have $\Delta(v) \leq \Delta(u)/\mu$. It follows that $S$ is a $\mu$-SHST. We are left to show that $S$ probabilistically-approximates $T'$. Consider two vertices with lca $u$ in $T'$ at level $\ell$. Let $d = \Delta(u)$. For the $h$ different choices of $i$, the new lca of these vertices in $S$ will be at a different level between $\ell$ and $\ell + h$. This implies that the expected distance between them in $S$ is $d/h \sum_{j<h} 2^j \leq d\mu/\log\mu$ which concludes the claim. $\square$

The following theorem follows from [2, 3] together with the lemma above:

THEOREM 1. *If there exists an $\alpha$ approximation algorithm for the $k$-clustering (respectively, balanced $k$-median) problem on any $\mu$-SHST then there exists an $O(\alpha\mu\log_\mu n \log\log n)$ approximation algorithm in any metric space.*

In the rest of the paper we will use the definition of a $\mu$-SHST. However we will abuse notation and use the term $\mu$-HST instead.

# 3. A NEAR LOGARITHMIC APPROXIMATION

The goal of this section is to give an approximation algorithm for the min-sum $k$-clustering problem on an HST. In order to ensure a polynomial running time, we will need a $\Omega(\log^\epsilon n)$-HST. This results in an approximation ratio of $O(\frac{1}{\epsilon}\log^{1+\epsilon} n)$ and running time $n^{O(1/\epsilon)}$.

Lemma 1 and Theorem 1 allow us to focus on getting a constant factor approximation to the *balanced $k$-median* problem on a $\mu$-HST. We do this using dynamic programming. We first define a simple (but still non-trivial) dynamic programming algorithm for an HST. Intuitively speaking this algorithm computes solutions for the possible partitions of points into sets which connect to particular centers from the same distance. This algorithm runs in time $n^{O(\log n)}$.

In order to decrease the running time, we restrict the set of possible solutions. This is done by approximating the sizes of the sets we choose to within a small factor $1 + 1/\mathrm{polylog}(n)$. We also prove that the optimal solution to the original instance can be converted to a solution of the special restricted form that the dynamic program computes with an extra cost of only a constant factor. The running time of this algorithm is $(\log n)^{O(L)}$ where $L$ is the number of levels in the $\mu$-HST. We observe that without loss of generality we may assume that the $\mu$-HST has $O(\log_\mu n)$ levels. In order to ensure this, we run an algorithm for every edge in $G$ assuming that it is the highest level edge used by the optimal solution. All edges that are a $1/n^2$ factor smaller than this can be collapsed, since their total contribution to the cost is at most a constant times the optimal. This consideration implies the running time is $(\log n)^{O(\log_\mu n)}$. In order to get a polynomial running time, we use a $\mu$-HST with $\mu = \Omega(\log^\epsilon n)$; this results in a running time of $(\log n)^{O(\frac{1}{\epsilon}\log n/\log\log n)} = n^{O(1/\epsilon)}$.

Theorem 1 implies an approximation algorithm for arbitary metric spaces with similar running time and approximation ratio of $O(\alpha\mu\log_\mu n\log\log n) = O(\frac{1}{\epsilon}\log^{1+\epsilon} n)$.

## 3.1 Basic tree structure for the dynamic programs

Consider a $\mu$-HST. We define the level of a node $v$ in a (rooted) tree to be the number of nodes on the shortest path from $v$ to the root of the tree. Thus the root is at level 1, its children are at level 2 and so on. Also, if an edge $e$ connects two vertices at levels $\ell$ and $\ell+1$, then we say that $e$ is at level $\ell$. Firstly, in order to run the dynamic program, we convert the tree to a binary tree. In order to do this, we replace a node at level $\ell$ with $t$ children at level $\ell + 1$ by a $\log t$ level binary tree, introducing dummy nodes in the tree. We will refer to the levels in the original tree as *real* levels and the new levels introduced as *virtual* levels. Further, we will say that the internal nodes in the tree constructed are all at *real* level $\ell + 1$; the root itself is at *real* level $\ell$. Also note, that in the task of reducing the number of levels in the HST, we have collapsed edges, and thus a number of points in the original metric space could be mapped to the same leaf. If a leaf consists of $t$ points in the original metric space, we replace it by a $\log t$ level tree. Each leaf in the tree obtained corresponds to exactly one point in the original metric.

## 3.2 A simple dynamic program

The dynamic program will compute partial solutions for subtrees of the HST in a bottom up fashion. We say that a vertex $u$ connects to a center in the subtree $T_v$ *from level* $i$, if $u$ is located outside the subtree $T_v$ and the highest level (with the smallest index) in the path from $u$ to $v$ is $i$. Note that in this case, the distance between $u$ and any leaf in $T_v$ is $\Delta\mu^{-i}$ as we explained above.

The dynamic program proceeds as follows. For every node $v$ located at *real* level $\ell$ and vector of integers $(k, s, n_1, n_2, \ldots n_\ell)$, we compute $D^*(v, k, s, n_1, n_2, \ldots, n_\ell)$. This is defined to be the value of the optimal solution for the subtree $T_v$ rooted at $v$ such that $k$ leaves are chosen as centers from amongst the leaves in $T_v$, at most $s$ of the leaves in $T_v$ draw their service from outside the tree (and are *not* accounted for in the cost) and for $i = 1, \ldots \ell$, $n_i$ vertices connect to centers within the subtree $T_v$ from level $i$. Note that the value of $D^*(v, k, s, n_1, \ldots n_\ell)$, includes the contribution of all the clusters whose centers are located within $T_v$, including the cost contribution of leaves that connect from outside $T_v$. In computing the cost contribution of such leaves, the only information we need is the distance of the leaf from the center it is connecting to; this in turn is completely determined by the level that the leaf is connecting from. On the other hand the cost contribution of the $s$ leaves in $T_v$ which connect to centers outside $T_v$ is accounted for in the subtrees containing the centers that the leaves connect to, and *not* in the cost of $T_v$ .

We now describe the steps in the dynamic program in greater detail. The value $D(v, \ldots)$ is defined by induction on the tree structure:

*Internal Node.* Consider a node $v$ at real level $\ell$ with children $v_1$ and $v_2$ at real level $\ell'$. Note that the value of $\ell'$ is either $\ell$ or $\ell + 1$. Since we assume that each center belongs to its cluster, $D(v, k, s, \ldots)$ is undefined if $k$ is greater than the number of leaves in $T_v$.

$$\begin{aligned} D(v, k, s, n_1, n_2 \ldots n_\ell) = \min\{&D(v_1, k_1, s_1, n_1^{(1)}, n_2^{(1)} \ldots n_{\ell'}^{(1)}) \\ &+ D(v_2, k_2, s_2, n_1^{(2)}, n_2^{(2)} \ldots n_{\ell'}^{(2)})\} \end{aligned}$$

where the RHS of the above is minimized over all settings of values such that

1. $k = k_1 + k_2$, $k_1$ is not greater than the number of leaves in $T_{v_1}$ and $k_2$ is not greater than the number of leaves in $T_{v_2}$.
2. $\forall i < \ell'$, $n_i = n_i^{(1)} + n_i^{(2)}$.
3. $s_1 = s_{1,2} + s_{1,out}$, $s_2 = s_{2,1} + s_{2,out}$, $s = s_{1,out} + s_{2,out}$, $n_{\ell'}^{(1)} \geq s_{2,1}$ and $n_{\ell'}^{(2)} \geq s_{1,2}$.
4. If $\ell' = \ell$, $n_\ell = n_\ell^{(1)} - s_{2,1} + n_\ell^{(2)} - s_{1,2}$.
5. If $\ell' = \ell + 1$, then $n_{\ell'}^{(1)} = n_{\ell'}^{(2)} = 0$.

Here $s_{1,2}$ ($s_{2,1}$) represents the number of nodes in $T_{v_1}$ ($T_{v_2}$) supported by centers in $T_{v_2}$ ($T_{v_1}$), and $s_{1,out}$ ($s_{2,out}$) represents the number of nodes in $T_{v_1}$ ($T_{v_2}$) supported by centers outside both $T_{v_2}$ and $T_{v_1}$.

*Base Case.* This involves computing $D(v, k, n_1, n_2, \ldots, n_\ell)$ for a leaf node $v$. Recall that $v$ corresponds to a single point in the original metric.

The value $D(v, k, n_1, n_2, \ldots, n_\ell)$ is undefined for $k > 1$. The value is undefined if $k = 0$ and $n_i > 0$. For $k = 0$ and all $n_i = 0$, the value is 0. For $k = 1$, the value is $size \cdot \sum_i n_i \cdot \Delta\mu^{-i}$ where $size$ is $1 + \sum_i n_i$ if $s = 0$ and $\sum_i n_i$ is $s > 0$.

*Running Time.* Note that for each node $v$, we have to compute $n^{O(\log n)}$ values. The computation of each value takes $n^{O(\log n)}$ time. Thus the entire dynamic programming algorithm runs in $n^{O(\log n)}$ time.

The following lemma asserts that the value $D(\ldots)$ computed by the algorithm is exactly the value of the optimal solution $D^*(\ldots)$.

LEMMA 3. *For every node $v$ and parameters $k, s, n_1, \ldots, n_\ell$, $D(v, k, s, n_1, \ldots, n_\ell) = D^*(v, k, s, n_1, \ldots, n_\ell)$*

PROOF. We prove this by induction on the height of the node $v$.

The base case corresponds to a leaf node $v$. In this case, $D$ and $D^*$ are defined only if either $k = 0$ and all $n_i = 0$ or if $k = 1$. In the former case, $D(v, \ldots) = D^*(v, \ldots) = 0$. The latter case corresponds to exactly one cluster with $v$ as a center and $n_i$ nodes connecting from level $i$. The size of the cluster is either $1 + \sum n_i$ (if $s = 0$) or $\sum n_i$ (if $s = 1$). The cost of the cluster is the cluster size multiplied by $\sum_i n_i \cdot \Delta\mu^{-i}$. This is exactly the value of $D^*(v, \ldots)$ computed by the algorithm.

Suppose the lemma holds for all nodes of height $j$. Consider a node $v$ of height $j + 1$ with children $v_1$ and $v_2$ of height $j$.

We claim that $D(v, k, s, n_1, \ldots, n_\ell) \leq D^*(v, k, s, n_1, \ldots, n_\ell)$. Consider the optimal solution for subtree $T_v$. It is obtained by putting together optimal solutions for subtrees $T_{v_1}$ and $T_{v_2}$.

$$\begin{aligned} D^*(v, k, s, n_1, \ldots, n_\ell) \quad = \quad & D^*(v_1, k_1, s_1, n_1^{(1)}, \ldots, n_{\ell'}^{(1)}) \\ & + D^*(v_2, k_2, s_2, n_1^{(2)}, \ldots, n_{\ell'}^{(2)}). \end{aligned}$$

In the optimal solution, of the $k$ centers in subtree $T_v$, $k_1$ centers are located in subtree $T_{v_1}$ and $k_2$ centers are located in subtree $T_{v_2}$. Thus $k = k_1 + k_2$; $k_1$ is at most the number of leaves in $T_{v_1}$ and $k_2$ is at most the number of leaves in $T_{v_2}$.

Of the $n_i$ nodes that connect from level $i$, $n_i^{(1)}$ are assigned to centers in subtree $T_{v_1}$ and $n_i^{(2)}$ are assigned to centers in subtree $T_{v_2}$. Thus $\forall i < \ell', n_i = n_i^{(1)} + n_i^{(2)}$.

A total of $s_1$ nodes connect to centers outside subtree $T_{v_1}$. Of these, suppose $s_{1,2}$ nodes connect to centers within subtree $T_{v_2}$ and $s_{1,out}$ nodes connect to centers outside of $T_v$. A total of $s_2$ nodes connect to centers outside subtree $T_{v_2}$. Of these, suppose $s_{2,1}$ nodes connect to centers within subtree $T_{v_1}$ and $s_{2,out}$ nodes connect to centers outside of $T_v$. Then, the total number of nodes within subtree $T_v$ that connect to centers outside $T_v$ is $s = s_{1,out} + s_{2,out}$.

If $\ell' = \ell+1$, the $s_{2,1}$ nodes from subtree $T_{v_2}$ that connect to centers within subtree $T_{v_1}$ do so at level $\ell$, and therefor $n^{(1)}_{\ell'} = n^{(2)}_{\ell'} = 0$.

If $\ell' = \ell$, then the $s_{2,1}$ nodes from subtree $T_{v,2}$ that connect to centers within subtree $T_{v,1}$ do so at level $\ell$. Since subtree $T_{v_1}$ has a total of $n^{(1)}_{\ell}$ nodes connecting from level $\ell$, the remaining $n^{(1)}_{\ell} - s_{2,1}$ nodes must connect at level $\ell$ from outside subtree $T_v$. Similarly, for subtree $T_{v_2}$, $n^{(2)}_{\ell} - s_{1,2}$ nodes must connect at level $\ell$ from outside subtree $T_v$. Hence the total number of nodes connecting at level $\ell$ to centers in subtree $T_v$ (from outside $T_v$) is $n_\ell = n^{(1)}_{\ell} - s_{2,1} + n^{(2)}_{\ell} - s_{1,2}$.

By the inductive hypothesis,

$$D(v_1, k_1, s_1, n^{(1)}_1, \ldots n^{(1)}_{\ell'}) = D^*(v_1, k_1, s_1, n^{(1)}_1, \ldots, n^{(1)}_{\ell'})$$
$$D(v_2, k_2, s_2, n^{(2)}_1, \ldots n^{(2)}_{\ell'}) = D^*(v_2, k_2, s_2, n^{(2)}_1, \ldots n^{(2)}_{\ell'})$$

Notice the the values $k_1, k_2, n^{(1)}_1, \ldots, n^{(1)}_{\ell'}, n^{(2)}_1, \ldots, n^{(2)}_{\ell'}, s_1, s_2, s_{1,2}, s_{1,out}, s_{2,1}, s_{2,out}$ satisfy the conditions checked by the algorithm.

$$\begin{aligned}
&D(v, k, s, n_1 \ldots n_\ell) \\
&\leq D(v_1, k_1, s_1, n^{(1)}_1 \ldots n^{(1)}_{\ell'}) + D(v_2, k_2, s_2, n^{(2)}_1 \ldots n^{(2)}_{\ell'}) \\
&= D^*(v_1, k_1, s_1, n^{(1)}_1 \ldots n^{(1)}_{\ell'}) + D^*(v_2, k_2, s_2, n^{(2)}_1 \ldots n^{(2)}_{\ell'}) \\
&= D^*(v, k, s, n_1 \ldots n_\ell)
\end{aligned}$$

To complete the proof of equality of $D(\ldots)$ and $D^*(\ldots)$, we show that $D^*(v, k, s, n_1, \ldots, n_\ell) \leq D(v, k, s, n_1, \ldots, n_\ell)$.

$$\begin{aligned}
&D(v, k, s, n_1 \ldots n_\ell) \\
&= D(v_1, k_1, s_1, n^{(1)}_1 \ldots n^{(1)}_{\ell'}) + D(v_2, k_2, s_2, n^{(2)}_1 \ldots n^{(2)}_{\ell'}) \\
&= D^*(v_1, k_1, s_1, n^{(1)}_1 \ldots n^{(1)}_{\ell'}) + D^*(v_2, k_2, s_2, n^{(2)}_1 \ldots n^{(2)}_{\ell'})
\end{aligned}$$

By the inductive hypothesis, there exist solutions for subtrees $T_{v_1}$ and $T_{v_2}$ with costs $D(v_1, k_1, s_1, n^{(1)}_1, \ldots, n^{(1)}_{\ell'})$ and $D(v_2, k_2, s_2, n^{(2)}_1, \ldots, n^{(2)}_{\ell'})$ respectively. Since the algorithm considered these two solutions in the computation of $D(v, k, s, n_1, \ldots, n_\ell)$, the conditions checked by the algorithm imply that the two solutions can be combined to produce a solution for $T_v$. Thus there exists a solution for $T_v$ with cost at most $D(v, k, s, n_1, \ldots, n_\ell)$. $\square$

## 3.3 An improved dynamic program

In order to reduce the running time, we want to reduce the number of values computed by the dynamic program. Let $\alpha = 1 + \frac{1}{c \log^2 n}$. We define the set of *valid integers* to be the set $\{\lceil \alpha^i \rceil : 0 \leq i \leq B\}$, where $B = O(\log^3 n)$ is such that $\alpha^B \geq n$. Roughly speaking, this is the set of integral powers of $\alpha$. In the new dynamic program, we only compute values of the form $P(v, k, s, n_1, \ldots, n_\ell)$ where each $n_i$ is a valid integer. Note that the values of $k$ and $s$ can be *any* integer. Of course, we need to modify the details of the dynamic program computation accordingly. Since we restrict the values of all variables to be *valid integers*, we cannot enforce equality in such cases. Instead, wherever a single variable was equal to the sum of two others in the previous algorithm description, we replace the equality by an inequalities expressing the necessary approximation to within an $\alpha$ factor.

*Internal Node.* Consider a node $v$ at real level $\ell$ with children $v_1$ and $v_2$ at real levels $\ell'$. Note that the value of $\ell'$ is either $\ell$ or $\ell+1$. Note that $P(v, k, s, \ldots)$ is undefined if $p$ is greater than the number of leaves in $T_v$.

$$\begin{aligned}
&P(v, k, s, n_1, n_2, \ldots n_\ell) \\
&= \min\{P(v_1, k_1, s_1, n^{(1)}_1, n^{(1)}_2, \ldots, n^{(1)}_{\ell'}) \\
&\qquad + P(v_2, k_2, s_2, n^{(2)}_1, n^{(2)}_2, \ldots, n^{(2)}_{\ell'})\}
\end{aligned}$$

where the RHS of the above is minimized over all settings of values such that

1. $k = k_1 + k_2$, $k_1$ is not greater than the number of leaves in $T_{v_1}$ and $k_2$ is not greater than the number of leaves in $T_{v_2}$.
2. $\forall i < \ell$, $1/\alpha \leq n_i/(n^{(1)}_i + n^{(2)}_i) \leq 1$.
3. $s_1 = s_{1,2} + s_{1,out}$, $s_2 = s_{2,1} + s_{2,out}$, $s = s_{1,out} + s_{2,out}$, $n^{(1)}_{\ell'} \geq s_{2,1}$ and $n^{(2)}_{\ell'} \geq s_{1,2}$.
4. If $\ell' = \ell$, $1/\alpha \leq (n_\ell + s_{2,1} + s_{1,2})/(n^{(1)}_\ell + n^{(2)}_\ell) \leq 1$.
5. If $\ell' = \ell + 1$, then $n^{(1)}_{\ell'} = n^{(2)}_{\ell'} = 0$.

*Base Case.* This is unchanged from the previous algorithm.

*Running Time.* Note that for each node $v$, we have to compute $(\log^3 n)^{O(\log_\mu n)}$ values. The computation of each value takes $(\log^3 n)^{O(\log_\mu n)}$ time. Thus the entire dynamic programming algorithm runs in $(\log n)^{O(\log_\mu n)}$ time, which is $n^{O(\log\log n / \log \mu)}$, or $n^{O(1/\epsilon)}$ for $\mu = O(\log^\epsilon n)$.

The improved dynamic program computes the optimal solution from among a restricted set of solutions. In order to prove that the dynamic program works, we show that the optimal solution can be converted into the special restricted form considered by the dynamic program. This is achieved by increasing the cost of the solution by at most a constant factor. Further we show that the solution obtained by the dynamic program can indeed be translated to a solution of the original problem without any increase in the cost.

In order to prove that the optimal solution can be translated into the required form, we first prove the following lemma:

LEMMA 4. *For every $v$, $D^*(v, k, s, (1+\delta_1)n_1 \ldots (1+\delta_\ell)n_\ell) \leq (1+\delta)^3 D^*(v, k, s, n_1, \ldots n_\ell)$, where $\delta \geq \delta_i$.*

We begin by proving the following lemma.

LEMMA 5. *For every $v$, $D^*(v, k, s, (1+\delta_1)n_1, \ldots, (1+\delta_\ell)n_\ell) \leq (1+\delta)^2 D^*(v, k, s, n_1, \ldots n_\ell) + \delta \sum_{i=1}^{\ell} n_i \cdot \Delta\mu^{-i}$, where $\delta \geq \delta_i$.*

PROOF. $D^*(v, k, s, n_1, \ldots, n_\ell)$ denotes the value of the optimal solution OPT for $T_v$ with $k$ clusters, excluding up to $s$ nodes from $T_v$ and supporting $n_i$ nodes from level $i$. From this optimal solution, we will construct a solution OPT' for $T_v$ with $k$ clusters, excluding up to $s$ nodes from $T_v$ and supporting $(1+\delta_i)n_i$ nodes from level $i$. Intuitively, if we scale the size of a cluster and increase it by a factor of at most $1+\delta$, the cost of the solution should increase by a factor of $(1+\delta)^2$ (since the cost is quadratic in the cluster size). This intuition is almost correct, with a small corrective factor in the cost upper bound. We will prove a bound on the value of the constructed solution OPT' which will be the required upper bound on $D^*(v,k,s,(1+\delta_1)n_1,\ldots,(1+\delta_\ell)n_\ell)$. In fact, we give a randomized construction for OPT' and bound the expected value of this solution.

We construct the modified solution as follows. Suppose the $k$ clusters are numbered $1,2,\ldots k$. Now, suppose cluster $j$ has $n_{i,j}$ nodes connecting from level $i$. For $i\le\ell$, the $n_{i,j}$ nodes connect from outside subtree $T_v$ and $\sum_{j=1}^k n_{i,j}=n_i$. For $i>\ell$, the $n_{i,j}$ nodes connect from within subtree $T_v$. For $i\le\ell$, let $p_{i,j}=n_{i,j}/n_i$; $\sum_{j=1}^k p_{i,j}=1$. In the solution OPT', cluster $j$ has $n_{i,j}$ nodes connecting from level $i$. This assigns $n_i$ nodes amongst the $(1+\delta_i)n_i$ nodes connecting from level $i$. The remaining $\delta_i n_i$ nodes are assigned to clusters as follows: Each remaining node is assigned to one of the $k$ clusters independently and at random. The probability that it is assigned to cluster $j$ is $p_{i,j}$. Let $R_{i,j}$ be the number of nodes assigned to cluster $j$ in this random process. Note that $\mathbf{E}[R_{i,j}]=\delta_i n_{i,j}$. Thus, $\mathbf{E}[n_{i,j}+R_{i,j}]=(1+\delta_i)n_{i,j}$ Additionally, we define $R_{i,j}$ for $i>\ell$ to be 0. Note that in OPT', for all $i,j$, the number of nodes connecting to cluster $j$ from level $i$ is $n_{i,j}+R_{i,j}$. Also, $\mathbf{E}[n_{i,j}+R_{i,j}]\le(1+\delta)n_{i,j}$. The size of cluster $j$ is $\sum_i n_{i,j}+R_{i,j}$ The cost of cluster $j$ is

$$\left(\sum_i n_{i,j}+R_{i,j}\right)\left(\sum_i (n_{i,j}+R_{i,j})\Delta\mu^{-i}\right) \qquad (1)$$

Notice that the contribution of cluster $j$ to $(1+\delta)^2D^*(v,k,s,n_1,n_2,\ldots n_\ell)$ is

$$(1+\delta)^2\left(\sum_i n_{i,j}\right)\left(\sum_i n_{i,j}\Delta\mu^{-i}\right) \qquad (2)$$

We compare the expressions for the cost of the randomized solution (1) and the scaled optimal cost (2) term by term. The term $(n_{i_1,j}+R_{i_1 j})(n_{i_2,j}+R_{i_2,j})\Delta\mu^{-i_2}$ is compared to $(1+\delta)^2 n_{i_1,j}n_{i_2,j}\cdot\Delta\mu^{-i_2}$. Since $R_{i_1,j}$ and $R_{i_2,j}$ are the only random variables in the first expression and since $\mathbf{E}[n_{i_1,j}+R_{i_1,j}]\le(1+\delta)n_{i_1,j}$ and $\mathbf{E}[n_{i_2,j}+R_{i_2,j}]\le(1+\delta)n_{i_2,j}$, the expected value of the first expression is bounded by the second expression if $\mathbf{E}[R_{i_1,j}\cdot R_{i_2,j}]=\mathbf{E}[R_{i_1,j}]\cdot\mathbf{E}[R_{i_2,j}]$. This is true if $i_1\ne i_2$, since $R_{i_1,j}$ and $R_{i_2,j}$ are independent random variables. This is also true if either $i_1>\ell$ or $i_2>\ell$ since either $R_{i_1,j}$ or $R_{i_2,j}$ is identically zero. The only case that remains to be considered is $i_1=i_2=i$ say and $i\le\ell$. In this case, the expected value of the first expression exceeds the second expression by at most $\left(\mathbf{E}[R_{i,j}^2]-\mathbf{E}[R_{i,j}]^2\right)\Delta\mu^{-i}$. Now, $R_{i,j}$ is the sum of $\delta_i n_i$ i.i.d. 0-1 random variables each of which is 1 with probability $p_{i,j}$. Hence, $\mathbf{E}[R_{i,j}^2]-\mathbf{E}[R_{i,j}]^2\le\delta_i n_i\cdot p_{i,j}=\delta_i n_{i,j}$. Thus the expected value of the first expression exceeds the second expression by at most $\delta_i n_{i,j}\cdot\Delta\mu^{-i}$. Summing this excess over all levels $i\le\ell$ and clusters $j$, we get that the expected value of the cost of the randomized solution (1) exceeds the scaled optimum cost (2) by at most

$$\sum_{i=1}^{\ell}\sum_{j=1}^{k}\delta_i n_{i,j}\cdot\Delta\mu^{-i}\le\delta\sum_{i=1}^{\ell}n_i\cdot\Delta\mu^{-i}$$

□

PROOF OF LEMMA 4. From Lemma 5, we know that

$$\begin{aligned}&D^*(v,k,s,(1+\delta_1)n_1,\ldots,(1+\delta_\ell)n_\ell)\\&\le(1+\delta)^2D^*(v,k,s,n_1,\ldots n_\ell)+\delta\sum_{i=1}^{\ell}n_i\cdot\Delta\mu^{-i}.\end{aligned}$$

Now,

$$\sum_{i=1}^{\ell}n_i\cdot\Delta\mu^{-i}\le D^*(v,k,s,n_1,\ldots,n_\ell).$$

This follows from the fact that each of the $n_i$ nodes connecting from level $i$ contribute at least $\Delta\mu^{-i}$ to the cost of the solution.

$$\begin{aligned}&D^*(v,k,s,(1+\delta_1)n_1,\ldots,(1+\delta_\ell)n_\ell)\\&\le((1+\delta)^2+\delta)D^*(v,k,s,n_1,\ldots,n_\ell)\\&\le(1+\delta)^3D^*(v,k,s,n_1,\ldots,n_\ell)\end{aligned}$$

□

Let $h(v)$ be the height of node $v$ in the HST, that is $h(v)$ is the longest distance from node $v$ to a leaf descendant in the real binary tree. The following lemma implies that the solution computed by the dynamic programming $P$ is within a constant factor of that of $D^*$.

LEMMA 6. *For every node $v$ and parameters $k,s,n_1,\ldots,n_\ell$, there exist $\tilde{n}_1,\ldots\tilde{n}_\ell$ which are valid for $P$ such that* $P(v,k,s,\tilde{n}_1,\ldots,\tilde{n}_\ell)\le\alpha^{3(h(v)+1)}D^*(v,k,s,n_1,\ldots,n_\ell)$

We begin by proving the following lemma.

LEMMA 7. *For every node $v$ and parameters $k,s,n_1\ldots n_\ell$, (which are valid for $P$)*
$P(v,k,s,n_1,\ldots,n_\ell)\le\alpha^{3h(v)}D^*(v,k,s,n_1,\ldots,n_\ell)$

PROOF. The proof is obtained by induction on the height $h(v)$ of node $v$. The base case corresponds to a leaf $v$. In this case, similar to the base case in the proof of Lemma 3, $P(v,k,s,n_1,\ldots,n_\ell)=D^*(v,k,s,n_1,\ldots,n_\ell)$

Consider node $v$ with height $h(v)=j$. By the inductive hypothesis, the lemma is true for all nodes of height $j-1$. Let $v_1$ and $v_2$ be the children of node $v$. Note that $v_1$ and $v_2$ have heights $j-1$. Let $f_\alpha(x)$ denote the smallest valid integer greater than or equal to $x$.

Suppose

$$\begin{aligned}D^*(v,k,s,n_1,\ldots,n_\ell)=&D^*(v_1,k_1,s_1,n_1^{(1)},\ldots,n_{\ell'}^{(1)})\\&+D^*(v_2,k_2,s_2,n_1^{(2)},\ldots,n_{\ell'}^{(2)}).\end{aligned}$$

Then, the computation of $P(v,k,s,n_1,\ldots,n_\ell)$ considers the term $P(v_1,k_1,s_1,f_\alpha(n_1^{(1)}),\ldots,f_\alpha(n_{\ell'}^{(1)}))+P(v_2,k_2,s_2,f_\alpha(n_1^{(2)}),\ldots,f_\alpha(n_{\ell'}^{(2)}))$. The conditions checked by the dynamic program are satisfied as follows: (the interpretation of the various parameters is similar to that in the proof of Lemma 3)

1. $k=k_1+k_2$, $k_1$ is at most the number of leaves in $T_{v_1}$ and $k_2$ is at most the number of leaves in $T_{v_2}$.

2. $\forall i<\ell'$, $n_i^{(1)}+n_i^{(2)}=n_i$. This implies that $f_\alpha(n_i^{(1)})+f_\alpha(n_i^{(2)})\ge n_i$ and $f_\alpha(n_i^{(1)})+f_\alpha(n_i^{(2)})\le\alpha\cdot n_i$.

3. $s_1 = s_{1,2} + s_{1,out}$, $s_2 = s_{2,1} + s_{2,out}$, $s = s_{1,out} + s_{2,out}$, $n_{\ell'}^{(1)} \geq s_{2,1}$ and $n_{\ell'}^{(2)} \geq s_{1,2}$.

4. If $\ell' = \ell$, $n_\ell^{(1)} + n_\ell^{(2)} = n_\ell + s_{2,1} + s_{1,2}$. This implies that $f_\alpha(n_\ell^{(1)}) + f_\alpha(n_\ell^{(2)}) \geq n_\ell + s_{2,1} + s_{1,2}$ and $f_\alpha(n_\ell^{(1)}) + f_\alpha(n_\ell^{(2)}) \leq \alpha(n_\ell + s_{2,1} + s_{1,2})$

By the inductive hypothesis,

$$\begin{aligned} &P(v_1, k_1, s_1, f_\alpha(n_1^{(1)}), \ldots, f_\alpha(n_{\ell'}^{(1)})) \\ &\leq \alpha^{3(j-1)} D^*(v_1, k_1, s_1, f_\alpha(n_1^{(1)}), \ldots, f_\alpha(n_{\ell'}^{(1)})) \\ &\leq \alpha^{3j} D^*(v_1, k_1, s_1, n_1^{(1)}, \ldots, n_{\ell'}^{(1)}) \end{aligned}$$

The last inequality follows from Lemma 4. Similarly,

$$\begin{aligned} &P(v_2, k_2, s_2, f_\alpha(n_1^{(2)}), \ldots, f_\alpha(n_{\ell'}^{(2)})) \\ &\leq \alpha^{3j} D^*(v_2, k_2, s_2, n_1^{(2)}, \ldots, n_{\ell'}^{(2)}) \end{aligned}$$

Hence,

$$\begin{aligned} P(v, k, s, n_1, \ldots, n_\ell) &\leq P(v_1, k_1, s_1, f_\alpha(n_1^{(1)}), \ldots, f_\alpha(n_{\ell'}^{(1)})) \\ &\quad + P(v_2, k_2, s_2, f_\alpha(n_1^{(2)}), \ldots, f_\alpha(n_{\ell'}^{(2)})) \\ &\leq \alpha^{3j} D^*(v_1, k_1, s_1, n_1^{(1)}, \ldots, n_{\ell'}^{(1)}) \\ &\quad + \alpha^{3j} D^*(v_2, k_2, s_2, n_1^{(2)}, \ldots, n_{\ell'}^{(2)}) \\ &= \alpha^{3j} D^*(v, k, s, n_1, \ldots, n_\ell) \end{aligned}$$

□

Now, let $f_\alpha(x)$ denote the smallest valid integer greater than or equal to $x$. Note that $x \leq f_\alpha(x) \leq \alpha \cdot x$.

Proof of Lemma 6. From Lemma 7 we have

$$\begin{aligned} &P(v, k, s, f_\alpha(n_1), \ldots, f_\alpha(n_\ell)) \\ &\leq \alpha^{3h(v)} D^*(v, k, s, f_\alpha(n_1), \ldots, f_\alpha(n_\ell)) \\ &\leq \alpha^{3h(v)} \alpha^3 D^*(v, k, s, n_1, \ldots, n_\ell), \end{aligned}$$

using lemma 4. □

The following lemma states that the value $P(\ldots)$ computed by the dynamic program is reasonable, i.e. there is indeed a solution whose cost is at most the value computed by the dynamic program.

Lemma 8. *For every node $v$, and parameters $k, s, n_1, \ldots, n_\ell$, (which are valid for $P$) the dynamic program computes a correct balanced $k$-median solution of cost at most $P(v, k, s, n_1, \ldots, n_\ell)$.*

The proof proceeds by induction on the height of the node $v$. We need to verify that the conditions imposed by the dynamic program in the computation of $P(\ldots)$ ensure that for every node $v$ with children $v_1$ and $v_2$, valid solutions for subtrees $T_{v_1}$ and $T_{v_2}$ are combined correctly to obtain a valid solution for $T_v$. The interpretation of the various parameters used in the dynamic program computation is similar to the interpretation in the proof of Lemma 3. We omit the details in this extended abstract.

Putting Lemma 6 and Lemma 8 together, we get the following guarantee on the final approximation ratio achieved.

Theorem 2. *The algorithm computes an $O(\frac{1}{\epsilon}\log^{1+\epsilon} n)$-approximate solution for the* MIN-SUM $k$-CLUSTERING *problem in time $n^{O(1/\epsilon)}$.*

Proof. From Lemma 6, the value of the solution computed by the dynamic program is at most a factor $\alpha^{O(hi)}$ away from the value of the optimal solution on the tree where $h$ is the height of the tree. The height $h$ is $O(\log^2 n)$ and $\alpha = 1 + \frac{1}{c \log^2 n}$. Hence, the solution computed by the dynamic program is at most a constant factor away from the optimal solution on the tree. Since the algorithm uses a $\mu$-HST where $\mu = O(\log^\epsilon n)$, the approximation ratio is $O(\mu \log_\mu n \log\log n) = O(\frac{1}{\epsilon}\log^{1+\epsilon} n)$. As stated before, the running time is $n^{O(1/\epsilon)}$. □

# 4. BICRITERIA APPROXIMATION

We will reformulate the *balanced $k$-median* problem on graph $G$ as a capacitated $k$-median problem on a bipartite graph $G'(A, B)$ (derived from $G$). Further, the distance function we define on $G'$ will not satisfy triangle inequality. In fact, we will devise a series of different problem formulations, each with a slightly different distance function on $G'$.

We begin with the definition of $G'$. For every potential center $i$ in the original problem, we introduce auxiliary vertices $i^{(1)}, i^{(2)}, \ldots i^{(n)}$ which collectively form set $A$. In addition, $G'$ has copies of the original vertices of $G$, which collectively form set $B$.

In the *balanced $k$-median*, if $j$ is connected to center $i$ and is part of a cluster of size $r$, the contribution to the objective function is $r \cdot c_{ij}$. In order to simulate this, we set the distance of $j$ from $i^{(r)}$ to be $r \cdot c_{ij}$. Note that the distances as defined do not form a metric. Further, the capacity of $i^{(r)}$ is $r$, i.e. a center at $i^{(r)}$ can serve at most $r$ vertices. The problem is to choose $p$ centers from amongst the vertices in $A$ and connect each vertex in $B$ to a center in $A$ so that the capacity constraints are satisfied. The objective function is the sum of distances of vertices in $B$ to their assigned centers.

The correspondence between solutions to the *balanced $k$-median* problem on $G$ and solutions to the capacitated $k$-median problem on $G'$ is as follows: If $i$ is chosen as a center for a cluster of size $r$, then $i^{(r)}$ is chosen as a center in $G'$. Note that multiple centers can be built at the same vertex $i^{(r)}$. The significance of $i^{(r)}$ being chosen as a center is that it defines a cluster of size at most $r$; it is not necessary that $i$ belong to this cluster. It is easy to see that the original problem is equivalent to the capacitated $k$-median problem on $G'$.

Lemma 9. *For every solution to the* BALANCED $k$-MEDIAN PROBLEM *on $G$, there is a solution to the capacitated $k$ median problem on $G'$ of the same cost. For every solution to the capacitated $k$ median problem on $G'$, of cost $c$, there is a solution to the* BALANCED $k$-MEDIAN *problem on $G$, of cost at most $c$.*

Proof. Consider a solution to the BALANCED $k$-MEDIAN PROBLEM. Suppose vertex $i$ is chosen as a center of a cluster of size $r$ consisting of vertices $j_1, j_2, \ldots, j_r$. Then $i^{(r)}$ is chosen as a center in $G'$ and vertices $j_1, \ldots, j_r$ are connected to it in the capacitated $k$-median solution. Note that the capacity constraint is satisfied and the contribution to the cost in both cases is $r \sum_{t=1}^{r} c_{ij_t}$.

Suppose we have a solution to the capacitated $k$-median problem. We will construct a solution to the original BAL-

ANCED $k$-MEDIAN PROBLEM as follows. Suppose $i^{(r)}$ is chosen as a center with vertices $j_1, \ldots, j_{r'}$ connected to it ($r' \leq r$). Then in the BALANCED $k$-MEDIAN solution, $i$ is chosen as a center and vertices $j_1, \ldots, j_{r'}$ are connected to it. The contribution of this cluster to the cost of the BALANCED $k$-MEDIAN solution is $r' \sum_{t=1}^{r'} c_{ij_t} \leq r \sum_{t=1}^{r'} c_{ij_t}$, which is the contribution to the cost of the capacitated $k$-median problem. $\square$

We present an LP relaxation (LP1) for the capacitated $k$-median problem described earlier.

$$\forall\, ij \quad \mathbf{x}_{ij}^{(r)} \leq \mathbf{y}_i^{(r)} \tag{3}$$

$$\forall\, j \quad \sum_r \sum_i \mathbf{x}_{ij}^{(r)} \geq 1 \tag{4}$$

$$\forall\, i, r \quad \sum_j \mathbf{x}_{ij}^{(r)} \leq r \cdot \mathbf{y}_i^{(r)} \tag{5}$$

$$\sum_r \sum_i \mathbf{y}_i^{(r)} \leq k \tag{6}$$

$$\min \sum_r \sum_{ij} \mathbf{x}_{ij}^{(r)} \cdot r \cdot c_{ij} \tag{7}$$

0-1 solutions of this LP correspond to feasible solutions of the capacitated $k$-median problem on $G'$. The variable $\mathbf{y}_i^{(r)}$ indicates whether there is a center at $i$ for a cluster of size $r$, i.e. a center at $i^{(r)}$ in $G'$. The variable $\mathbf{x}_{ij}^{(r)}$ indicates whether $j$ is assigned to center $i$ of a size $r$ cluster, i.e. assigned to $i^{(r)}$ in $G'$. Consider a feasible solution to the capacitated $k$-median problem on $G'$. In the corresponding 0-1 LP solution, $\mathbf{x}_{ij}^{(r)} = 1$ iff $j$ is assigned to center $i^{(r)}$; the contribution to the objective function is $r \cdot c_{ij}$.

Suppose $C^*$ is the optimal cost of a solution to the BALANCED $k$-MEDIAN problem on $G$.

LEMMA 10.

$$OPT_{LP1} \leq C^*.$$

## 4.1 A capacitated facility location problem

We do not know how to solve the above capacitated $k$-median problem directly. Instead, we will solve a slightly different problem – a facility location type problem and obtain a bicriteria approximation. Our bicriteria approximation is obtained in the style of the bicriteria approximations of Marathe *et al.* [8].

Suppose we have an optimization problem with two objective functions $A$ and $B$. We wish to minimize $A$ subject to the constraint $B \leq B^*$. Say the optimal solution has $A = A^*$, $B = B^*$. Then a $(p, q)$ bicriteria approximation is a solution with $A \leq p \cdot A^*$ and $B \leq q \cdot B^*$. We consider the optimization problem of minimizing $A + \lambda B$ where $\lambda$ is a scalar. Suppose we have a $\sigma$ approximation for this problem. Then we can get a bicriteria approximation as follows: Set $\lambda = \frac{A^*}{\gamma B^*}$. The approximation algorithm returns a solution such that:

$$\begin{aligned}
A + \lambda B &\leq \sigma(A^* + \lambda B^*) \\
\lambda &= \frac{A^*}{\gamma B^*} \\
A &\leq \sigma\left(A^* + \lambda B^*\right) = \sigma\left(1 + \frac{1}{\gamma}\right) A^* \\
B &\leq \sigma\left(\frac{A^*}{\lambda} + B^*\right) = \sigma(1+\gamma) B^*
\end{aligned}$$

This gives a $(\sigma(1 + \frac{1}{\gamma}), \sigma(1 + \gamma))$ bicriteria approximation. For most problems, the correct value of $\lambda$ can be guessed to within a factor of $(1+\epsilon)$. This then yields the same bicriteria approximation within $(1 + \epsilon)$ factors.

Employing these ideas, we now present an LP relaxation (LP2) for a facility location analog of the capacitated $k$-median problem we defined earlier. Here, we remove the constraint (6) on the number of centers. Instead, we add a cost of $z$ to the objective function for each center used by the solution ($z$ is a parameter we will fix later).

$$\forall\, ij \quad \mathbf{x}_{ij}^{(r)} \leq \mathbf{y}_i^{(r)} \tag{8}$$

$$\forall\, j \quad \sum_r \sum_i \mathbf{x}_{ij}^{(r)} \geq 1 \tag{9}$$

$$\forall\, i, r \quad \sum_j \mathbf{x}_{ij}^{(r)} \leq r \cdot \mathbf{y}_i^{(r)} \tag{10}$$

$$\min \sum_r \sum_{ij} \mathbf{x}_{ij}^{(r)} \cdot r \cdot c_{ij} + \sum_r \sum_i \mathbf{y}_i^{(r)} \cdot z \tag{11}$$

The following lemma is straightforward.

LEMMA 11.

$$OPT_{LP2} \leq OPT_{LP1} + k \cdot z \leq C^* + k \cdot z$$

We will use the primal dual algorithm of Jain and Vazirani [6] to solve this facility location problem. We first make a slight detour and give an overview of their algorithm, since we will use some details of their analysis later.

The primal-dual algorithm produces a solution to the facility location problem together with a feasible solution to the dual of the LP relaxation. The value of the dual serves as a lower bound on the optimal solution. The algorithm works in two phases. The first phase grows dual variables $\alpha_j$ for each vertex $j$ and $\beta_{ij}$ for each edge $ij$. All variables are initially 0. The variables $\alpha_j$ are grown uniformly. At various points of time, some of the dual variables $\alpha_j$ stop growing in order to maintain feasibility of the dual solution. As long as $\alpha_j \leq c_{ij}$, the value of $\beta_{ij}$ is 0. Suppose $\alpha_j > c_{ij}$, then $\beta_{ij}$ is set to $\alpha_j - c_{ij}$; such an edge $ij$ is called *saturated.* As this process continues, the following events occur:

1. $\sum_j \beta_{ij} = z$.
   In this case we say that facility $i$ is *paid for.* For all $j$ with $\beta_{ij} > 0$, $\alpha_j$ stops growing and $i$ is said to be the *connecting witness* of $j$ (provided $j$ has not been assigned a connecting witness already). Let $t_i$ be the time at which facility $i$ gets paid for.
2. $\alpha_j = c_{ij}$ and $i$ is paid for.
   In this case, $\alpha_j$ stops growing and $i$ is said to be a connecting witness of $j$.

When several events occur at the same time, they are dealt with an arbitrary order. The first phase terminates when all demand points are assigned connecting witnesses. The second phase is a cleanup phase. We pick facility $i$ that is paid for the earliest, delete facilities reachable from $i$ by two saturated edges and repeat this process. A demand point $j$ is said to be *directly connected* if there exists a facility $i$ in the final solution such that $\beta_{ij} > 0$; $j$ is assigned to $i$ in this case. (Note that there can be at most one such facility $i$ because of the cleanup step). On the other hand, if there is no facility $i$ in the final solution such that $\beta_{ij} > 0$, then the demand point $j$ is either assigned to its connecting witness or to the facility that caused the deletion of its connecting witness in the cleanup step. Such demand points are called *indirectly connected*. (Though the original Jain Vazirani algorithm examined facilities in the order that they were paid for, it turns out that we can examine facilities in an arbitrary order in the cleanup step for their analysis to go through and this will be important for us).

The analysis of the algorithm yields:

$$C + 3F \leq 3 \sum_j \alpha_j \leq 3OPT.$$

Here $C$ is the assignment cost and $F$ is the facility cost of the solution produced by the algorithm. We will need to modify the algorithm to deal with capacities and the fact that the distances do not satisfy triangle inequality.

## Dealing with non-metricity

We now explain how we deal with the fact that distances do not satisfy triangle inequality (ignoring capacities for now).

Note that the algorithm itself does not require the triangle inequality in order to run it. However the triangle inequality is used in the analysis of an indirectly connected demand point. Suppose facility $i_1$ is deleted by $i_2$ and demand point $j_1$ whose connecting witness is $i_1$ is indirectly connected to $i_2$. We first review the facts used in the analysis in [6] and later explain how these facts can be made to hold in our case as well so that essentially the same analysis goes through. The analysis of the assignment cost of $j_1$ proceeds as follows:

$$\begin{aligned} c_{i_2 j_1} &\leq c_{i_2 j_2} + c_{i_1 j_2} + c_{i_1 j_1} \\ &\leq \alpha_{j_2} + \alpha_{j_2} + \alpha_{j_1} \leq 3\alpha_{j_1} \end{aligned}$$

The first step follows from the triangle inequality and the remaining follow from the properties of the algorithm.

Since we do not have triangle inequality, we need the following condition for this step of the analysis to go through:

$$r_2 \cdot c_{i_2 j_1} \quad \leq \quad r_2 \cdot c_{i_2 j_2} + r_1 \cdot c_{i_1 j_2} + r_1 \cdot c_{i_1 j_1}.$$

However, this is not true in general, but this is true if $r_2 \leq r_1$. This implies that for the analysis to go through, if $i_1^{(r_1)}$ is deleted by $i_2^{(r_2)}$, then $r_2 \leq r_1$. This can be achieved by the following modified cleanup step:

We pick the paid for facility $i^{(r)}$ with the smallest value of $r$, delete all facilities reachable from $i^{(r)}$ by two saturated edges and repeat. Demand point reassignments are performed as before.

We can verify that the rest of the required properties go through as a result of the modified cleanup step. We refer the reader to [6] for the details of how these inequalities are used in the proof of the approximation guarantee. Recall that $t_{i1}$ is the time at which facility $i1$ gets paid for.

$$\begin{array}{rcl} i_1 j_2 \text{ saturated } \Rightarrow \beta_{i_1 j_2} > 0 & \Rightarrow & \alpha_{j_2} \leq t_{i_1} \\ i_1 \text{ connecting witness for } j_1 & \Rightarrow & \alpha_{j_1} \geq t_{i_1} \geq \alpha_{j_2} \\ r_2 \cdot c_{i_2 j_2} & \leq & \alpha_{j_2} \\ r_2 \cdot c_{i_1 j_2} & \leq & \alpha_{j_2} \\ r_1 \cdot c_{i_1 j_1} & \leq & \alpha_{j_1} \end{array}$$

## Dealing with capacities

Jain and Vazirani [6] explain how to use their primal dual algorithm in order to solve a capacitated facility location variant where a number of facilities can be built at the same location. We employ their ideas for our problem. The idea is to modify the distance function by adding an additive term and then applying the uncapacitated algorithm to this modified problem. For every demand node $j$ that is connected to a facility $i$ of capacity $r$, we increase the distance of $j$ to $i$ by a term of the form $z/r$. (Recall that $z$ is the cost of the facility).

The uncapacitated facility location objective charges a cost of $z$ for every facility in the solution irrespective of the number of demand points connected to the facility. On the other hand the capacitated facility location objective charges a cost of $z$ for every $r$ demand points connected to the facility. The modification in the distance function ensures that we pay roughly $z$ for every $r$ demand points that connect to the facility. In the solution produced by the uncapacitated algorithm, if $t$ nodes are connected to facility $i^{(r)}$, then we build $\lceil \frac{t}{r} \rceil$ centers at $i^{(r)}$.

We need to verify that the modification in the distance function satisfies the triangle inequality condition required in the analysis of an indirectly connected facility. Indeed this is the case:

$$\begin{aligned} r_2 \cdot c_{i_2 j_1} + z/r_2 \quad \leq & \quad r_2 \cdot c_{i_2 j_2} + z/r_2 \\ + & \quad r_1 \cdot c_{i_1 j_2} + z/r_1 \\ + & \quad r_1 \cdot c_{i_1 j_1} + z/r_1 \end{aligned}$$

We now explain how we derive an instance of uncapacitated facility location with a modified distance function by changing the LP formulation LP2. We remove the capacity constraint (10)

$$\sum_j \mathbf{x}_{ij}^{(r)} \leq r \cdot \mathbf{y}_i^{(r)} \Rightarrow \mathbf{y}_i^{(r)} \geq \frac{\sum_j \mathbf{x}_{ij}^{(r)}}{r}.$$

Further, we replace $\mathbf{y}_i^{(r)}$ in the objective function
$$\sum_r \sum_{ij} \mathbf{x}_{ij}^{(r)} \cdot r \cdot c_{ij} + \sum_r \sum_i \mathbf{y}_i^{(r)} \cdot z$$

by the convex combination $\alpha \sum_i \frac{\mathbf{x}_{ij}^{(r)}}{r} + (1-\alpha)\mathbf{y}_i^{(r)}$.

Here $\alpha$ is a parameter that will be determined later.

These modifications give rise to the following LP (LP3):

$$\begin{aligned} &\forall\, ij \quad \mathbf{x}_{ij}^{(r)} \leq \mathbf{y}_i^{(r)} \\ &\forall\, j \quad \sum_r \sum_i \mathbf{x}_{ij}^{(r)} \geq 1 \\ &\min \sum_r \sum_{ij} \mathbf{x}_{ij}^{(r)} \left( r \cdot c_{ij} + \frac{\alpha \cdot z}{r} \right) + \sum_i \mathbf{y}_i^{(r)} \cdot (1-\alpha) z \end{aligned}$$

Notice that LP3 corresponds to an uncapacitated facility location problem on $G'$. The facilities are the vertices $i^{(r)}$ with facility cost $(1-\alpha)z$. The distance of $j$ from $i^{(r)}$ is $r \cdot c_{ij} + \alpha \cdot z/r$, of the same form as introduced earlier.

LEMMA 12.

$$OPT_{LP3} \leq OPT_{LP2} \leq C^* + k \cdot z.$$

## 4.2 Analysis of bicriteria approximation

We now use the primal dual algorithm to solve this uncapacitated facility location problem. We will abuse notation slightly and let $\mathbf{x}_{ij}^{(r)}$ indicate whether $j$ is connected to $i^{(r)}$ in the solution produced by the algorithm and $\mathbf{y}_i^{(r)}$ denote whether a facility is built at $i$ in the solution. The analysis of the primal dual algorithm gives the following guarantee:

$$\sum_r \sum_{ij} \mathbf{x}_{ij}^{(r)} \left( r \cdot c_{i,j} + \frac{\alpha \cdot z}{r} \right) + 3 \sum_i \mathbf{y}_i^{(r)} \cdot (1-\alpha) z \leq 3(C^* + k \cdot z)$$

Suppose $t$ nodes are connected to $i^{(r)}$ in the uncapacitated solution. Then, we build $\left\lceil \frac{t}{\beta \cdot r} \right\rceil$ centers at $i^{(r)}$. ($\beta$ is a parameter we will specify later). Let $C$ be the cost of the solution produced, let $k'$ be the number of centers (clusters) in the solution and let $\mathbf{Y}_i^{(r)}$ be the number of centers at $i^{(r)}$.

$$\begin{aligned} \mathbf{Y}_i^{(r)} &\leq \sum_j \frac{\mathbf{x}_{ij}^{(r)}}{\beta \cdot r} + \mathbf{y}_i^{(r)} \\ k' &= \sum_{i,r} \mathbf{Y}_i^{(r)} \\ C &\leq 2\beta \sum_r \sum_{ij} \mathbf{x}_{ij}^{(r)} \cdot r \cdot c_{ij} \end{aligned}$$

Note that in obtaining the bound for the cost $C$ of the solution, we multiply the sum of distances of points to $i^{(r)}$ by the size of the cluster $\beta r$. A further factor of 2 comes from moving to the original MIN-SUM $k$-CLUSTERING objective function.

Using (12), we get the following bound in terms of $C$ and $k'$.

$$\frac{C}{2\beta} + \min(3(1-\alpha), \alpha\beta) \cdot k' \cdot z \leq 3(C^* + k \cdot z) \tag{12}$$

Set $\alpha\beta = 3(1-\alpha)$, i.e. $\beta = 3\left(\frac{1}{\alpha} - 1\right)$. Further, set $z = \frac{C^*}{\gamma k}$. (Actually, we guess this value within $1+\epsilon$, but we will assume for simplicity that we guess it exactly). Plugging in these values, we get the following bounds on $C$ and $k'$.

$$C \leq 18\left(1 + \frac{1}{\gamma}\right)\left(\frac{1}{\alpha} - 1\right) C^* = C_{\max} \tag{13}$$

$$k' \leq \frac{(1+\gamma)}{(1-\alpha)} k = k'_{\max} \tag{14}$$

In fact, we can obtain a slightly improved bound on $C$ by making a small change in the solution in the end. We omit the details in this extended abstract and state the final result.

$$C \leq 19.30 \left(\frac{1+\delta}{\delta^2}\right) C^*, \qquad k' \leq k \cdot (1+\delta)$$

Here $\delta$ is a tradeoff parameter. For example, the bounds imply that we can get a 6.03 approximation using $5k$ clusters.

## 5. CONCLUDING REMARKS

The results in this paper make two significant contributions to the min-sum $k$-clustering problem. We give the first non-trivial approximation algorithm for the problem. The algorithm is based on metric embeddings into HSTs. The techniques used for the solution in HSTs may serve as a basis for approximating other similar problems, using the HST metric embeddings.

We also give a bicriteria algorithm for the problem. We note that our current knowledge of the approximability of the MIN-SUM $k$-CLUSTERING problem is similar to the status of approximability of the $k$-median problem before constant factor approximations were developed for it. This suggests that a constant factor approximation may be possible for MIN-SUM $k$-CLUSTERING as well.

Our work on the bicriteria approximation lead to an interesting generalization of the Jain Vazirani [6] primal dual facility location algorithm. We can get a 3 approximation for a generalization of facility location that we call FACILITY LOCATION WITH CENTER MULTIPLIERS. Here the cost of assigning demand node $j$ to a center at $i$ is $f(i) \cdot c_{ij}$ where the distance $c_{ij}$ forms a metric and $f(i)$ is an arbitrary function of $i$. It is interesting to note that while the same guarantee that applies to the facility location algorithm carries over to the generalization, we do not know how to use this to obtain a constant factor approximation for the corresponding $k$-median generalization, i.e. $k$-MEDIAN WITH CENTER MULTIPLIERS. Nevertheless, we believe that this is an interesting generalization of facility location and will have other applications. Note that our generalization carries over to the capacitated facility location variant where multiple facilities are permitted at the same location (the center multipliers need not have any relationship with the capacities).

## 6. REFERENCES


[1] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic application. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 184–193, 1996.

[2] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 161–168, 1998.

[3] M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. Plotkin. Approximating an arbitrary metric by a small number of tree metrics. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 379–388, 1998.

[4] N. Guttman-Beck and R. Hassin. Approximation algorithms for min-sum $p$-clustering. *Discrete Applied Mathematics.*

[5] P. Indyk. A sublinear time approximation scheme for clustering in metric spaces. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 154–159, 1999.

[6] K. Jain and V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 2–13, 1999. to appear in Journal of the ACM.

[7] V. Kann, S. Khanna, J. Lagergren, and A. Panconesi. On the hardness of max $k$-cut and its dual. In *Israeli Symposium on Theoretical Computer Science*, pages 61–67, 1996.

[8] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt. Bicriteria network design problems. *Journal of Algorithms*, 28(1):142–171, 1998.

[9] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23:555–566, 1976.

[10] L. Schulman. Clustering for edge-cost minimization. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, 2000.