Approximation Algorithms for Min-Sum k-Clustering and Balanced k-Median

Babak Behsaz[1], Zachary Friggstad[1], Mohammad R. Salavatipour[1⋆], and Rohit Sivakumar[1]

Department of Computing Science, University of Alberta
{behsaz,zacharyf,mreza,rohit2}@ualberta.ca

Abstract. We consider two closely related fundamental clustering problems in this paper. In the *Min-Sum k-Clustering* problem, one is given a metric space and has to partition the points into k clusters while minimizing the total pairwise distances between the points assigned to the same cluster. In the *Balanced k-Median* problem, the instance is the same and one has to obtain a partitioning into k clusters $C_1, \ldots, C_k$, where each cluster C_i has a center c_i, while minimizing the total assignment costs for the points in the metric; here the cost of assigning a point j to a cluster C_i is equal to $|C_i|$ times the distance between j and c_i in the metric.

In this paper, we present an $O(\log n)$-approximation for both these problems where n is the number of points in the metric that are to be served. This is an improvement over the $O(\epsilon^{-1}\log^{1+\epsilon} n)$-approximation (for any constant $\epsilon > 0$) obtained by Bartal, Charikar, and Raz [STOC '01]. We also obtain a quasi-PTAS for Balanced k-Median in metrics with constant doubling dimension.

As in the work of Bartal et al., our approximation for general metrics uses embeddings into tree metrics. The main technical contribution in this paper is an $O(1)$-approximation for Balanced k-Median in hierarchically separated trees (HSTs). Our improvement comes from a more direct dynamic programming approach that heavily exploits properties of standard HSTs. In this way, we avoid the reduction to special types of HSTs that were considered by Bartal et al., thereby avoiding an additional $O(\epsilon^{-1}\log^{\epsilon} n)$ loss.

1 Introduction

One of the most ubiquitous problems encountered in computing science is clustering. At a high level, a clustering problem arises when we want to aggregate data points into groups of similar objects. Often, there are underlying metric distances $d(u, v)$ between data points u, v that quantify their similarities. Ideally, we want to cluster the objects into few clusters while ensuring that the distances within a cluster are small.

⋆ Supported by NSERC.

In this paper, we focus on two closely related problems, which are referred to in the literature as *Min-Sum k-clustering* (MSkC) and *Balanced k-Median* (BkM). In both problems, we are given a metric space over a set of n points V, which we assume is given as a graph $G = (V, E)$ with metric distances $d(u, v)$ between any two vertices $u, v \in V$. In the MSkC problem the goal is to partition the points V into k clusters $C_1, \ldots, C_k$ to minimize the sum of pair-wise distances between points assigned to the same cluster: $\sum_{i=1}^{k} \sum_{\{j,j'\} \subseteq C_i} d(j, j')$.

This problem (MSkC) was first introduced by Sahni and Gonzalez [14] and is the complement of the Max k-Cut problem. Bartal et al. [4] gave an $O(\epsilon^{-1} \log^{\epsilon} n)$-approximation for any constant $\epsilon > 0$, for the case of Hierarchically Separated Trees (HSTs), which in turn (using the $O(\log n)$ bound for approximating metrics using HSTs [8]), gives an $O(\epsilon^{-1} \log^{1+\epsilon} n)$-approximation for general metrics. To do this, Bartal et al. introduced BkM, where the input is the same as MSkC and the goal is to select k points $c_1, \ldots, c_k \in V$ as the centers of the clusters and partition the nodes V into clusters $C_1, \ldots, C_k$ to minimize $\sum_{i=1}^{k} |C_i| \sum_{v \in C_i} d(v, c_i)$. The multiplier $|C_i|$ on the contribution of $d(v, c_i)$ to the objective function penalizes clusters for being too large, hence the term *balanced*. As observed in [4], it is easy to show that an α-approximation for either MSkC or BkM implies a 2α-approximation for the other problem in metric graphs. The approximation of [4] for MSkC was obtained by presenting such an approximation for BkM.

1.1 Related Work

The facility location interpretation of the BkM leads to a natural generalization of the problem. In this generalization, we are given a set of clients $C \subseteq V$ and a set of facilities $F \subseteq V$. We need to choose k facilities from F to open and the clients in C must be served by these k facilities. In other words, the set of clients must be partitioned into k clusters and the center assigned to each partition must be chosen from F. Note that C and F can have common vertices. The special case that $C = F = V$ is the original problem we defined. We often use the term "facility" to refer to the center of a cluster in BkM and the points assigned to that center are the "clients" that get served by that facility.

The $O(\epsilon^{-1} \log^{1+\epsilon} n)$-approximation of [4] stands as the best approximation for both MSkC and BkM after thirteen years. They also describe a bicriteria $O(1)$-approximation (for BkM) that uses $O(k)$ clusters. Fernandez de la Vega et al. [9] gave a $(1+\epsilon)$-approximation for MSkC with running time of $O(n^{3k} 2^{\epsilon^{-k^2}})$.

BkM and MSkC have been further studied in more restricted settings. BkM can be solved in time $n^{O(k)}$ by "guessing" the center locations and their capacities, and then finding a minimum-cost assignment from the clients to these centers [10]. This yields a 2-approximation for MSkC when k is regarded as a constant. Furthermore, Indyk gives a PTAS [11] for MSkC when $k = 2$.

The factor-2 reduction between BkM and MSkC fails to hold when the distances are not in a metric space. Indeed, one can still solve non-metric instances of BkM in $n^{O(k)}$ time, however no $n^{2-\epsilon}$-approximation is possible for non-metric MSkC for any constant $\epsilon > 0$ and any $k \geq 3$ [12]. An $O(\sqrt{\log n})$-approximation

for non-metric MSkC for $k = 2$ is known as this is just a reformulation of the Minimum Uncut problem [1].

These problems have been studied in geometric spaces as well. For point sets in $\mathbb{R}^d$ and a constant k Schulman [15] gave an algorithm for MSkC that either outputs a $(1+\epsilon)$-approximation, or a solution that agrees with the optimum clustering on $(1-\epsilon)$-fraction of the points but may have a much larger than optimum cost. Finally, Czumaj and Sohler [7] have developed a $(4+\epsilon)$-approximation algorithm for MSkC for the case when $k = o(\log n/\log\log n)$ and constant ϵ.

Perhaps the most well studied related problem is the classical *k-Median* problem where one has to find a partition of the point set into k sets $C_1, \ldots, C_k$, each having a center c_i while minimizing the total sum of distances of the points to their respective center. Some of the most recent results, following a long line of research, are [13, 5, 17], which bring down the approximation ratio to $2.592 + \epsilon$. It is worth pointing out that both MSkC and BkM seem significantly more difficult than the classical k-median problem. For instance, for the case of k-median if one is given the set of k centers the clustering of the points is immediate as each point will be assigned to the nearest center point; this has been used in a simple local search algorithm that is proved to have approximation ratio $3 + \epsilon$ [2]. However, for the case of BkM, even if one is given the location of k centers it is not clear how to cluster the points optimally.

1.2 Results and Techniques

Our two primary results are $O(\log n)$-approximation algorithms for both BkM and MSkC, improving over their previous $O(\epsilon^{-1}\log^{1+\epsilon} n)$-approximations for any constant $\epsilon > 0$ [4], and a quasi-polynomial time approximation scheme (QPTAS) for BkM in metrics with constant doubling dimension (a.k.a. doubling metrics). Note that this includes Euclidean spaces of constant dimension. Before this work, there were no results known for Euclidean metrics apart from what was known about general metrics.

Similar to the approximation in [4], our improved $O(\log n)$-approximation for general metrics uses Hierarchically Separated Trees (HSTs), defined formally in Section 2. Specifically, we give a deterministic constant-factor approximation for BkM on HSTs. As is well-known, an arbitrary metric can be probabilistically embedded into an HST with the expected stretch of each edge being $O(\log n)$ [8], thus our algorithm leads immediately to a randomized, polynomial time algorithm that computes a solution with expected cost $O(\log n)$ times the optimum solution cost.

The approximation in [4] relied on slightly non-conventional HSTs where the diameters of the subtrees drop by an $O(\log^{\epsilon} n)$-factor instead of the usual $O(1)$ factor. One can obtain such HSTs with $O(\frac{1}{\epsilon}\log n/\log\log n)$ height which was necessary in order to ensure that their algorithm runs in polynomial time. Our dynamic programming approach is quite different and requires a few observations about the structure of optimal solutions in 2-HSTs. In this way, we avoid dependence on the height of the tree in the running time of our algorithm,

thereby obtaining a polynomial-time, constant-factor approximation for 2-HSTs and ultimately, a $O(\log n)$-approximation in general metrics.

Our second result, which is a QPTAS for BkM, is essentially a dynamic programming algorithm which builds on the hierarchical decomposition of a metric space with constant doubling dimensions. We start this by presenting a QPTAS for BkM for the case of a tree metric and show how this can be extended to metrics with constant doubling dimensions. This result strongly suggests that the problem is not APX-hard and therefore should have a PTAS.

For our algorithms we consider a special case of the BkM problem in which each cluster has a *type* based on rounding up the size of the cluster to the nearest power of $(1+\epsilon)$ for some given constant $\epsilon > 0$; we call this the *ϵ-Restricted Balanced k-median* (RBkM) problem. Here each cluster has one of the types $0, 1, \ldots, \lceil \log_{1+\epsilon} n \rceil$, where n denotes the number of clients, i.e., $n = |C|$. A cluster that is of type i can serve at most $(1+\epsilon)^i$ clients and the cost of serving each client j in a type i cluster with center (facility) c is $(1+\epsilon)^i \cdot d(c, j)$ (regardless of how many clients are served by the facility). We sometimes refer to $(1+\epsilon)^i$ as the capacity or the multiplier of the center (facility) of the cluster. We also say that the center of the cluster and all the clients of that cluster are of type i. It is not hard to see that an α-approximation algorithm for this version results in a $((1+\epsilon)\alpha)$-approximation algorithm for the BkM problem.

Section 2 outlines our approach for the general $O(\log n)$-approximation, including specific definitions of the HSTs we use. The dynamic programming approach for HSTs appears in Section 3. We present the QPTAS for BkM in doubling metrics in Section 4.

2 An $O(\log n)$-Approximation for General BkM

As noted earlier, our $O(\log n)$-approximation uses embeddings into tree metrics. In particular, we use the fact that an arbitrary metric can be probabilistically approximated by Hierarchically Separated Trees with $O(\log n)$ distortion. We begin by listing some properties of μ-HSTs that we use in our algorithm.

Definition 1. *For $\mu > 1$, a* μ-Hierarchical Well Separated Tree *(μ-HST) is a metric space defined on the leaves of a 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 Δ 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 have the same weight.*
- *All leaves are at the same level.*
- *For any internal node u at level i, $\Delta(u) = \Delta \cdot \mu^{-i}$.*

By this definition, any two leaf nodes u and v with a least common ancestor w are at distance exactly $\Delta(w)$ from each other. If T is a μ-HST then we let $d_T(u, v)$ denote the distance between u and v in T. It follows from [8] that for any integer $\mu > 1$, any metric can be probabilistically embedded into μ-HSTs with

stretch $O(\mu \cdot \log_\mu n)$. Furthermore, we can sample a μ-HST from this distribution in polynomial time.

In an instance of BkM on μ-HSTs T, only the leaf nodes of T correspond to clients and all the cluster centers must be leaf nodes of T. We use this in a standard way to get a randomized $O(\log n)$-approximation for BkM and MSkC.

Note: Our techniques guarantee a PTAS for μ-HSTs for any constant μ by solving the ϵ-RBkM problem exactly for appropriately small values of ϵ, but it is enough to describe a 2-approximation for BkM in 2-HSTs to get an $O(\log n)$-approximation in general metrics. Thus, we focus on this case for simplicity.

3 Dynamic Programming for BkM in 2-HSTs

Recall that in ϵ-RBkM, the capacity of each facility (or the size of each cluster) is rounded up to the nearest power of $1 + \epsilon$. For ease of exposition, we focus on the 1-RBkM problem (i.e. where all cluster sizes are powers of two) and present an exact algorithm for this problem on 2-HSTs. Clearly, this implies a 2-approximation for the BkM problem on such graphs. In this section we simply use RBkM to refer to 1-RBkM. We prove the following:

Theorem 1. *RBkM instances in 2-HSTs can be solved in polynomial time.*

To solve RBkM exactly on 2-HSTs using Dynamic Programming, we start by demonstrating the existence of an optimal solution with certain helpful properties. Let $T = (V, E)$ denote the 2-HST rooted at a vertex $r \in V$. For any vertex $v \in V$, let T_v denote the subtree of T rooted at v. It is obvious that T_v itself is a 2-HST. A client (or facility) is said to be located in the subtree T_v if its corresponding vertex in the tree belongs to T_v. In the same vein, a client (or facility) is located outside T_v if it is located in the subtree $T \backslash T_v$.

We say that a facility at location v_f serves a client at location v_c if v_c is part of the cluster with center v_f. We emphasize that only the leaf nodes of a 2-HST are clients and we can only open facilities at leaf nodes. We say that a facility at v_f is of type i if it is open with capacity 2^i. Thus, each client v being served by v_f is being served with cost $2^i \cdot d(v_f, v)$.

The following two lemmas are helpful in narrowing our search for the optimum solution.

Lemma 1. *In an optimal solution, each open facility serves its collocated client.*

Lemma 2. *For every optimal solution and for each vertex v, there is at most one type i of facility in T_v that serves clients located outside T_v. Also, any other facility in T_v has type at least i.*

We record a few more simple observations before describing our recurrence.

Observation 1 *In an optimal solution to RBkM with two vertices $u, v \in V$ such that T_u and T_v are disjoint, there cannot exist two facilities f_u and f_v and clients c_u and c_v in the subtrees rooted at u and v, respectively, such that f_u serves c_v and f_v serves c_u.*

If this were not the case, we can reduce the cost by swapping the clients and having f_u serving to c_u and f_v serving to c_v to get a cheaper solution.

Observation 2 *For any feasible solution to RBkM and a vertex v in the tree, if $u, w \in T_v$ are two clients served by two facilites $f_u, f_w \notin T_v$ then the cost of pairing u with f_u and w with f_w is the same as the cost of pairing u with f_w and w with f_u.*

This is because for every vertex $v \in T$, all clients and facilities in T_v are equidistant from v by Definition 1. For the next observation, recall that all the leaves in T are located at the same level.

Observation 3 *For a facility with multiplier m_f located at v_f and a client located at v_c, let v_{lca} denote their least common ancestor. Then the cost of serving v_c at v_f is $2 \cdot m_f \cdot d(v_f, v_{lca})$.*

This will be helpful in our algorithm because, in some sense, it only keeps track of the distance between v_f and v_{lca} for a client v_c served by v_f. For an edge e between v_f and v_{lca}, we call $2 \cdot m_f \cdot d(e)$ the *actual cost* of the edge e for the (v_c, v_f) pair, where $d(e)$ is the weight of e in the metric. Note that the sum of the actual costs of edges between v_f and v_{lca} is precisely $m_f \cdot d(v_f, v_c)$.

Definition 2. *For a subtree T_v of T and any feasible solution to RBkM, we use $cost^{in}_{T_v}$ to refer to the sum of the* actual *costs of edges within T_v accrued due to all the facility-client pairs (v_f, v_c) where $v_f \in T_v$.*

Thus, for any feasible solution to RBkM, $cost^{in}_{T_r}$ is the cost of this solution.

Definition 3. *In a partial assignment of clients to facilities, the* slack *of a facility f is the difference between its capacity and the number of clients assigned to f. The slack of a subtree T_v rooted at a v is the total slack of facilities in T_v.*

We first present our dynamic programming algorithm under the assumption that the 2-HST is a full binary tree. This cannot be assumed in general, but we present this first because it is simpler than the general case and still introduces the key ideas behind our algorithm.

The general case is more technical and requires two levels of DP; the details will appear in the full version of this paper. Some intuition regarding this case is discussed at the end of this section.

3.1 The Special Case of Full Binary Trees

To define a subproblem for the DP, let us consider an arbitrary feasible solution and focus on a subtree T_v, for $v \in T$. We start by defining a few parameters:

- k_v is the number of facilities opened in the subtree T_v.
- t_v denotes the type of the facility, if any, in T_v which serves clients located outside T_v (c.f. Lemma 2). We assign a value of -1 to t_v if no client in $T \backslash T_v$ is served by a facility in T_v.

- u_v is the number of clients in $T \backslash T_v$ that are served by facilities in T_v.
- d_v is the number of clients in T_v that are served by facilities in $T \backslash T_v$ (and)
- o is the slack of T_v.

Each table entry is of the form $\mathcal{A}[v, k_v, t_v, u_v, d_v, o]$. For a vertex $v \in V$, the value stored in this table entry is the minimum of $cost^{in}_{T_v}$ over all feasible solutions with parameters k_v, t_v, u_v, d_v, o if the cell is a non-pessimal state (defined below).

Observation 3 in the previous section provides insight on why it is sufficient to keep track of the d_v values without caring about the type or the location of the facilities outside of T_v for calculating the cost of the solution. Our algorithm for RBkM fills the table for all permissible values of parameters v, k_v, t_v, u_v, d_v and o for every vertex v in a bottom-up fashion (from leaf to root). For vertices in the same level, ties are broken arbitrarily.

Pessimal States and Base Cases An entry of the dynamic programming table is said to be *trivially suboptimal* if it is forced to contain a facility that does not cover its collocated client and is said to be *infeasible* when either the number of clients to be covered or the number of facilities to be opened within a subtree is greater than the total number of nodes in the subtree. We call an entry of the table *pessimal* when it is either infeasible or trivially suboptimal. It is easy to determine the pessimal states in the DP table at the leaf level of the tree. For other subproblems, a cell in the table is pessimal if and only if all its subproblems are pessimal states. For the ease of execution of our DP, we assign a value of ∞ to these cells in our table.

Notice that, at the leaf level of a 2-HST, all the vertices are client nodes. But some of these nodes may also have a collocated facility opened. At this stage, the only non-pessimal subproblems are the following:

(a) Facility nodes that correspond to subproblems of the kind $\mathcal{A}[v, 1, t_v, u_v, 0, o]$ satisfying the capacity constraint that $u_v + o + 1 = 2^{t_v}$, where the number 1 indicates the facility's collocated client from Lemma 1 (and)
(b) Client nodes which have subproblems of the form $\mathcal{A}[v, 0, -1, 0, 1, 0]$.

The value stored in these entries are zero.

The Recurrence If the vertex v has two children v_1 and v_2 and the values for the dynamic program are already computed for all subproblems of T_{v_1} and T_{v_2}, then the recurrence we use is given as follows:

$$\begin{aligned}\mathcal{A}[v, k_v, t_v, u_v, d_v, o] = \min_{k', k'', t_1^*, t_2^*, u_1^*, u_2^*, d_1^*, d_2^*, o_1, o_2} (&\mathcal{A}[v_1, k', t_1^*, u_1^*, d_1^*, o_1] \\ &+ \mathcal{A}[v_2, k'', t_2^*, u_2^*, d_2^*, o_2] \\ &+ 2 \sum_{i \in \{1,2\}, t_i^* \geq 0} 2^{t_i^*} \cdot u_i^* \cdot d(v, v_i)), \end{aligned} \tag{1}$$

where the subproblems in the above equation satisfy the following "consistency constraints":

Type consistency: We consider two cases for the type t_v assuming that $u_v > 0$. If $u_v = 0$, the problem boils down to the case where $t_v = -1$.

1. If $t_v = -1$, then no facility in T_v serves clients located in $T \backslash T_v$. Therefore, all the clients served by facilities in T_{v_1} are located within T_{v_1} or in T_{v_2}. Similarly, for the subtree T_{v_2}, every client served by a facility in T_{v_2} is either located in T_{v_1} or in T_{v_2}. But it is clear from Observation 1 that an optimal solution cannot simultaneously have a facility in T_{v_1} serving a client in T_{v_2} and a facility in T_{v_2} serving a client in T_{v_1}. Hence, $\min(t_{v_1}, t_{v_2}) = t_v = -1$.
2. If $t_v \geq 0$, then there exists at least one client in $T \backslash T_v$ that will be served by a facility in T_v. Without loss of generality, if one of the two subtrees, say T_{v_1} has a type $t_{v_1} = -1$, then the type of the other subtree t_{v_2} must be equal to the type of the facility leaving its parent, t_v. Otherwise, if both the values t_{v_1} and t_{v_2} are non-negative, Lemma 2 implies that $\min(t_{v_1}, t_{v_2}) = t_v$.

Slack consistency: The slack of T_v comes from the combined slack of facilities in both its subtrees, T_{v_1} and T_{v_2}. Therefore, $o = o_1 + o_2$.

Consistency in the number of facilities : k_v is the number of facilities opened in T_v. Since these facilities belong to either of the two subtrees T_{v_1} and T_{v_2}, we have that $k_v = k' + k''$.

Flow consistency: $u_1^* + u_2^* + d_v = d_1^* + d_2^* + u_v$. This constraint ensures that the subproblems we are looking at are consistent with the u_v and d_v values in hand. More specifically, note that u_1^* is the number of clients in $T \backslash T_{v_1}$ served by facilities in T_{v_1} and that these u_1^* clients can either be located in T_{v_2} or in the subtree $T \backslash T_v$. Let us denote by u_{1a}^*, the number of such clients in $T \backslash T_v$ and by u_{1b}^*, the number of clients in T_{v_2} served by facilities in T_{v_1}. Likewise, let u_{2a}^* be the number of clients in $T \backslash T_v$ and u_{2b}^*, the number of clients in T_{v_1} which are served by facilities in T_{v_2}. It is easy to see that $u_{1a}^* + u_{1b}^* = u_1^*$ and $u_{2a}^* + u_{2b}^* = u_2^*$. Also, by accounting for the clients in $T \backslash T_v$ served by facilities in T_v we see

$$u_v = u_{1a}^* + u_{2a}^* \tag{2}$$

Out of the d_1^* clients in T_{v_1} and d_2^* clients in T_{v_2} which are served by facilities located outside their respective subtrees, d_v of these clients are served by facilities in $T \backslash T_v$, while the remaining clients $d_1^* + d_2^* - d_v$ must either be served by the u_{1b}^* facilities situated in T_{v_1} and u_{2b}^* situated in T_{v_2}. Hence,

$$d_1^* + d_2^* = d_v + u_{1b}^* + u_{2b}^* \tag{3}$$

Summing up the Equations (2) and (3) and from the observation that $u_{1a}^* + u_{1b}^* = u_1^*$ and $u_{2a}^* + u_{2b}^* = u_2^*$, we get the flow constraint stated above.

The last term in Equation (1) gives the sum of actual costs of the edges between v and its children for the client-facility pairs where the facility is inside one of the two subtrees T_{v_1} or T_{v_2}. From Definition 2, this value is equal to the

difference, $cost^{in}_{T_v} - (cost^{in}_{T_{v_1}} + cost^{in}_{T_{v_2}})$.

The optimal RBkM solution is the minimum value from among the entries $\mathcal{A}[r, k, -1, 0, 0, o]$ for all values of o. Note that the number of different values each parameter can take is bounded by the number of nodes in the tree (we assume k is at most the number of leaves, or else the problem is trivial) and the number of recursive calls made to compute a single entry is also polynomially-bounded, so these values can be computed in polynomial time using dynamic programming.

Intuition Behind General HSTs

In HSTs that are not necessarily binary, we still computes the values $\mathcal{A}$ as described in the binary case. However, computing these values for subproblems rooted at a verticex v with multiple children $u_1, \ldots, u_\ell$ requires a more sophisticated approach. For this, we use an "inner" dynamic programming algorithm that, for each $0 \leq i \leq k$, tracks the movement of clients between $\{T_{u_1}, \ldots, T_{u_i}\}$ and $\{T_{u_{i+1}}, \ldots, T_{u_\ell}\}$, as well as movement in and out of T_v. Using observations like in the binary case, we only have to keep track of the number of clients from a constant number of types.

4 QPTAS for Doubling Metrics

In this section we consider the generalization of BkM where C and F are not necessarily equal and present a QPTAS for it when the input metric has constant doubling dimension. We also assume that $\epsilon > 0$ is a fixed constant (error parameter) and present an exact algorithm for ϵ-RBkM which clearly implies a $(1+\epsilon)$-approximation for BkM.

For simplicity of explanation, we will describe the QPTAS only for tree metrics and defer the details for doubling metrics to the full version of this paper. At a high level, the extension to doubling metrics uses similar ideas as our QPTAS in trees, modified appropriately to work with the hierarchical decomposition of doubling metrics described by by Talwar [16].

4.1 A QPTAS for Tree Metrics

In this section, we present an exact quasi-polynomial time algorithm for the ϵ-RBkM problem on trees. Without loss of generality, we assume the tree is rooted at an arbitrary vertex r. We repeatedly remove leaves with no client or facility until there is no such leaf in the tree. We also repeatedly remove internal vertices of degree two with no client or facility by consolidating their incident edges into one edge of the total length. Also, it is not hard to see that by introducing dummy vertices and zero length edges, we can convert this modified rooted tree into an equivalent binary tree[1] in which the clients and facilities are only located on *distinct* leaves. In other words, each leaf has either a client or a facility. The

[1] A tree in which every node other than the leaves has two children.

number of vertices and edges in this binary tree remains linear in the size of the original instance.

Let $p = \lceil \log_{1+\epsilon} n \rceil$. In a solution for the ϵ-RBkM problem, we say a client or facility has type i if it belongs to a type i cluster for some $0 \leq i \leq p$. We first observe a structural property in an optimal solution of an instance of ϵ-RBkM. We think of the clients get connected to facilities (the center of the cluster) to get some service. Having said this, we prove that there is an optimal solution in which type i clients either enter or leave a subtree but not both. In other words, in this solution, there are no two clients of the same type such that one enters the subtree to get connected to a facility and one leaves the subtree to get connected to a facility. To see this, let T_v be the subtree rooted at an arbitrary vertex v, and assume clients j_1 and j_2 have the same type, j_1 is not in T_v but enters this subtree to be served by facility i_1, and client j_2 is in T_v but leaves this subtree to be served by a facility i_2. Then, it is not hard to see that because j_1 and j_2 have the same type, if we send j_1 to i_2 and j_2 to i_1, we get another feasible clustering with no more cost. Therefore, starting from an optimal solution, one can transform it to a new optimal solution satisfying the above property. We now present a dynamic programming to compute the optimal solution for the given instance of ϵ-RBkM in quasi-polynomial time.

The Table The table in our dynamic programming algorithm captures "snapshots" of solutions in a particular subtree which includes the information of how many clients of each type either enter or leave this subtree. The subproblems have the form $(v, k', \mathbf{Q})$, where v is a vertex of the tree, $k' \leq k$, and $\mathbf{Q}$ is a vector of length $p+1$ of integers; we describe these parameters below. We want to find the minimum cost solution to cover all the clients in T_v, the subtree rooted at v, such that:

1. There are at most $0 \leq k' \leq k$ open facilities in T_v. These facilities serve clients inside or outside T_v.
2. The clients in T_v are covered by the facilities inside T_v or outside T_v.
3. $\mathbf{Q}$ is a $p+1$ dimensional vector. The ith component of this vector q_i determines the number of type i clients that enter or leave T_v. When $0 \leq q_i \leq n$, q_i is the number of type i clients that enter T_v and when $-n \leq q_i \leq 0$, $|q_i|$ is the number of type i clients that leave T_v.

In a partial solution for the subproblem, the types of clients in T_v and the at most k' facilities to be opened in T_v must be determined. Each client must be assigned to an open facility of the same type in T_v or sent to v to be serviced outside, and each client shipped from outside to v must be assigned to a facility of its type inside T_v. The cost of a partial solution accounts for the cost of sending a client in T_v to a facility inside or to v (i.e., distance to the facility of v times $(1+\epsilon)^i$ where i is the type client) plus, for the clients shipped from outside of T_v to v, the cost of sending them from v to their designated facility in T_v (i.e. the distance from v to the facility times $(1+\epsilon)^i$ where i is the type client). We keep the value of a minimum cost partial solution in table entry $A[v, k', \mathbf{Q}]$.

After filling this table, the final answer will be in the entry $A[r, k, \mathbf{0}]$ where $\mathbf{0}$ is a vector with $p+1$ zero components.

Base Case 1: There is a client on v. Then, for each $0 \leq j \leq p$, we do as follows. We form a vector $\mathbf{Q}$ with $p+1$ components such that the ith component $q_i = 0$ for all $i \neq j$ and $q_i = -1$ for $i = j$. Then, we set $A[v, 0, \mathbf{Q}] = 0$. We set all other entries of the form $A[v, ., .]$ to infinity.
Base Case 2: There is a facility on v. Then, for each type $0 \leq j \leq p$ and for each integer $(1+\epsilon)^{j-1} < t \leq (1+\epsilon)^j$, we do as follows. We form a vector $\mathbf{Q}$ with $p+1$ components such that the ith component $q_i = 0$ for all $i \neq j$ and $q_i = t$ for $i = j$. We set $A[v, 1, \mathbf{Q}] = 0$ and all other entries of the form $A[v, ., .]$ to infinity.
Recursive Case: Consider a subtree rooted at a vertex v with two children v_1 and v_2. We say the subproblem corresponding $(v, k', \mathbf{Q})$ is *consistent* with subproblems $(v_1, k'_1, \mathbf{Q}_1)$ and $(v_2, k'_2, \mathbf{Q}_2)$ if $k'_1 + k'_2 \leq k'$ and $\mathbf{Q}_1 + \mathbf{Q}_2 = \mathbf{Q}$.

To find the value of a subproblem $(v, k', \mathbf{Q})$, we initialize $A[v, k', \mathbf{Q}] = \infty$ and enumerate over all subproblems for its children v_1 and v_2. For each pair of consitent subproblems $(v_1, k'_1, \mathbf{Q}_1)$ and $(v_2, k'_2, \mathbf{Q}_2)$, we update the entry to the minimum of its current value and:

$$\sum_{i=1}^{2}(A[v_i, k'_i, \mathbf{Q}_i] + \sum_{j=0}^{p}|q_j^{(i)}| \cdot (1+\epsilon)^j \cdot d(v_i, v)),$$

where $q_j^{(i)}$ is the jth component of $\mathbf{Q}_i$.

Note that the size of the DP table is $O(n^{p+3})$ and we can compute each entry in time $n^{O(p)}$, therefore:

Theorem 2. *There is a QPTAS for the BkM problem on tree metrics.*

5 Conclusion

In this paper, we have given an $O(\log n)$-approximation for BkM and MSkC in general metrics and also a quasi-PTAS for BkM in doubling metrics. Of course, the most natural open problem is to determine if either of these problems admits a true constant-factor approximation in arbitrary metric spaces. A PTAS for BkM in doubling dimension metrics or even Euclidean metrics seems quite plausible but even obtaining a constant-factor approximation in such cases is an interesting open problem. Perhaps one direction of attack would be to consider LP relaxation for the problem. It can be shown that the most natural configuration based LP (where we would have a variable $x_{i,C}$ for every possible facility location i and a set C of clients assigned to it) is equivalent to the natural LP relaxation. One of the difficulties of using LP for BkM is that most of the standard rounding techniques that have been used successfully for facility location or the k-median problem (such as filtering, clustering, etc) do not seem to work for the BkM due to the multiplier of cluster sizes. For example, the bicriteria approximation of [4] relies on a correspondence between BkM and a variant of capacitated k-median on a semi-metric space. They then used a Lagrangian relaxation and

a primal-dual method to solve the capacitated k-median; the end result though opens $O(k)$ centers. Chuzhoy and Rabani [6] presented a better approximation for capacitated k-median where there are at most k locations of centers while up to $O(1)$ centers may be open at each location. Adapting their algorithm to work for the semi-metric space resulting from the work of [4] breaks down at a technical point. In particular, where one has to combine two solutions obtained from the primal-dual method with $k_1 < k < k_2$ number of centers. If one could overcome this technical difficulty then it could lead to a $O(1)$-approximation for MSkC and BkM on general metrics. Overall, it would be interesting to see if the standard LP relaxation has a constant integrality gap.

References

1. A. Agarwal, M. Charikar, K. Makarychev, and Y. Makarychev. $O(\sqrt{\log n})$-approximation algorithms for Min UnCut, Min-2CNF Deletion, and directed cut problems. In Proc. of STOC, 2005.
2. V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, V. Pandit. Local Search Heuristics for k-Median and Facility Location Problem. SIAM Journal on Computing, 33:544-562, 2004
3. Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic application. In Proc. of FOCS, 1996.
4. Y. Bartal, M. Charikar, and D. Raz. Approximating min-sum k-Clustering in metric spaces. In Proc. of STOC, 2001.
5. J. Byrka, T. Pensyl, B. Rybicki, A. Srinivasan, and K. Trinh. An Improved Approximation for k-median, and Positive Correlation in Budgeted Optimization. In Proc. of SODA, 2015.
6. J. Chuzhoy and Y. Rabani. Approximating k-median with non-uniform capacities In Proc. of SODA, 2005.
7. A. Czumaj and C. Sohler. Small space representations for metric min-sum k-clustering and their applications. In Proc. STACS, 2007.
8. J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In Proc. of STOC, 2003.
9. W. Fernandez de la Vega, M. Karpinski, C. Kenyon, and Yuval Rabani. Approximation schemes for clustering problems. In Proc. STOC, 2003.
10. N. Guttman-Beck and R. Hassin. Approximation algorithms for min-sum p-clustering. Discrete Applied Mathematics, 89:125–142, 1998.
11. P. Indyk. A sublinear time approximation scheme for clustering in metric spaces. In Proc. of FOCS, 1999.
12. V. Kann, S. Khanna, J. Lagergren, and A. Panconessi. On the hardness of max k-cut and its dual. In Israeli Symposium on Theoretical Computer Science, 1996.
13. S. Li and O. Svensson. Approximating k-median via pseudo-approximation. In Proc. of STOC, 2013.
14. S. Sahni and T. Gonzalez, P-Complete Approximation Problems, J. of the ACM (JACM), v.23 n.3, p.555-565, July 1976
15. L.J. Schulman. Clustering for edge-cost minimization. In Proc. of STOC, 2000.
16. K. Talwar, Bypassing the embedding: algorithms for low dimensional metrics. In Proc. of STOC, 2004.
17. C. Wu, D. Xu, D. Du, and Y. Wang. An improved approximation algorithm for k-median problem using a new factor-revealing LP http://arxiv.org/abs/1410.4161