Approximation Schemas for the Min Sum k-Clustering Problem

by

Ismail Naderi Beni

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

In this thesis, we present Approximation Schemes for the Min Sum k Clustering problem on a number of classes of graph metrics. In *Min Sum k Clustering* problem introduced by Sahni and Gonzalez [22] in 1976, given a graph $G(V, E)$ with metric edge costs and parameter k, we are asked to partition V (i.e. assume all V must be clustered) into clusters $C_1, C_2, \cdots, C_k$ such that the sum of pairwise distances of points in the same cluster is minimized. This is a generalization of k-median clustering with avoiding unbalanced cluster sizes. The best known algorithm for Min Sum k Clustering by Behsaz, Friggstad, Salavatipour and Sivakumar [6] on metric spaces is an $O(\log n)$ approximation algorithm. Even for the case of trees, the best approximation factor given by the same authors is $2 + \epsilon$. Cohen-Addad, Karthik, and Lee [11] proved that min sum k clustering is APX-Hard on Metric spaces, i.e. there isn't any PTAS unless P=NP. In this work, we will make an improvement on this problem by presenting the first Quasi Polynomial Time Approximation Schemes for several graph metrics: graphs of bounded treewidth, graphs of bounded highway dimension, and graphs of bounded doubling dimensions.

Acknowledgements

I would like to thank my supervisor, Mohammad Salavatipour, for his support and guidance over the last two years, and for his patient reviews of various stages of this thesis. Thank you again for always pushing to make it better. I would also like to thank Mohsen Rezapour, without whom this work would have not been possible, not only for his thoughtful discussions but also for his academic advice and motivations.

Besides my supervisor, I am grateful to all members of my examining committee for their time and their helpful comments on my thesis.

I am thankful to my parents for their overwhelming support, and for constantly and unconditionally placing my interests ahead of theirs. I want to thank my brother for being there for me whenever I needed help. Finally, I would like to thank the Department of Computing Science for financially supporting me during my course of study.

I dedicate this thesis to the people of Iran and all the people who are fighting for their freedom.

Contents

List of Figures

Chapter 1

Introduction

Clustering is a fundamental problem in Computer Science with many applications in machine learning, data mining, and bioinformatics, among others. Given a set of points with a notion of similarity (distance) between every pair of points, in a typical k clustering problem, the task is to partition the points into k clusters to minimize dissimilarities of the points that fall into the same group. Clustering analysis has been regarded as an effective method to extract useful features and explore potential data patterns.

In the well-known *center-based* clustering problems (*k-center*, *k-median*, *k-means*), the partition is obtained by selecting a set of k *centers* and assigning each point to its nearest center. The clusters are then *evaluated* based on the distances between the points and their centers: in the case of k-center, the objective is to minimize the maximum distance of a point to its nearest center, while in the case of k-median (k-means), respectively, the objective is to minimize the sum of distances (the sum of squared distances, respectively) between points and their centers. Compared to other clustering algorithms, center-based algorithms are very efficient for clustering large and high-dimensional datasets as the main task reduces to selecting k centers.

One can argue that center-based clustering is suitable for *well-shaped* data points where each cluster can be naturally represented by only one center. Center-based clustering can be contrasted with *pairwise* clustering. In this approach, points are partitioned using pairwise similarity between data points. For example, in the case of the *k-diameter* problem, the goal is to minimize the maximum distance between two points in a cluster (the diameter of the cluster); or in the *min-sum k-clustering* problem (which is the main focus of this work), the goal is to minimize the sum of pairwise distances between

points of a cluster.

The Min-Sum k-Clustering problem (k-**MSC**) is defined as the following.

Definition 1 (Min-Sum k-Clustering)**.** *Given a metric space over a set of n points V with metric distances $d(u, v)$ between any two $u, v \in V$. In the **k-MSC** problem the goal is to partition points V into k clusters $C_1, ..., C_k$ to minimize the sum of pairwise distances between points assigned to the same cluster: $\sum_{i=1}^{k} \sum_{\{u,v\} \subseteq C_i} d(u, v)$.*

Bartal et al. [4] showed that at the cost of a (multiplicative) error of 2, instead of solving this problem, one can solve the following (center-based clustering) problem.

Definition 2 (Balanced k-Median)**.** *Given a metric space over a set of n points V with metric distances $d(u, v)$ between any two $u, v \in V$. In the **k-BM** problem, the goal is to select k points $c_1, ..., c_k \in V$ as the centers of the clusters and partition points V into clusters $C_1, ..., C_k$ to minimize $\sum_{i=1}^{k} |C_i| \sum_{v \in C_i} d(v, c_i)$*
.

Indeed, Bartal et al. [4] (as well as the follow-up papers) consider solving k-MSC by devising algorithms for this closely related (center-based clustering) problem. Going in this direction, a 2-approximation solution is the best one can get.

How about evaluating the cost of a (pairwise-based) cluster with respect to a group of centers (instead of only one center as in k-BM)? We show that at the cost of a negligible error, one can compute the min-sum cost of a pairwise-based cluster with respect to a (sufficiently large) constant-size set of *proper* centers (we will call them *proper hubs* of the cluster). Using this, we formulate a somewhat center-based clustering problem with a structure that enables us to use the dynamic programming approach. Solving this leads to the first approximation scheme for the *min-sum k-clustering* problem on different metrics as described below.

In this thesis, we give the first Quasi-Polynomial Time Approximation Schemas (QPTAS) for the k-MSC problem on tree metrics (see Chapter 2). Then, we show that the QPTAS for trees can be extended to graphs of bounded treewidth (see Chapter 3) and also graphs of bounded doubling dimension and bounded highway dimension (see Chapter 4).

1.1 Preliminaries

In this section, we give the definitions and the notation we will use throughout the thesis. Most of the definitions are based on [26], [20], [21]. We begin by giving the basic definitions related to graph theory and metrics on graphs.

1.1.1 Graphs and Metrics

A **graph** G is a triple consisting of **vertex set** $V(G)$, an **edge set** $E(G)$, and a relation that associates with each edge two vertices called its **endpoints**. When u, v are endpoints of some edge, we say u, v are neighbours or adjacent. A **loop** is an edge whose endpoints are equal. **Multiple edges** are edges having the same pair of endpoints. A **simple graph** is a graph having no loops or multiple edges. This thesis uses the term graph instead of a simple one. A **path** is a simple graph whose vertices can be ordered so that two vertices are adjacent if and only if they are consecutive in the list. A **cycle** is a graph with an equal number of vertices and edges whose vertices can be placed around a circle so that two vertices are adjacent if and only if they appear consecutively along the circle.

A **subgraph** of graph G is a graph H that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$ and the assignment of endpoints to edges in H is the same as in G. We then write $H \subseteq G$ and say that "G **contains** H". We say that there is a uv-path in graph G if G contains a path starting from u and ending to v. A graph G is **connected** if it has a u, v-path whenever $u, v \in V(G)$.

A **complete graph** is a graph whose vertices are pairwise adjacent. A graph with no cycle is called **acyclic**. A **tree** is a connected acyclic graph. A rooted tree has one vertex r chosen as **root**. For each vertex v, let $P(v)$ be the unique vr-path. The **parent** of v is its neighbour on $P(v)$; its **children** are its other neighbours. Its **ancestors** are the vertices of $P(v) - v$. Its **descendants** are the vertices u such that $P(u)$ contains v. Let the *degree* of each vertex be the number of the adjacent vertices of it. The **leaves** are the vertices with degree equal to 1. A **planted tree** is a rooted tree with a left-to-right ordering specified for the children of each vertex. A **binary tree** is a planted tree where each vertex has at most two children, and each child of a vertex is designated as its **left child** or **right child**.

A **tree decomposition** of $G = (V, E)$ consists of a tree T (on a different node set from G), and a subset $V_t \subseteq V$ associated with each node t of T. (We

will call these subsets V_t the "bags" of the tree decomposition.) We sometimes write this as the ordered pair $(T, \{V_t : t \in T\})$. The tree T and the collection of bags $V_t : t \in T$ must satisfy the following three properties.

- (Node Coverage) Every node of G belongs to at least one bag V_t.
- (Edge Coverage) For every edge e of G, there is some bag V_t containing both ends of e.
- (Coherence) Let t_1, t_2, and t_3 be three nodes of T such that t_2 lies on the path from t_1 to t_3. Then, if a node v of G belongs to both V_{t_1} and V_{t_3} , it also belongs to V_{t_2} .

Width of a tree decomposition $(T, \{V_t\})$ is one less than the maximum size of any bag V_t:

$$\text{width}(T, \{V_t\}) = \max_t |V_t| - 1$$

We then define the **tree-width** of G to be the minimum width of any tree decomposition of G.

A **metric space** is an ordered pair (X, d) where X is a set and d is a metric on X, i.e. a function $d : X \times X \to \mathbb{R}$ satisfying the following axioms for all points $x, y, z \in X$:

1. (Identity) $d(x, x) = 0$
2. (Positivity) $x \neq y \to d(x, y) > 0$
3. (Symmetry) $d(x, y) = d(y, x)$
4. (Triangle inequality) : $d(x, z) \leq d(x, y) + d(y, z)$

A **weighted graph** is a graph with numerical labels on edges denoted by $w(e)$. We only work with **non-negative** edge weights. We can convert Metric (X, d) into a complete weighted graph G such that $V(G) = X$ and $w(uv) = d(u, v)$. The wights represent distances; therefore we define the length of a path to be the sum of its edge weights. Let $d_G(u, v)$ be the length of the shortest uv-path in G, if u, v are not connected, it is infinite. Let $d_{max}(G)$ be $\max_{u,v \in G} d_G(u, v)$, $d_{min}(G)$ be $\min_{u,v \in G} d_(_G(u, v)$ and aspect ratio Δ be $\frac{d_{max}(G)}{d_{min}(G)}$. In the rest of this section, we will introduce tree, doubling and highway metrics.

A metric (X, d) is a **tree metric** if there exist an edge-weighted tree $T(X', E)$ such that $X \subseteq X'$ and for all $x, y \in X$, $d(x, y) = d_T(x, y)$.

A metric space (X, d) is said to be **doubling** if there is some constant $\kappa > 0$ such that for any $x \in X$ and $r > 0$, it is possible to cover any ball $B(x, r) = \{y | d(x, y) < r\}$ with the union of at most $2^{O(\kappa)}$ balls of radius $\frac{r}{2}$. κ is referred to as the doubling dimension of the metric. This definition is from [18].

The **highway dimension** of a weighted graph $G = (V, E)$ is the smallest integer k such that for some universal constant $c \geq 4$, for every $r \in R > 0$ and every ball $B_{cr}(v)$ of radius cr, there are at most k vertices in $B_{cr}(v)$ hitting all shortest paths of length more than r that lie completely in $B_{cr}(v)$. This definition is from [14].

1.1.2 Metric Embeddings

This section is mostly based on [8]. For some of the optimization problems devising an approximation algorithm is hard. One approach to tackling this issue is to solve it in smaller cases or by modifying a general case into a special one by preserving some properties. For example, converting graph metrics to tree metrics, due to the powerful technique of dynamic programming on trees is widely used, which we will apply in Chapter 2. Usually, we lose the optimal solution in this procedure, but we try to do it with a cost as low as possible. A metric mapping $f : M \to N$ is defined on two metric spaces $(M, d_M), (N, d_M)$ such that $\forall x_1, x_2 \in M, d_M(x_1, x_1) \leq d_N(f(x_1), f(x_2))$.

The best kind of metric mapping is isometric mapping which preserves the exact values. A mapping $f : X \to Y$ of a metric space (X, d_X) to a metric space (Y, d_Y) is an **isometric embedding** if for every two points $x_1, x_2 \in X$,

$$d_Y\big(f(x_1), f(x_2)\big) = d_X(x_1, x_2)$$

Note that the point of mappings is to embed a large space into a small space, which enables us to solve the problem in a shorter time, however we lose some of the original instance information. We define this approximation on metrics as below: A mapping $f : X \to Y$ of a metric space (X, d_X) to a metric space (Y, d_Y) is an **embedding with distortion** α if there exist a constant $r > 0$ such that for every two points $x_1, x_2 \in X$,

$$r.d_X(x_1, x_2) \leq d_Y\big(f(x_1), f(x_2)\big) \leq \alpha.r.d_X(x_1, x_2)$$

Note that an isometric embedding is an embedding with distortion 1. The constant r here is just a scaling factor because we only care about the relative values.

1.1.3 Optimization problems and Approximation algorithms

In this section, we will explain what an approximation algorithm is and how they work. This section is mostly based on [12],[27].

Decision Problems vs Optimization Problems The problems we study are optimization problems where we are asked to find a feasible solution with the best value. However, decision problems are the class of problems whose answer is "Yes" or "No".

We say that an algorithm solves a problem in time $O(T(n))$ if, when it is provided a problem instance i of size $n = |i|$, the algorithm can produce the solution in $O(T(n))$ time. Therefore we say a problem is polynomial-time solvable if there exists an algorithm to solve it in time $O(n^k)$ for some constant k. The complexity class P is the set of decision problems that are polynomial-time solvable.

The complexity class **NP** is the class of decision problems in which a polynomial-time algorithm can verify a proposed solution (certificate). The notion of a verifier requires more language theory to be formulated in detail but roughly saying, a verifier is defined as a polynomial algorithm which, for a given certificate of a decision problem, outputs whether this solution is feasible or not. An **NP-Hard** problem is a problem that every NP problem can be polynomially reduced to. There is an essential question of whether $P = NP$ or not.

Approximation Algorithms Still, we don't know whether $P = NP$ or not, although most of the important problems belong to class NP. One approach to attack this challenge is to find near-optimal solutions in sub-exponential time. We refer to algorithms returning near-optimal solutions as **approximation algorithms**. We say that an algorithm has **approximation factor** $\rho(n)$, if the solution of algorithm C for an input with size n is at most $\rho(n)$ times worse than optimal solution C^*, i.e. $\max(\frac{C}{C^*}, \frac{C^*}{C}) \leq \rho(n)$. If an algorithm achieves

an approximation ratio of $\rho(n)$, we call it a $\rho(n)$**-approximation algorithm**.

An **approximation scheme** for an optimization problem is a class of algorithms (or parameterized) that takes as input not only an instance of the problem but also a value $\epsilon > 0$ such that for any fixed ϵ, the scheme is a $(1+\epsilon)$ approximation algorithm. The class of **PTAS** class consists of problems with a polynomial time approximation scheme.

The class APX is the set of NP optimization problems that allow polynomial-time approximation algorithms with approximation ratio bounded by a constant; therefore PTAS $\subseteq$ APX. A **PTAS** reduction [28] is an approximation-preserving reduction that is often used to perform reductions between solutions to optimization problems, which it preserves the property that a problem has a polynomial time approximation scheme (PTAS). A problem is said to be **APX-hard** if there is a PTAS reduction from every problem in APX to that problem, and to be APX-complete if the problem is APX-hard and also in APX. If $P \neq NP$ is assumed to be true, then it implies that $PTAS \neq APX$, one example problem in the APX-Hard class, is the bin-packing problem.

1.2 Previous Works

Min Sum k clustering was introduced by Sahni, and Gonzalez [23] in 1976. They proved k-Max Cut is NP-Hard (where the goal is to partition points into k clusters such that the summation of distances of points in different clusters is maximized), which is the dual of Min Sum k clustering problem. Also, they presented a polynomial time k-approximation algorithm for it. Kann, Sanjeev, Lagergren and Panconesi [19] show that unless $P = NP$, there cannot be a polynomial time algorithm on general graphs with approximation factor better than $1 + \frac{1}{34k}$. They also show that it is NP-Hard to approximate Min k-Partition with $k \geq 3$ (an equivalent problem of k-MSC) within $O(n^{2-\epsilon})$. Frieze and Jerrum [15] give a $(\frac{k}{k-1})$-approximation for MAX k-CUT problem.

Different Metrics One other case of this problem is considering metrics on continuous space. The important difference in continuous space is that the facility will be selected from $\mathbb{R}^d$ against a discrete finite space. Vega, Karpinski, Mathieu and Rabani [25] gave the first polynomial time approximation scheme for k-MSC on metric spaces when k is fixed. They also gave a PTAS for k-MSC on ℓ_2^2 (Note that $(\mathbb{R}^d, ||.||_2^2)$ is not a metric) and a PTAS for k-Median

on ℓ_2^2. The running time of their algorithm is $O(f(k,\epsilon)n^{3k})$. Cohen-Addad, Karthik, and Lee [11] proved that it is NP-hard to approximate continuous k-MSC to a factor of 1.415.

Parameterized Algorithms One other modification allows the algorithm to exclude a small number of points. Banerjee, Ostrovsky and Rabani [3] devised an algorithm that for any $\epsilon > 0$, for any instance with n input points and for any positive integer $n' \leq n$, how to compute in polynomial time a clustering of at least $(1-\epsilon)n'$ points of cost at most a constant factor greater than the optimal cost of clustering n' points. Their approximation guarantee grows with $\frac{1}{\epsilon}$. Czumaj and Sohler [13] introduced the first sublinear time $(4+\epsilon)$ approximation algorithm for k-MSC on metric spaces where k is bounded $k = o(\log n/\log\log n)$. For arbitrary values of k, they present a sublinear time polylogarithmic-factor-approximation algorithm for k-MSC. Hassin and Or [17] formulated Penalized k-Min Sum Clustering as following: *Given a metric space* (V,d) *and* $p : V \to \mathbb{N} \cup 0$ *a penalty function on its vertices, the penalized min sum* k*-clustering problem is the problem of finding a partition of* V *into* $k+1$ *sets,* $S_1, ..., S_{k+1}$*, minimizing* $\sum_{i=1}^{k}\big(\sum_{\{u,v\}\in S_i} d(u,v)\big) + \sum_{u\in S_{k+1}} p(u)$. They presented an approximation scheme for penalized 1-min-sum clustering and also a 2-approximation algorithm for penalized k-min-sum problem for constant k.

2-factor reduction to k-BM Guttman-Beck and Hassin [16] first showed the 2-factor relation between k-MSC and k-BM to give a 2-approximation algorithm for k-MSC with running time $n^{O(k)}$ on general graphs. They do this by enumerating all possible cluster sizes and solving problems for each case. This makes the running time exponent dependent on k.

Bartal, Charikar and Raz [5] considered k-MSC in metric spaces and by using this relation between k-MSC and k-BM they gave the first polynomial time approximation algorithm for k-MSC. They presented an $O(\frac{1}{\epsilon}\log^{1+\epsilon} n)$ approximation algorithm for k-MSC with running time $n^{\frac{1}{\epsilon}}$. Their work is based on the embedding of arbitrary metric spaces into hierarchically separated trees (HSTs). They also presented a bi-criteria approximation algorithm with a constant approximation factor such that the number of the clusters is $O(k)$.

Behsaz, Friggstad, Salavatipour and Sivakumar [6] improved Bartal et al.s work and presented $O(\log n)$ approximation algorithm for both k-MSC and

k-BM, which is the best result for general metrics. They also present a quasi-polynomial time approximation scheme (QPTAS) for k-BM in constant doubling dimension metrics. To reach this result, they give $(1+\epsilon)$ approximation for k-BM in HSTs.

Close problems A similar problem to k-BM is capacitated k-Median defined later which Bartal et al. [5] used to give a bi-criteria for k-BM. They presented a constant factor approximation algorithm such that the number of the clusters is $O(k)$. One important property of k-BM, which makes it hard, is that even given the allocation of centers and cluster sizes, it is not easy to assign clients, since a client does not necessarily go to its closest center.

There is no known constant factor approximation algorithm for capacitated k-median unless violating the number of facilities or capacity. Adamczyk, Byrka, Marcinkowski, Meesum and Włodarczyk [1] gave $(7+\epsilon)$ approximation algorithm running in time $2^{O(k \log k)} n^{O(1)}$.

Still, there isn't any $O(1)-$approximation algorithm for k-MSC and capacitated k-median on general graphs with arbitrary k.

Chapter 2

k-MSC on Trees

This chapter gives a Quasi-Polynomial Time Approximation Scheme (QPTAS) for the k-MSC problem on trees with logarithmic heights. We will later extend this to graphs of bounded treewidth and show that one can assume the height is $O(\log n)$. This implies a QPTAS for k-MSC on trees as they have bounded treewidth.

2.1 Preliminaries

Suppose we have a tree metric $T(V, E)$ with edge weights $w(uv)$, for all $uv \in E$, and a set of points $P \subseteq V$. Let $d_T(u, v)$ denote the (shortest-path) distance in T between u and v. We let T be rooted at an arbitrary vertex $r \in V$. The parent of a vertex $v \in V \setminus \{r\}$ is the vertex adjacent to v on the path from v to r. If u is the parent of v then v is a *child* of u. A tree vertex is called a *leaf* if it has no children and is called an *internal* vertex otherwise. The *ancestors* of vertex v are all the vertices (including v) on the path connecting v to r and the *descendants* of v are all the vertices for which v is an ancestor. The *level* of each node is the number of edges on the path from it to r. The *height* of the tree is the number of edges on the longest path between the root and a leaf. We use T_v to denote the *subtree* rooted at v, $V(T_v)$ to denote the vertex set of T_v, and $E(T_v)$ to denote the edge set of T_v.

For a given set $C \subseteq P$ of points, we use $D(C)$ to denote *the sum of pairwise distances* of the points in C : $D(C) = \sum_{\{u,v\} \subseteq C} d_T(u, v)$. Also, For a given set $C \subseteq V$ of points and a given set of vertices $H \subseteq V$, called *hubs*, we denote *the sum of pairwise hub-distances* of the points in C by $D_H(C) =$

$\sum_{\{u,v\}\subseteq C} d_H(u,v)$, where $d_H(u,v) = \min_{h_1,h_2\in H} \big(d_T(u,h_1)+d_T(h_1,h_2)+d_T(h_2,v)\big)$[1]. Let $p_H(u,v)$ be the shortest path between u and v through hubs H (namely, the path with the total weight $d_H(u,v)$).

We assume every vertex of T has at most two children; if not, as long as there exists a vertex v with more than two children, say $v_1, v_2, \ldots, v_t$, do the following: add a new vertex v' and connect v' to v with a zero-cost edge and replace edges vv_1 and vv_2 with edges $v'v_1$ and $v'v_2$ of the same weights. Note that the resulting binary tree has at most $2|V|$ vertices.

Definition 3 (tree k-MSC)**.** *Given an edge-weighted tree $T = (V, E)$, a set of n points $P \subseteq V$, a positive integer k (such that $k \leq n$). The goal is to partition points P into k clusters $C_1, ..., C_k$ to minimize the sum of pairwise distances between points assigned to the same cluster: $\sum_{i=1}^{k} D(C_i)$.*

We formulate a closely related problem, called the Min β-Hub k-Clustering problem, **(k,β)-MHC**, with a structure that enables us to use dynamic programming approach.

Definition 4 (tree Min β-Hub k-Clustering)**.** *Given a set of points $P \subseteq V$ and a weighted tree $T = (V, E)$. In **(k,β)-MHC**, we are asked to partition points P into k clusters $C_1, ..., C_k$ and associate with each cluster C_i a set of hubs $H_i \subseteq V$ of size at most β (that is $|H_i| \leq \beta$) to minimize $\sum_{i=1}^{k} D_{H_i}(C_i)$.*

Observe that $(k, 1)$-MHC is equivalent to k-BM (recall that a ρ-approximation algorithm for BkM gives a 2ρ-approximation algorithm for k-MSC), and (k, n)-MHC is equivalent to k-MSC.

In the next section, we will show that for sufficiently large (constant) values of β a ρ-approximation algorithm for **(k,β)-MHC** gives $(1+\epsilon)\rho$-approximation of k-MSC on tree metrics.

2.2 Approximate Equivalence of k-MSC and (k,β)-MHC

Let $\epsilon > 0$. In this section, we show that finding a $(1 + \epsilon)$-approximation for $(k,O(\frac{1}{\epsilon}))$-MHC will imply a $(1 + O(\epsilon))$-approximation for k-MSC on tree metrics.

[1]For the sake of our dynamic programming algorithm, we let h_1 be different from h_2, however, one can argue that based on the definition of the hub distance the minimum value occurs when $h_1 = h_2$.

The following lemma (together with Theorem 6) is a tool we will use to locate a constant-size set H of "proper" hubs for a given set C of points such that $D(C) \leq D_H(C) \leq (1 + O(\epsilon))D(C)$.

For a subset of vertices $\hat{V} \subseteq V$, we use $\delta(\hat{V}) = \{v \in \hat{V} : \ uv \in E \ \& \ u \notin \hat{V}\}$ to denote the *border vertices* of $\hat{V}$.

Lemma 5. *Let $C \subseteq V$ be a set of points (say a cluster) and let $T = (V, E)$ be a given binary tree. For any $\epsilon > 0$, there exists a partition of V into a set of groups $\mathbb{C}_\epsilon = \{g_1, \ldots, g_\sigma\}$ such that all of the following properties hold:*

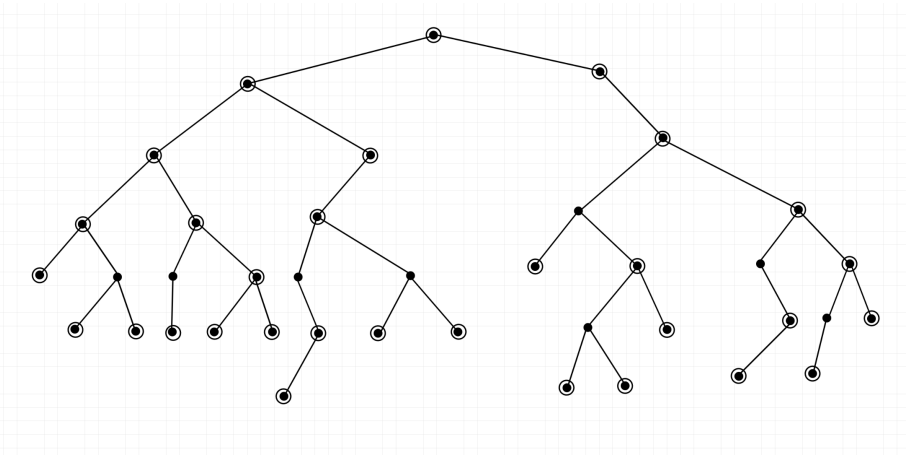
- *The subgraph induced by each group $g \in \mathbb{C}_\epsilon$ is connected.*
- *For each group $g \in \mathbb{C}_\epsilon$, $|g \cap C| \in [1, \max\{1, \epsilon|C|\}]$.*
- $|\mathbb{C}_\epsilon| = O(1/\epsilon)$.
- $\forall g \in \mathbb{C}_\epsilon, \quad |\delta(g)| = O(1/\epsilon)$.

Algorithm 1 Tree Partitioning Algorithm

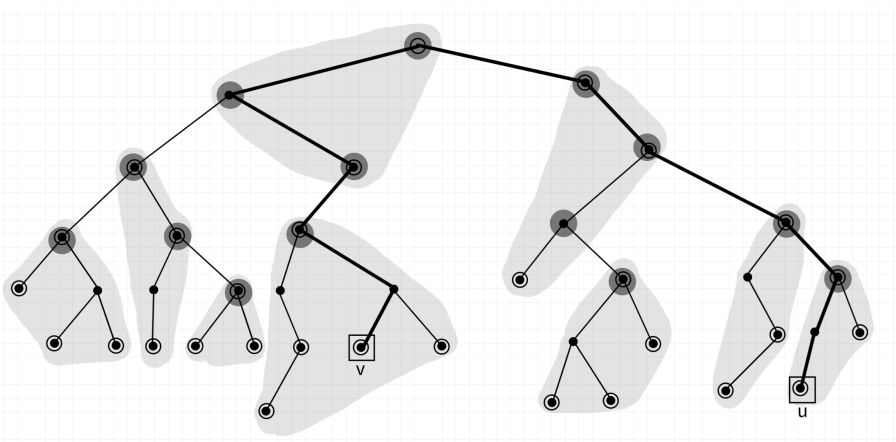
$\mathbb{C}_\epsilon \leftarrow \emptyset$
$\eta \leftarrow \max\{\epsilon|C|, 1\}$
$L \leftarrow \{v \in V : \quad \frac{1}{2}\eta \leq |V(T_v) \cap C| \leq \eta\}$
while $L \neq \emptyset$ **do**
  $\hat{v} \leftarrow v \in L$ ▷ If multiple, select v with the lowest level.
  $g \leftarrow V(T_{\hat{v}})$
  $\mathbb{C}_\epsilon \leftarrow \mathbb{C}_\epsilon \cup \{g\}$
  remove $T_{\hat{v}}$ from T
  $L \leftarrow \{v \in V(T) : \quad \frac{1}{2}\eta \leq |V(T_v) \cap C| \leq \eta\}$
end while
$\mathbb{C}_\epsilon \leftarrow \mathbb{C}_\epsilon \cup \{V(T_r)\}$

Proof. We use Algorithm 1 to compute $\mathbb{C}_\epsilon$: The algorithm iteratively selects a subtree $T_{\hat{v}}$ with approximately $\frac{\epsilon}{2}|C|$ points, adds the vertex set $V(T_{\hat{v}})$ to $\mathbb{C}_\epsilon$, and removes $T_{\hat{v}}$ from T. It is not hard to verify that at each iteration, except for the last one, such a subtree exists (recall that the tree is binary). This implies that the number of the iterations (so the number of the groups made by the algorithm) is at most $2/\epsilon$. Note that every vertex of V belongs to one group.

Note that there is at most one edge between any two groups, so $|\delta(g)| = O(1/\epsilon), \forall g \in \mathbb{C}_\epsilon$. The subgraphs induced by g_i's are connected by construction.



(a)



(b)

Figure 2.1: An illustration of the execution of Algorithm 1 on a set of points on the tree. (a) Small circles around dots specify points. (b) Shaded regions highlight the resulting groups. Note that for each pair of points that are in different groups (for example, see u and v), we have $d_T(u, v) = d_H(u, v)$ holds as long as H includes the border vertices; shaded circles depict the border vertices of the groups.

Hence, we conclude that the algorithm has constructed a partition with the desired properties. See Figure 2.1. □

Theorem 6. *Let $T = (V, E)$ be a binary tree with edge weights $w(uv)$, $\forall uv \in E$, and let $P \subseteq V$ be a given set of points. Consider a subset $C \subseteq P$ of points. For any $\epsilon > 0$, there exists a set of hubs $H \subseteq V$ of size $|H| = O(\frac{1}{\epsilon})$ for which the following holds: $D(C) \leq D_H(C) \leq (1 + O(\epsilon))D(C)$.*

Proof. Let $\mathbb{C}_\epsilon = \{g_1, \ldots, g_\sigma\}$ be the groups obtained by executing Algorithm 1 on C. For each group $g_i \in \mathbb{C}_\epsilon$, we let m_i be the median of g_i, i.e. $m_i = \arg\min_{v \in g_i} \left(\sum_{u \in g_i \cap C} d_T(u, v) \right)$. We define the set of hubs assigned to this group to be $H_i = \delta(g_i) \cup \{m_i\}$. Let $H = \cup_{i=1}^{\sigma} H_i$. By Lemma 16, we have $|H| = O(1/\epsilon)$. We shall show that inequalities $D(C) \leq D_H(C)$ and $D_H(C) - D(C) \leq O(\epsilon)D(C)$ hold with respect to this choice of H.

It is easy to verify that $D(C) \leq D_H(C)$ holds: recall that $D(C) = \sum_{\{u,v\} \subseteq C} d_T(u, v)$ and $D_H(C) = \sum_{\{u,v\} \subseteq C} d_H(u, v)$, and note that $d_T(u, v)$ is the shortest distance between u, v in T, whereas $d_H(u, v)$ is the shortest distance between u, v through hubs H (so $d_T(u, v) \leq d_H(u, v)$ for any u and v).

Let $X = \cup_{i=1}^{\sigma} X_i$, where X_i is the set of all pairs of C where one point is in g_i and one is outside, i.e. $X_i = \{(u, v) : u \in g_i \ \& \ v \in C \setminus g_i\}$. Let $I = \cup_{i=1}^{\sigma} I_i$, where I_i is the set of all pairs of points which both are in g_i, i.e. $I_i = \{(u, v) : u \in g_i \ \& \ v \in g_i\}$. Observe that $D(C) = \sum_{(u,v) \in X} d_T(u, v) + \frac{1}{2} \sum_{(u,v) \in I} d_T(u, v)$ and $D_H(C) = \sum_{(u,v) \in X} d_H(u, v) + \frac{1}{2} \sum_{(u,v) \in I} d_H(u, v)$. One can verify that $\sum_{(u,v) \in X} d_T(u, v) = \sum_{(u,v) \in X} d_H(u, v)$ holds by using the fact that $\cup_{i=1}^{\sigma} \delta(g_i) \subseteq H$ (see Figure 2.1). Therefore, to show $D_H(C) - D(C) \leq O(\epsilon)D(C)$, we just need to show that

$$\sum_{(u,v) \in I} d_H(u, v) - \sum_{(u,v) \in I} d_T(u, v) \leq O(\epsilon)D(C).$$

Claim 7. $\sum_{(u,v) \in I} d_H(u, v) \leq 2 \sum_{(u,v) \in I} d_T(u, v)$

Proof. The proof is analogous to the proof of Claim 1 of [4]. From the definitions, we have $\sum_{(u,v) \in I} d_H(u, v) = \sum_{i=1}^{\sigma} \sum_{\{u,v\} \subseteq g_i \cap C} d_H(u, v)$ and $\sum_{(u,v) \in I} d_T(u, v) = \sum_{i=1}^{\sigma} \sum_{\{u,v\} \subseteq g_i \cap C} d_T(u, v)$. To prove the claim, it suffices to show that, for each

group $g_i \in \mathbb{C}_\epsilon$, we have

$$\sum_{\{u,v\}\subseteq g_i\cap C} d_H(u,v) \le 2 \sum_{\{u,v\}\subseteq g_i\cap C} d_T(u,v) \tag{2.1}$$

Consider a group $g_i \in \mathbb{C}_\epsilon$. Since $m_i \in H$, by the definition, we have

$$\begin{aligned}\sum_{\{u,v\}\subseteq g_i\cap C} d_H(u,v) &\le \sum_{\{u,v\}\subseteq g_i\cap C} d_{\{m_i\}}(u,v) \\ &= \sum_{\{u,v\}\subseteq g_i\cap C} \big(d_T(u,m_i) + d_T(v,m_i)\big) \\ &\le |g_i\cap C| \sum_{v\in g_i\cap C} d_T(v,m_i)\end{aligned} \tag{2.2}$$

From the definition of m_i, it is easy to see that inequality $\sum_{v\in g_i\cap C} d_T(v,m_i) \le \sum_{v\in g_i\cap C} d_T(v,u)$ holds for each $u \in g_i \cap C$. Summing this over all u, we get $\sum_{u\in g_i\cap C}\sum_{v\in g_i\cap C} d_T(v,m_i) \le \sum_{u\in g_i\cap C}\sum_{v\in g_i\cap C} d_T(v,u)$. This implies

$$|g_i\cap C| \sum_{v\in g_i\cap C} d_T(v,m_i) \le \sum_{u\in g_i\cap C}\sum_{v\in g_i\cap C} d_T(v,u) \le 2 \sum_{u,v\in g_i\cap C} d_T(u,v) \tag{2.3}$$

Together (2.2) and (2.3) imply (2.1), hence the claim. □

Claim 8. $\sum_{(u,v)\in I} d_T(u,v) \le O(\epsilon)\sum_{(u,v)\in X} d_T(u,v)$

Proof. From the definitions, we have

$$\sum_{(u,v)\in I} d_T(u,v) = \sum_{i=1}^{\sigma} \sum_{u,v\in g_i\cap C} d_T(u,v)$$

and

$$\sum_{(u,v)\in X} d_T(u,v) = \sum_{i=1}^{\sigma-1}\sum_{j=i+1}^{\sigma} \sum_{u\in g_i\cap C, v\in g_j\cap C} d_T(u,v) = \sum_{i=1}^{\sigma} \frac{1}{2}\Big(\sum_{u\in g_i\cap C}\sum_{v\in C\setminus g_i} d_T(u,v)\Big).$$

Hence, to prove the claim, it suffices to show that, for each group $g_i \in \mathbb{C}_\epsilon$, we have

$$\sum_{u,v\in g_i\cap C} d_T(u,v) \le O(\epsilon)\Big(\sum_{u\in g_i\cap C}\sum_{v\in C\setminus g_i} d_T(u,v)\Big). \tag{2.4}$$

Consider a group $g_i \in \mathbb{C}_\epsilon$. For any $(u,v) \in X_i$, we let $b_i^{(u,v)}$ to be the border

vertex of $\delta(g_i)$ which lies on the u,v-path. For each $(u,v) \in X_i$, we let

$$b_i^{uv} = \arg\min_{\hat{v} \in \delta(g_i)} \big(d_T(u,\hat{v}) + d_T(\hat{v},v)\big).$$

We need a notation to show the number of the points that must cross z to connect to any point of g_i. Formally saying, for each $z \in \delta(g_i)$, let

$$c_{g_i}^z = |\{w \in C \setminus g_i : \exists u \in C \cap g_i \ s.t.\ b_i^{wu} = z\}|.$$

Notice that for any group g_i, we have $\sum_{z\in\delta(g)} c_{g_i}^z = |C \setminus g_i|$. We need another notation for the hub of $\delta(g_i)$, which has the least sum of distance to the points of g_i. Let

$$b_i = \arg\min_{v \in \delta(g_i)} \big(\sum_{u \in g_i \cap C} d_T(u,v)\big).$$

By Lemma 16, we have $|C \cap g_i| \le \epsilon |C|$ [2] and so $\sum_{z\in\delta(g)} c_{g_i}^z = |C \setminus g_i| \ge (1-\epsilon)|C|$. Using these together with the definitions of b_i^{uv} and b_i, we get

$$\begin{aligned}
\sum_{v\in C\setminus g_i} \sum_{u\in g_i\cap C} d_T(u,v) &= \sum_{v\in C\setminus g_i} \sum_{u\in g_i\cap C} \big(d_T(u,b_i^{uv}) + d_T(b_i^{uv},v)\big) \\
&\ge \sum_{v\in C\setminus g_i} \sum_{u\in g_i\cap C} d_T(u,b_i^{uv}) \\
&= \sum_{z\in\delta(g_i)} c_{g_i}^z \sum_{u\in g_i\cap C} d_T(u,z) \ge \sum_{z\in\delta(g_i)} c_{g_i}^z \sum_{u\in g_i\cap C} d_T(u,b_i) \\
&= |C\setminus g_i| \sum_{u\in g_i\cap C} d_T(u,b_i) \ge (1-\epsilon)|C| \sum_{u\in g_i\cap C} d_T(u,b_i) \\
&\ge \frac{(1-\epsilon)}{\epsilon} |C\cap g_i| \sum_{u\in g_i\cap C} d_T(u,b_i) \\
&\ge \frac{(1-\epsilon)}{\epsilon} \sum_{(u,v)\in X_i} d_T(u,v) = \frac{(1-\epsilon)}{\epsilon} D(g_i\cap C)
\end{aligned} \tag{2.5}$$

The last inequality follows from the triangle inequality. This implies $D(g_i \cap C) \le \frac{\epsilon}{(1-\epsilon)} \sum_{v\in C\setminus g_i} \sum_{u\in g_i\cap C} d_T(u,v)$, and hence (2.4). □

[2] Note that we can exclude the cases where $C \cap g_i$ is singleton as $D(C\cap g_i) = 0$ and so (2.4) holds trivially.

We have

$$\begin{aligned} D_H(C) - D(C) &= \sum_{(u,v)\in I} d_H(u,v) - \sum_{(u,v)\in I} d_T(u,v) \\ &\leq \sum_{(u,v)\in I} d_T(u,v) \\ &\leq O(\epsilon) \sum_{(u,v)\in X} d_T(u,v) \leq O(\epsilon) D(C) \end{aligned} \tag{2.6}$$

The first inequality follows from Claim 7, and the second one follows from Claim 8. □

Two immediate corollaries of Theorem 6 are the following. Let $\epsilon > 0$. Consider a cluster $C \subseteq P$. Let $\mathbb{C}_\epsilon = \{g_1, \ldots, g_\sigma\}$ be the cluster groups and let $m_i = \arg\min_{v\in g_i} \left(\sum_{u\in g_i \cap C} d_T(u,v)\right)$ be the median hub of each group (as described above). For each such cluster C and any constant $\epsilon > 0$, we let $H_\epsilon(C) = \cup_{i=1}^{\sigma} \delta(g_i)$ denote the *border hubs* of the cluster, $H_\epsilon^+(C) = \cup_{i=1}^{\sigma} \big(\delta(g_i) \cup \{m_i\}\big)$ denote the *proper hubs* of the cluster, and $\text{cost}_{H_\epsilon}(C) = \sum_{i=1}^{\sigma} \sum_{j=i+1}^{\sigma} \sum_{u\in g_i\cap C, v\in g_j\cap C} d_{H_\epsilon(C)}(u,v)$ be the *ϵ-approximate cost* of the cluster. Observe that $D(C) = \text{cost}_{H_\epsilon}(C) + \sum_{i=1}^{\sigma} \sum_{u,v\in g_i\cap C} d_T(u,v)$. By Claim 8, we have $\sum_{i=1}^{\sigma} \sum_{u,v\in g_i\cap C} d_T(u,v) \leq O(\epsilon)\text{cost}_{H_\epsilon}(C)$. Hence, we get

$$\text{cost}_{H_\epsilon}(C) \leq D(C) \leq (1 + O(\epsilon))\text{cost}_{H_\epsilon}(C). \tag{2.7}$$

Corollary 9. *Let $\epsilon > 0$. A $(1+\epsilon)$-approximation for $(k, O(1/\epsilon))$-MHC will imply a $(1+O(\epsilon))$-approximation for k-MSC on tree metrics.*

Proof. Consider an instance of the k-MSC problem with an optimal clustering $C_1, C_2, \cdots, C_k$. Let $OPT = \sum_{i=1}^{k} D(C_i)$. Let OPT' be the cost of the optimal solution to the $(k, O(1/\epsilon))$-MHC problem on the same instance. Let $\hat{C}_1, \hat{C}_2, \cdots, \hat{C}_k$ and $\hat{H}_1, \hat{H}_2, \cdots, \hat{H}_k$ be the clusters and the hubs corresponding to an $(1+\epsilon)$-approximation solution to the $(k, O(1/\epsilon))$-MHC problem on this instance. We have

$$\sum_{i=1}^{k} D_{\hat{H}_i}(\hat{C}_i) \leq (1+\epsilon)OPT' \tag{2.8}$$

We shall show that clusters $\hat{C}_1, \hat{C}_2, \cdots, \hat{C}_k$ form a k-MSC solution on the considered instance with cost $(1+O(\epsilon))OPT$.

Consider the optimal clusters $C_1, C_2, \cdots, C_k$. By Theorem 6, we have $|H_\epsilon^+(C_i)| = O(1/\epsilon)$ and $D_{H_\epsilon^+(C_i)}(C_i) \leq (1+\epsilon)D(C_i)$, for each cluster C_i. Hence,

clusters $C_1, C_2, \cdots, C_k$ and hubs $H_1 = H_\epsilon^+(C_1), H_2 = H_\epsilon^+(C_2), \cdots, H_k = H_\epsilon^+(C_k)$ form a feasible $(k, O(1/\epsilon))$-MHC solution of cost at most $(1+\epsilon)OPT$. This implies

$$OPT' \leq (1+\epsilon)OPT \tag{2.9}$$

Now, consider the $(1+\epsilon)$-approximation solution $\hat{C}_1, \hat{C}_2, \cdots, \hat{C}_k$ and $\hat{H}_1, \hat{H}_2, \cdots, \hat{H}_k$ to the $(k, O(1/\epsilon))$-MHC problem. By Theorem 6, we have $D_{H_\epsilon^+(\hat{C}_i)}(\hat{C}_i) \leq (1+\epsilon)D(\hat{C}_i)$ and from the definition we have $D(\hat{C}_i) \leq D_{\hat{H}_i}(\hat{C}_i)$, for each cluster $\hat{C}_i$. Hence, we have

$$\sum_{i=1}^{k} D_{H_\epsilon^+(\hat{C}_i)}(\hat{C}_i) \leq (1+\epsilon)\sum_{i=1}^{k} D_{\hat{H}_i}(\hat{C}_i) \tag{2.10}$$

Finally, by using Theorem 6 again, we get

$$\sum_{i=1}^{k} D(\hat{C}_i) \leq (1+\epsilon)\sum_{i=1}^{k} D_{H_\epsilon^+(\hat{C}_i)}(\hat{C}_i) \tag{2.11}$$

Together (2.8), (2.9), (2.10) and (2.11) imply

$$\sum_{i=1}^{k} D(\hat{C}_i) \leq (1+O(\epsilon))\sum_{i=1}^{k} D(C_i) = (1+O(\epsilon))OPT \tag{2.12}$$

as desired. □

We show that it is sufficient to obtain a QPTAS for a further simplification of (k, β)-MHC as defined below.

Definition 10 (tree Simplified Min-Hub k-Clustering problem (k-SMHC))**.** *Given a constant $\nu > 0$, a set of points $P \subseteq V$ and a weighted tree $T = (V, E)$. In k-SMHC, we are asked to partition points P into k clusters $C_1, ..., C_k$ to minimize $\sum_{i=1}^{k} cost_{H_\nu}(C_i)$.*

Corollary 11. *Let $\epsilon > 0$. A $(1+\epsilon)$-approximation for k-SMHC will imply a $(1+O(\epsilon))$-approximation for k-MSC on tree metrics.*

Proof. The proof is analogous to that of Corollary 18. Let $I = (T, P)$ be an instance of the k-MSC problem with an optimal clustering $C_1, C_2, \cdots, C_k$. Let $OPT = \sum_{i=1}^{k} D(C_i)$. Let OPT' be the cost of the optimal solution to the k-SMHC problem on the instance $I' = (T, P, \nu = \epsilon)$. Let $\hat{C}_1, \hat{C}_2, \cdots, \hat{C}_k$ be an

$(1+\epsilon)$-approximation solution to the k-SMHC problem on I'. So we have

$$\sum_{i=1}^{k} \text{cost}_{H_\epsilon}(\hat{C}_i) \leq (1+\epsilon)OPT' \tag{2.13}$$

We shall show that clusters $\hat{C}_1, \hat{C}_2, \cdots, \hat{C}_k$ form a k-MSC solution on I with cost $(1+O(\epsilon))OPT$.

Consider the optimal clusters $C_1, C_2, \cdots, C_k$. By Inequality 2.7, we have $\text{cost}_{H_\epsilon}(C_i) \leq D(C_i)$, for any ϵ and any cluster C_i. Hence, clusters $C_1, C_2, \cdots, C_k$ form a feasible k-SMHC solution on I' with cost at most OPT. This implies

$$OPT' \leq OPT \tag{2.14}$$

Now, consider the $(1+\epsilon)$-approximate k-SMHC solution $\hat{C}_1, \hat{C}_2, \cdots, \hat{C}_k$. By Inequality 2.7, we have $D(\hat{C}_i) \leq (1+O(\epsilon))\text{cost}_{H_\epsilon}(\hat{C}_i)$, for any ϵ and any cluster $\hat{C}_i$. Hence, we have

$$\sum_{i=1}^{k} D(\hat{C}_i) \leq (1+O(\epsilon)) \sum_{i=1}^{k} \text{cost}_{H_\epsilon}(\hat{C}_i) \tag{2.15}$$

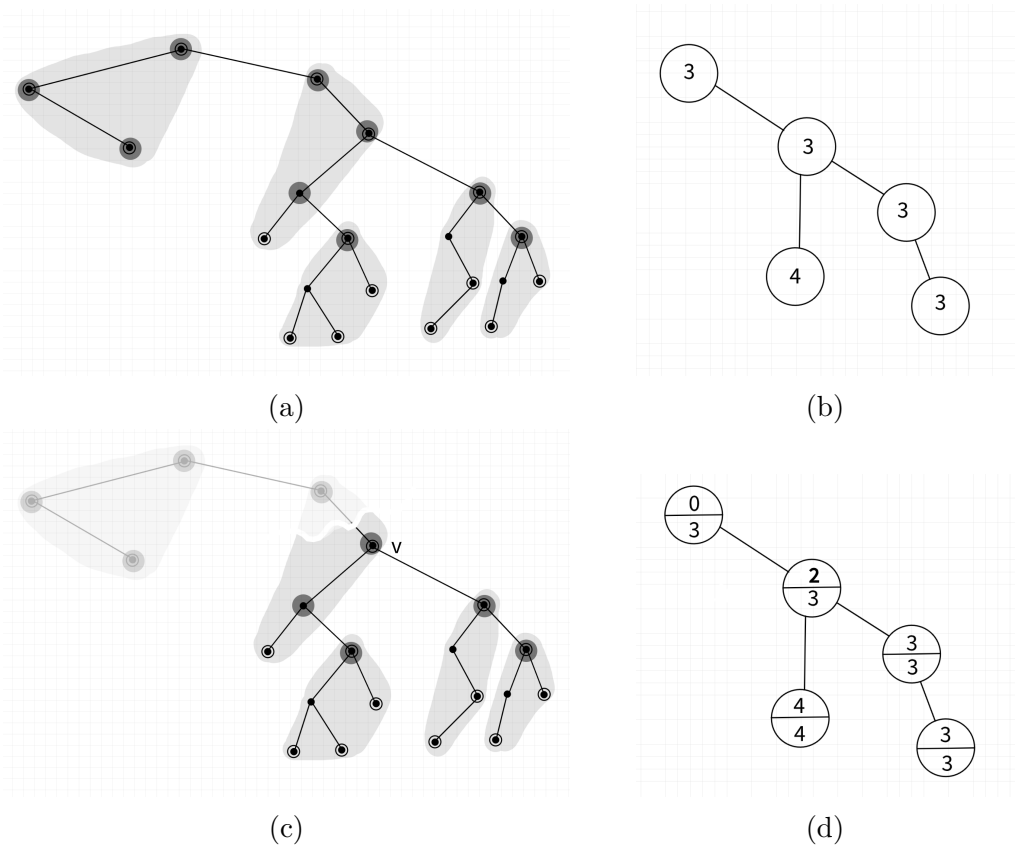
Together (2.15), (2.13), and (2.14) imply

$$\sum_{i=1}^{k} D(\hat{C}_i) \leq (1+O(\epsilon))OPT' \leq (1+O(\epsilon))OPT \tag{2.16}$$

as desired. □

2.3 An Exact (but Exponential Time) Dynamic Program

This section provides an exact exponential-time algorithm for the k-SMHC problem using the dynamic programming approach. This implies approximation schemas for the k-MSC problem (thanks to Corollary 20). We then show (in Section 2.4) how to improve the running time to quasi-polynomial. Consider a k-SMHC instance on a tree consisting of a weighted rooted tree $T = (V, E)$, a set of points $P \subseteq V$, and a constant $\nu > 0$. Our dynamic programming algorithm will find a partitioning of the points P into k clusters



(a) (b) (c) (d)

Figure 2.2: (a) A cluster and its corresponding groups. (b) The cluster type corresponds to the cluster (the backbone tree whose nodes are labelled with their weights accordingly). (c) The partial cluster with respect to T_v. (d) The corresponding partial cluster type (the backbone tree whose nodes are labelled according to their sizes/weights).

$C_1, ..., C_k$ such that $\sum_{i=1}^{k} \text{cost}_{H_\nu}(C_i)$ is minimized.

Preprocessing. We first need to do some preprocessing that helps us to simplify the explanation of the dynamic programming. We repeatedly remove leaves with no points until there is no such leaf in the tree. Also, by introducing zero-weight edges, we convert the tree into an equivalent binary tree in which the points are only located on distinct leaves. We also repeatedly remove internal vertices of degree two by consolidating their incident edges into one edge of the total weight. One can verify that the number of vertices and edges in this tree remains linear in the size of the original instance.

2.3.1 Cluster, Backbone Tree, and Partial Cluster Types

Backbone Tree. Let $\nu > 0$. Consider a cluster $C \subseteq P$. Let $\mathbb{C}_\nu = \{g_1, \ldots, g_\sigma\}$ be the groups obtained by executing Algorithm 1 on C (with $\epsilon = \nu$), where $\frac{1}{\nu} \leq \sigma \leq \frac{2}{\nu}$. We construct a tree (we call it the *backbone tree* of C) whose nodes correspond to groups $g_1, \ldots, g_\sigma$ and has edges between those nodes whose corresponding groups are connected by an edge. We abuse the notation and let g_i also denote the node corresponding to each group $g_i \in \mathbb{C}_\nu$. Note that the number of different trees which can be formed by $\tilde{n}$ labelled vertices is $\tilde{n}^{\tilde{n}-2}$ (Cayley's formula [2]). Hence, since $|\mathbb{C}_\nu|$ is $\sigma = O(1/\nu)$, the cluster's backbone tree has one of the types $1, 2, \cdots, \sigma^{\sigma-2}$.

Cluster Type. Consider a cluster $C \subseteq P$, the groups $\mathbb{C}_\nu = \{g_1, \ldots, g_\sigma\}$ as well as the backbone tree representing C (as described above). We associate with each node i of the backbone tree a *weight* that represents the number of points inside the i-th group (the group corresponding to node i) of the cluster. We use $\vec{w}$ to denote the node weights of the cluster, $\vec{w}[i] = |g_i \cap C|, \forall g_i \in \mathbb{C}_\nu$.

For each such cluster C, we associate a pair $(t_b, \vec{w})$, in which $t_b \in \{1, 2, \cdots, \sigma^{\sigma-2}\}$ specifies the type of the backbone tree of the cluster and $\vec{w}[i] \in \{0, 1, \cdots, n\}$ specifies the weight of the i-th group of the cluster (see Figure 2.2). Note that there are at most n^σ many ways (where $n = |P|$) to assign weights to nodes of a backbone tree, and hence there are $\sigma^{\sigma-2} n^\sigma$ many different pairs $(t_b, \vec{w})$ (we refer to them as the *cluster types*) to represent clusters in the optimal solution. In Section 2.4, we shall reduce the number of cluster types (and so the time complexity of the underlying DP) to a poly-logarithmic number by considering "approximate" weights of the groups for each cluster.

Partial Cluster Type. For each cluster $C \subseteq P$ and each vertex $v \in V$, we define the *partial cluster with respect to* T_v as follows. Consider the groups $\mathbb{C}_\nu = \{g_1, \ldots, g_\sigma\}$ as well as the weighted backbone tree $(t_b, \vec{w})$ representing C (as described above). Suppose $V(T_v) \cap C \neq \emptyset$. We associate with each node i of the backbone tree t_b a *size*, which specifies the number of points of the group i (the group corresponding to the node i) that intersect with $V(T_v)$. We use $\vec{s}_v$ to denote such node sizes; namely, $\vec{s}_v[i] = |(g_i \cap P) \cap V(T_v)|$ for each group i of the cluster. We let $\Gamma_v \subseteq \mathbb{C}_\nu$ indicate the groups (referred to as the *inner groups* of the partial cluster) whose vertices are entirely inside T_v. We refer to the group (of the partial cluster) that includes v as the *split group* of

the partial cluster and denote it by γ_v. Observe that $\vec{s}_v[i] = \vec{w}[i]$ holds for each group $g_i \in \Gamma_v$; and $\vec{s}_v[i] = 0$ holds for each group $g_i \in \mathbb{C}_\nu \setminus (\Gamma_v \cup \{\gamma_v\})$; see Figure 2.2. Note that any partial cluster C at root r is indeed the actual cluster C (so $\vec{s}_r[i] = \vec{w}[i]$ holds for every group i of the cluster).

For each partial cluster C with respect to T_v, we associate a triple $(t_c, \gamma_v, \vec{s}_v)$, in which $t_c \in \{1, 2, \ldots, \sigma^{\sigma-2}n^\sigma\}$ specifies the type of the cluster C, γ_v specifies the split group of the partial cluster, and $\vec{s}_v$ specifies the node (group) sizes of the partial cluster. Note that the inner groups Γ_v of the partial cluster can be induced from $t_c, \gamma_v, \vec{s}_v$. Observe that there are at most $\sigma^{\sigma-2}n^{2\sigma}$ many triples $(t_c, \gamma_v, \vec{s}_v)$ (we refer to them as the *partial cluster types*) to represent the partial clusters. Hence, each partial cluster in the optimal solution has one of the types $\ell \in \{1, 2, \cdots, \sigma^{\sigma-2}n^{2\sigma}\}$. To refer to the split group, the inner groups, the weight and the size vectors of a specific partial cluster type ℓ with respect to a specific subtree T_v we will use the notation $\gamma_v^\ell, \Gamma_v^\ell, \vec{s}_v^\ell, \vec{w}^\ell$ respectively.

Valid Partial Cluster Types. Consider a vertex v and a partial cluster type ℓ. For a given $\nu > 0$, we say the partial cluster type ℓ with respect to T_v is *valid* if the following holds:

- For each group i of ℓ, we have $0 \leq \vec{s}_v^\ell[i] \leq \vec{w}^\ell[i]$ (from the definition of the partial cluster),
- For each group i of ℓ, we have $\vec{w}^\ell[i] \leq \max\{\nu . \sum_{i'} \vec{w}^\ell[i'],\ 1\}$ (from the properties of the groups by Lemma 16),
- If v is a leaf vertex of T, then γ_v^ℓ is a leaf node of the backbone tree of ℓ. (from the definition of the backbone tree)

We say that ℓ is a **leaf partial cluster type at** v if γ_v^ℓ is a leaf node of the backbone tree of ℓ and $\vec{s}_v^\ell[\gamma_v^\ell] = 1$.

Edge Load, Cluster Cost, and Partial Cluster Cost. Let $\nu > 0$. For each cluster $C \subseteq P$ and each edge $e \in E$, we define the *load* of the edge with respect to the cluster as follows. Consider the groups $\mathbb{C}_\nu = \{g_1, \ldots, g_\sigma\}$ representing C. Let $H = H_\nu(C)$. We use $load^C(e)$ to denote the load of e with respect to C, which is the number of paths $p_H(u, v)$ that contains edge e over all $(u, v) \in X$ (recall that $X = \cup_{i=1}^{\sigma} X_i$, where $X_i = \{(u, v) : u \in g_i \ \&\ v \in C \setminus g_i\}$),

namely $load^C(e) = |\{(u,v) \in X : e \in p_H(u,v)\}|$. Observe that

$$\text{cost}_{H_\nu}(C) = \sum_{e \in E(T)} load^C(e).w(e).$$

Consider a vertex v and a partial cluster type ℓ. Let e_v denote the edge connecting v to its parent in T. One can verify that the load of edge e_v with respect to this partial cluster can be obtained as follows:

$$load^\ell(e_v) = \underbrace{\Big(\sum_{i=1, i\neq\gamma_v}^{\sigma} \vec{s}_v^{\,\ell}[i]\Big) \times \Big(\sum_{i=1}^{\sigma} (\vec{w}^\ell[i] - \vec{s}_v^{\,\ell}[i])\Big) +}_{\text{\#pairs such that one is below } \gamma_v} \underbrace{\vec{s}_v^{\,\ell}[\gamma_v] \times \Big(\sum_{i \notin \Gamma_v}^{\sigma} \vec{w}^\ell[i]\Big)}_{\text{\#pairs such that none of them is below } \gamma_v} \tag{2.17}$$

We define and compute the cost of a partial cluster type ℓ with respect to a vertex v (we denote it by $cost_v^\ell$) recursively as follows. For the base case, $cost_v^\ell = 0$, if v is a leaf vertex. For the recurrence, $cost_v^\ell = cost_{v_1}^\ell + cost_{v_2}^\ell + load^\ell(e_{v_1})w(e_{v_1}) + load^\ell(e_{v_2})w(e_{v_2})$, where v_1, v_2 are children of v.

Consider a cluster $C \subseteq P$. Note that the groups of each cluster always include r (see Algorithm 1). Let ℓ be the partial cluster type representing C at r. Observe that $cost_r^\ell = \text{cost}_{H_\nu}(C)$.

2.3.2 Dynamic Program

The Dynamic Program (DP) traverses T starting at the leaves and moving upward and considers all ways clusters can be made. We have a subproblem entry in our dynamic program table for each vertex v and each configuration (to be defined) for partial clusters for T_v. A configuration $\mathbb{P}_v \in [k]^{\sigma^{\sigma-2}n^{2\sigma}}$ specifies, for each type $\ell \in \{1, 2, \cdots, \sigma^{\sigma-2}n^{2\sigma}\}$, the number of partial clusters of type ℓ covering points inside subtree T_v.

Consider a vertex v of T. Assume for now that we have access to an inner table $\lambda[\mathbb{P}_v, \mathbb{P}_{v_1}, \mathbb{P}_{v_2}]$ that for every combination of configurations of $\mathbb{P}_v$ on v and $\mathbb{P}_{v_1}, \mathbb{P}_{v_2}$ on its children, v_1, v_2, indicates whether they are *consistent* or not, i.e. there is a solution such that for the subtrees v, v_1, v_2, the description of partial clusters below v, v_1, v_1 is $\mathbb{P}_v, \mathbb{P}_{v_1}, \mathbb{P}_{v_2}$. We will explain in the next section (Section 2.3.3) how λ is computed. We let $A[v, \mathbb{P}_v]$ store the minimum cost of a set of partial clusters, whose types are consistent with $\mathbb{P}_v$, covering all points in T_v.

We will compute subproblems $A[v, \mathbb{P}_v]$ in a bottom-up manner: we compute

$A[v, \mathbb{P}_v]$ after we have computed the subproblems $A[v_1, \mathbb{P}_{v_1}]$, $A[v_2, \mathbb{P}_{v_2}]$ for the children of v.

Base Case. For each leaf vertex v and every configuration $\mathbb{P}_v$:

$$A[v, \mathbb{P}_v] = \begin{cases} 0 & \text{if } \mathbb{P}_v[\ell] = 1 \text{ for some } \ell \text{ which is a leaf partial cluster at } v. \\ \infty & o.w. \end{cases} \tag{2.18}$$

Recurrence. For each internal vertex v and its children, v_1, v_2 and every combination of configurations of $\mathbb{P}_v$ on v and $\mathbb{P}_{v_1}, \mathbb{P}_{v_2}$:

$$A[v, \mathbb{P}_v] = \min_{\mathbb{P}_v, \mathbb{P}_{v_1}, \mathbb{P}_{v_2} : \lambda[\mathbb{P}_v, \mathbb{P}_{v_1}, \mathbb{P}_{v_2}] = True} \sum_{i=1,2} \big(A[v_i, \mathbb{P}_{v_i}] + load(v_i) w(v v_i) \big) \tag{2.19}$$

where $load(v_i) = \sum_\ell \mathbb{P}_{v_i}[\ell] load^\ell(e_{v_i})$.

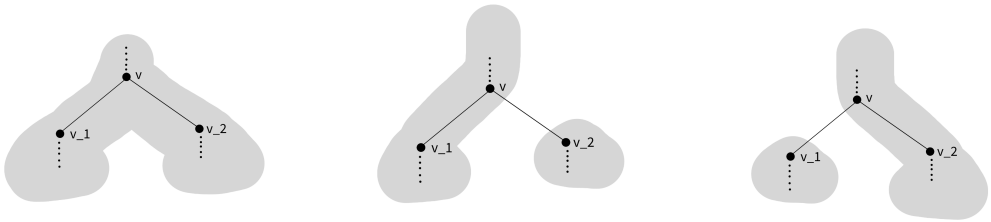
The optimal solution of k-SMHC is the minimum of $A[r, \mathbb{P}_r]$ over all $\mathbb{P}_r$ such that:

1. $\sum_\ell \mathbb{P}_r[\ell] = k$
2. For each partial cluster type ℓ with $\mathbb{P}_r[\ell] > 0$, we have $\vec{s}_r^{\ell}[i] = \vec{w}^{\ell}[i]$ holds for all i.

2.3.3 Consistency Constraints

Consider a vertex v and its two children v_1, v_2. Let $P_v = (t_c, \gamma_v, \vec{s}_v), P_{v_1} = (t_{c_1}, \gamma_{v_1}, \vec{s}_{v_1}), P_{v_2} = (t_{c_2}, \gamma_{v_2}, \vec{s}_{v_2})$ be considered valid partial cluster types at v, v_1, v_2, respectively. We say the partial cluster type P_v (with respect to T_v) is *consistent* with the two partial clusters P_{v_1} and P_{v_2} (with respect to T_{v_1} and T_{v_2}, respectively) if the following holds:

- **Type Consistency**. The cluster types of P_{v_1} and P_{v_2} are consistent with that of P_v: $t_c = t_{c_1} = t_{c_2}$.
- **Group Consistency**. The groups of P_{v_1} and P_{v_2} are consistent with those of P_v: Recall that γ_v indicates the split group of a partial cluster P_v and Γ_v indicates the inner groups of P_v. Let $\delta_v^{in} = \delta(\{\gamma_v\}) \cap \Gamma_v$ be the inner groups adjacent to γ_v where $\delta(\{\gamma_v\})$ indicates groups adjacent



(a) (b) (c)

Figure 2.3: Three possible ways that a vertex v and its children v_1 and v_2 may belong to one or two groups of a cluster. Note that each cluster covers only a subset of points, however, the groups of the cluster always include all the vertices of T (see Algorithm 1).

to γ_v (in the backbone tree). Depending on the values of $\gamma_v, \gamma_{v_1}, \gamma_{v_2}$, one of the following cases holds:

- If $\gamma_v = \gamma_{v_1} = \gamma_{v_2}$ (See Figure 3.7a), then we ensure that:
  - $\delta^{in}_{v_1} \cup \delta^{in}_{v_2} = \delta^{in}_v$
  - $\delta^{in}_{v_1} \cap \delta^{in}_{v_2} = \emptyset$
- If $\gamma_v = \gamma_{v_1}$ and $\gamma_{v_2} \in \delta^{in}_v$ (See Figure 3.7b), then we ensure that:
  - $\delta^{in}_{v_1} = \delta^{in}_v \setminus \{\gamma_{v_2}\}$
  - $\delta^{in}_{v_2} = \delta(\{\gamma_{v_2}\}) \setminus \{\gamma_v\}$
- If $\gamma_v = \gamma_{v_2}$ and $\gamma_{v_1} \in \delta^{in}_v$ (See Figure 2.3c), then we ensure that:
  - $\delta^{in}_{v_2} = \delta^{in}_v \setminus \{\gamma_{v_1}\}$
  - $\delta^{in}_{v_1} = \delta(\{\gamma_{v_1}\}) \setminus \{\gamma_v\}$

- **Size Consistency**. The group sizes of P_1 and P_2 are consistent with those of P. Note that points are located only at leaves. Depending on the values of $\gamma_v, \gamma_{v_1}, \gamma_{v_2}$, one of the following cases holds:
  - If $\gamma_v = \gamma_{v_1} = \gamma_{v_2}$ (See Figure 3.7a), then we ensure that:
    - $\vec{s}_{v_1}[\gamma_{v_1}] + \vec{s}_{v_2}[\gamma_{v_2}] = \vec{s}_v[\gamma_v]$
  - If $\gamma_v = \gamma_{v_1}$ and $\gamma_{v_2} \in \delta^{in}_v$ (See Figure 3.7b), then we ensure that:
    - $\vec{s}_{v_2}[\gamma_{v_2}] = \vec{w}[\gamma_{v_2}]$
    - $\vec{s}_{v_1}[\gamma_{v_1}] = \vec{s}_v[\gamma_v]$
  - If $\gamma_v = \gamma_{v_2}$ and $\gamma_{v_1} \in \delta^{in}_v$ (See Figure 2.3c), then we ensure that:

* $\vec{s}_{v_1}[\gamma_{v_1}] = \vec{w}[\gamma_{v_1}]$
* $\vec{s}_{v_2}[\gamma_{v_2}] = \vec{s}_v[\gamma_v]$

Note that the case that $\gamma_{v_1} = \gamma_{v_2}, \gamma_v \neq \gamma_{v_1}$ is impossible since each group of the cluster covers a connected subtree. Furthermore, the case when $\gamma_{v_1} \in \delta_v^{in}$ & $\gamma_{v_2} \in \delta_v^{in}$ is impossible using the fact that there is no point on the internal node v (see Algorithm 1).

For every combination of configurations on v and its children, v_1, v_2, $\lambda[\mathbb{P}_v, \mathbb{P}_{v_1}, \mathbb{P}_{v_2}]$ is computed recursively as below. For the base case $\lambda[\vec{0}, \vec{0}, \vec{0}]$ = TRUE. For the recurrence, we consider all possible *consistent* valid partial cluster types P_v, P_{v_1} and P_{v_2}

$$\lambda[\mathbb{P}_v, \mathbb{P}_{v_1}, \mathbb{P}_{v_2}] = \bigvee_{\forall \text{ consistent } P_v, P_{v_1}, P_{v_2}} \lambda[\mathbb{P}_v - P_v, \mathbb{P}_{v_1} - P_{v_1}, \mathbb{P}_{v_2} - P_{v_1}]$$

where $\mathbb{P}_v - P_v$ indicates the configuration of $\mathbb{P}_v$ with one less partial cluster of type P_v.

2.3.4 Analysis

It is not hard to verify that the dynamic program computes the optimal solution. Note that for each vertex v, we compute values of $n^{O(\sigma^{\sigma-2}n^{2\sigma})}$ subproblems and each such a value can be computed in time $n^{O(\sigma^{\sigma-2}n^{2\sigma})}$. Thus our algorithm runs in exponential time.

In the next section, we decrease the size of our dynamic program (the number of subproblems) by approximately storing the weights and sizes of the partial cluster type. This will lead to the main result of this chapter, namely, the first QPTAS for the k-MSC problem.

2.4 A Quasi-Polynomial Time Dynamic Program

Consider a k-SMHC instance on a tree consisting of a weighted rooted tree $T = (V, E)$, a set of points $P \subseteq V$, and a constant $\nu > 0$. Let h be the height of the tree T. Let OPT be the cost of the optimal k-SMHC solution to this instance. For any given $\epsilon > 0$, our modified dynamic programming algorithm

will find a partitioning of the points P into k clusters $C_1, ..., C_k$ such that $\sum_{i=1}^{k} \text{cost}_{H_\nu}(C_i)$ is at most $(1 + O(\epsilon h))OPT$.

ϵ-Restricted Partial Cluster Types. Let $\nu > 0$. As before, we associate with each cluster $C \subseteq P$ a pair $(t_b, \vec{w})$, in which $t_b \in \{1, 2, \cdots, \sigma^{\sigma-2}\}$ specifies the type of the backbone tree of the cluster and $\vec{w}[i]$ specifies the weight of the i-th group of the cluster, where $\sigma = O(1/\nu)$ is an upper bound for the number of the groups. However, here we store the groups' weights approximately. We define a set of threshold values (see below), and for each group i of the cluster (instead of storing the exact weight of the group), we round up the weight of the group to the nearest threshold and store this (rounded) value at $\vec{w}[i]$.

Definition 12. *For $\epsilon > 0$, let $\Phi_\epsilon = \{\phi_1, \cdots, \phi_\tau\}$ be a set of **threshold values**, where $\phi_i = i$ for $1 \leq i \leq \lceil \frac{1}{\epsilon} \rceil$, $\phi_i = \lceil \phi_{i-1}(1+\epsilon) \rceil$ for $i > \lceil \frac{1}{\epsilon} \rceil$, and $\phi_\tau = n$. So $\tau = O(\log n/\epsilon)$. We define a **mapping** ϕ which associates with each value $1 \leq i \leq n$ the minimum threshold value ϕ_j for which $i \leq \phi_j$ holds.*

This gives $O(\sigma^{\sigma-2} \log^\sigma n/\epsilon)$ many different pairs $(t_b, \vec{w})$ (we refer to them as *ϵ-restricted cluster types*) to represent the ϵ-restricted partial clusters. Remind that $\nu = \epsilon$ and $\sigma = O(1/\nu)$.

We also associate with each partial cluster $C \subseteq P$ and each vertex $v \in V$ a triple $(t_c, \gamma_v, \vec{s}_v)$ in which $t_c \in \{1, 2, \ldots, O(\sigma^{\sigma-2} \log^\sigma n/\epsilon)\}$ indicates the type of the ϵ-restricted cluster C, and $\vec{s}_v \in \Phi_\epsilon^\sigma$ denotes the node sizes rounded up to the nearest thresholds. That gives a poly-logarithmic number of *ϵ-restricted partial cluster types*.

As before, we use $\ell \in \{1, 2, \ldots, O(\sigma^{\sigma-2} \log^\sigma n/\epsilon)\}\}$ to refer to a specific ϵ-restricted partial cluster type. We retain all the notations and terminologies we defined for the partial cluster types. Note that since all the notations and terminologies are written in terms of triple $(t_c, \gamma_v, \vec{s}_v)$, they can be extended naturally to the case of the ϵ-restricted partial cluster types. We say an ϵ-restricted partial cluster type ℓ is *valid* if its corresponding triple $(t_c, \gamma_v, \vec{s}_v)$ satisfies the conditions mentioned for a valid partial cluster type. For a vertex v and a ϵ-partial cluster type ℓ, the load of edge e_v with respect to this ϵ-restricted partial cluster can be computed as $load^\ell(e_v) = \big(\sum_{i=1, i\neq\gamma_v}^{\sigma} \vec{s}_v^\ell[i]\big) \times \big(\sum_{i=1}^{\sigma} (\vec{w}^\ell[i] - \vec{s}_v^\ell[i])\big) + \vec{s}_v^\ell[\gamma_v] \times \big(\sum_{i \notin \Gamma_v}^{\sigma} \vec{w}^\ell[i]\big)$, and thereby the cost of the ϵ-restricted partial cluster $cost_v^\ell$ can be obtained recursively, as before.

Note that (unlike the case of the partial cluster type) here $(t_c, \gamma_v, \vec{s}_v)$ stores the weights and the sizes of the cluster approximately (rounded up to the nearest thresholds) and so $load^\ell(e_v)$ we just computed might overestimate the actual load of the edge by a factor of $(1+\epsilon)$. In the next section, we will see how this causes our dynamic program to obtain a $(1+O(\epsilon h))$-approximation solution.

2.4.1 Dynamic Program

The dynamic program is very similar to the one presented in the previous section. We use (almost) the same configuration for each $v \in V(T)$. A configuration $\mathbb{P}_v \in [k]^{O(\sigma^{\sigma-2}\log^\sigma n/\epsilon)\}}$ at a vertex v specifies, for each type $\ell \in \{1,2,\ldots,O(\sigma^{\sigma-2}\log^\sigma n/\epsilon)\}\}$, the number of ϵ-restricted partial cluster types of type ℓ covering points inside subtree T_v. We let $A[v,\mathbb{P}_v]$ store the minimum cost of a set of ϵ-restricted partial clusters, whose types are consistent with $\mathbb{P}_v$, covering all points in T_v. Observe that the number of such subproblems is at most $n^{O(\sigma^{\sigma-2}\log^\sigma n/\epsilon)\}}$.

As before, we assume that we have access to an inner table $\lambda[\mathbb{P},\mathbb{P}_1,\mathbb{P}_2]$ that for every combination of configurations of $\mathbb{P}$ on v and $\mathbb{P}_1,\mathbb{P}_2$ on its children, v_1, v_2, indicates whether they are *consistent* or not. We will explain in the next section how λ is computed.

We will compute subproblems $A[v,\mathbb{P}_v]$ in a bottom-up manner: we compute $A[v,\mathbb{P}_v]$ after we have computed the subproblems $A[v_1,\mathbb{P}_{v_1}]$, $A[v_2,\mathbb{P}_{v_2}]$ for the children of v. Analogous to our previous dynamic program, subproblems are computed in post order as follows.

Base Case. For each leaf vertex v:

$$A[v,\mathbb{P}_v] = \begin{cases} 0 & \text{if } \mathbb{P}_v[\ell]=1 \text{ for some } \ell \text{ which is a leaf partial cluster at } v. \\ \infty & o.w. \end{cases} \tag{2.20}$$

Recurrence. For each internal vertex v and its children, v_1, v_2:

$$A[v,\mathbb{P}_v] = \min_{\mathbb{P}_v,\mathbb{P}_{v_1},\mathbb{P}_{v_2}:\lambda[\mathbb{P}_v,\mathbb{P}_{v_1},\mathbb{P}_{v_2}]=True} \sum_{i=1,2} \big(A[v_i,\mathbb{P}_{v_i}] + load(v_i)w(vv_i)\big) \tag{2.21}$$

where $load(v_i) = \sum_\ell \mathbb{P}_{v_i}[\ell].load^\ell(e_{v_i})$.

The final solution of k-SMHC on points $P \subseteq V$ is the minimum of $A[r, \mathbb{P}_r]$ over all $\mathbb{P}_r$ such that $\sum_\ell \mathbb{P}_r[\ell] = k$ and for each ϵ-restricted partial cluster type ℓ with $\mathbb{P}_r[\ell] > 0$, we have $\vec{s}_r^{\ell}[i] = \vec{w}^{\ell}[i]$ holds for all i.

2.4.2 Consistency Constraints

The consistency-checking process is very similar to that in the previous section, though here (when we check the size consistency), we need to take the fact that the weights and sizes are stored approximately into our consideration. Let $P = (t_c, \gamma_v, \vec{s}_v), P_1 = (t_{c_1}, \gamma_{v_1}, \vec{s}_{v_1}), P_2 = (t_{c_2}, \gamma_{v_2}, \vec{s}_{v_2})$ be considered valid ϵ-restricted partial cluster types at v, v_1, v_2, respectively. We say the ϵ-restricted partial cluster P (with respect to T_v) is consistent with the two ϵ-restricted partial clusters P_1 and P_2 (with respect to T_{v_1} and T_{v_2}, respectively) if the following holds:

- **Type Consistency**. The cluster types of P_{v_1} and P_{v_2} are consistent with that of P_v: $t_c = t_{c_1} = t_{c_2}$.
- **Group Consistency**. The groups of P_{v_1} and P_{v_2} are consistent with those of P_v. Depending on the values of $\gamma_v, \gamma_{v_1}, \gamma_{v_2}$, one of the following cases holds:
  - If $\gamma_v = \gamma_{v_1} = \gamma_{v_2}$, then we ensure that:
    - $\delta^{in}_{v_1} \cup \delta^{in}_{v_2} = \delta^{in}_v$
    - $\delta^{in}_{v_1} \cap \delta^{in}_{v_2} = \emptyset$
  - If $\gamma_v = \gamma_{v_1}$ `and` $\gamma_{v_2} \in \delta^{in}_v$, then we ensure that:
    - $\delta^{in}_{v_1} = \delta^{in}_v \setminus \{\gamma_{v_2}\}$
    - $\delta^{in}_{v_2} = \delta(\{\gamma_{v_2}\}) \setminus \{\gamma_v\}$
  - If $\gamma_v = \gamma_{v_2}$ `and` $\gamma_{v_1} \in \delta^{in}_v$, then we ensure that:
    - $\delta^{in}_{v_2} = \delta^{in}_v \setminus \{\gamma_{v_1}\}$
    - $\delta^{in}_{v_1} = \delta(\{\gamma_{v_1}\}) \setminus \{\gamma_v\}$
- **Size Consistency**. The sizes of P_{v_1} and P_{v_2} are consistent with those of P_v. Depending on the values of $\gamma_v, \gamma_{v_1}, \gamma_{v_2}$, one of the following cases holds:
  - If $\gamma_v = \gamma_{v_1} = \gamma_{v_2}$, then we ensure that $\phi\big(\vec{s}_{v_1}[\gamma_{v_1}] + \vec{s}_{v_2}[\gamma_{v_2}]\big) = \vec{s}_v[\gamma_v]$.

– If $\gamma_v = \gamma_{v_1}$ and $\gamma_{v_2} \in \delta_v^{in}$, then we ensure that $\vec{s}_{v_2}[\gamma_{v_2}] = w[\gamma_{v_2}]$ and $\vec{s}_{v_1}[\gamma_{v_1}] = \vec{s}_v[\gamma_v]$.

– If $\gamma_v = \gamma_{v_2}$ and $\gamma_{v_1} \in \delta_v^{in}$, then we ensure that $\vec{s}_{v_1}[\gamma_{v_1}] = w[\gamma_{v_1}]$ and $\vec{s}_{v_2}[\gamma_{v_2}] = \vec{s}_v[\gamma_v]$

For every combination of configurations on v and its children, v_1, v_2, $\lambda[\mathbb{P}_v, \mathbb{P}_{v_1}, \mathbb{P}_{v_2}]$ is computed recursively as below. For the base case $\lambda[\vec{0}, \vec{0}, \vec{0}]$ = TRUE. For the recurrence, we consider all possible *consistent* partial cluster types P_v, P_{v_1} and P_{v_2}

$$\lambda[\mathbb{P}_v, \mathbb{P}_{v_1}, \mathbb{P}_{v_2}] = \bigvee_{\forall \text{ consistent } P_v, P_{v_1}, P_{v_2}} \lambda[\mathbb{P}_v - P_v, \mathbb{P}_{v_1} - P_{v_1}, \mathbb{P}_{v_2} - P_{v_1}]$$

where $\mathbb{P}_v - P_v$ indicates the configuration of $\mathbb{P}_v$ with one less partial cluster of type P_v.

2.4.3 Analysis

Recall that in our DP, configurations store the rounded sizes (and the rounded weights) of the partial clusters' groups, i.e. to check the consistency of sizes of groups of a subproblem at v with children v_1, v_1, we allowed the size of the group at v to be a $(1+\epsilon)$ upper bound for the added size of this group at v_1, v_2. This causes a multiplicative error of at most $(1+\epsilon)$, in the calculation of edges' loads and so the cost of the partial cluster, at every vertex of the tree when we round the sizes (weights) of a merged partial cluster. Since the height of the tree is h, it is not hard to verify that our DP finds an $(1+\epsilon)^h$-approximation solution to the problem.

Note that for each vertex v, the number of the possible configurations $\mathbb{P}_v$ is at most $n^{O(\sigma^{\sigma-2}\log^\sigma n/\epsilon)\}}$. Therefore the number of the dynamic program table entries is $n^{O(\sigma^{\sigma-2}\log^\sigma n/\epsilon)\}}$. To compute each dynamic program table entry at v, we iterated over all consistent configurations at v, v_1, v_2. Similar to v, the number of the possible configurations at v_1, v_2 is $n^{O(\sigma^{\sigma-2}\log^\sigma n/\epsilon)\}}$ too. Deciding whether configurations $\mathbb{P}_v, \mathbb{P}_{v_1}, \mathbb{P}_{v_2}$ are consistent requires iterating over all consistent partial clusters P_v, P_{v_1}, P_{v_2} which is at most equal to $O(\sigma^{\sigma-2}\log^\sigma n/\epsilon)\}$. Therefore computing each table entry takes $n^{O(\sigma^{\sigma-2}\log^\sigma n/\epsilon)\}}$ time. Hence running time is $n^{O(\sigma^{\sigma-2}\log^\sigma n/\epsilon)\}}$ which is quasi-polynomial in n.

Suppose $h = \log n$. Note that for sufficiently small values of ϵ we have $(1+\epsilon)^h \approx 1 + \epsilon h$. Hence, by setting $\epsilon' = \frac{\epsilon}{\log n}$ in the threshold mapping and

without modifying ν , our DP finds a $(1+\epsilon)$ approximation solution in time $n^{\frac{\log^{\sigma} n}{\epsilon^{\prime}/\log n}} = n^{\log^{\sigma+1} n/\epsilon}$ which is still quasi-polynomial in n.

Theorem 13. *There is a QPTAS for the k-MSC problem on trees with logarithmic heights.*

In the next chapter, we will show how to extend the result to the general trees and the graphs with bounded treewidth.

Chapter 3

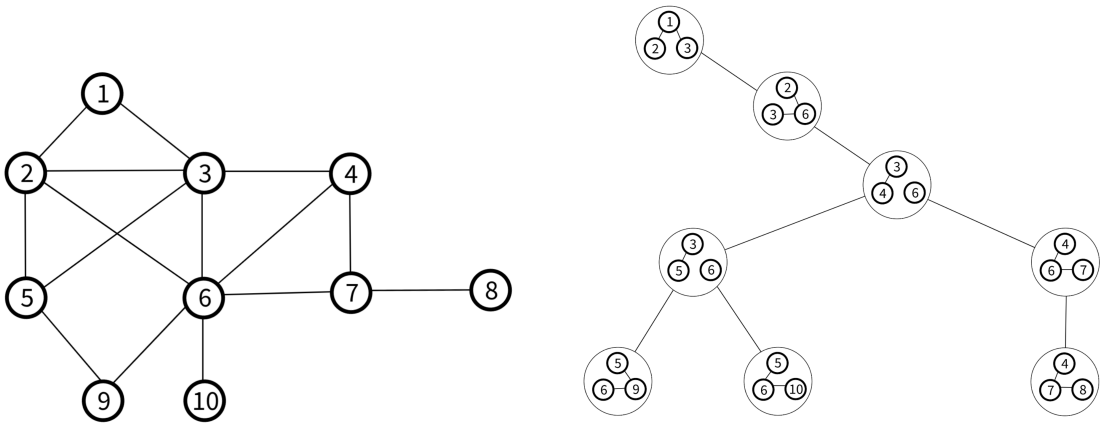
k-MSC on Graphs of Bounded Treewidth

This chapter will provide a quasi-polynomial time $(1+\epsilon)$-approximation dynamic program for the k-MSC problem on graphs with bounded treewidth. We will extend the approximation schemas presented in the previous chapter (for the case of tree metrics) to the more general case of graphs with bounded treewidth. First, we will show that (similar to the case of tree metrics) a QPTAS for a simplified variant of the (k,β)-MHC problem gives a QPTAS for the k-MSC problem on graphs with bounded treewidth and thereby we will find a $(1+\epsilon)$-approximation solution for the problem using the dynamic programming approach.

3.1 Preliminaries

Bounded Treewidth Graphs. Let $G = (V, E)$ be a graph. A *tree decomposition* of G is a tree $T = (V', E')$ on a new set of nodes V', where each $i \in V'$ corresponds to a subset b_i, called a *bag*, of vertices of V with the following properties:

- $\cup_{i \in V'} b_i = V$.
- For every edge $uv \in E$, there exists a bag t of T such that b_t contains both u and v.
- If b_i, b_j contain vertex v then every bag on the path between i and j in T contains v.



(a) (b)

Figure 3.1: (a) A graph G. (b) A tree decomposition T of the graph.

The *width* of a tree decomposition T is the size of the largest bag of T minus one; this is $\max_{i \in V'}(|b_i| - 1)$. The *treewidth* of a graph G is the minimum width over all possible tree decompositions of G. Note that the treewidth of trees is 1. See Figure 3.1.

The authors of [7] showed that any graph $G = (V, E)$ with treewidth f has a tree decomposition T of width at most $3f + 2$ that has the following two extra properties:

- T is binary.
- The height of T is $O(\log |V|)$.

Notations. Consider a graph $G = (V, E)$ with a tree decomposition $T = (V', E')$. We will refer to G as the graph and T as the tree. We will refer to vertices in V as *nodes* and vertices in V' as *bags*. We abuse the notation and use b to refer to a bag and its corresponding vertex in V'. We will refer to edges in G as *edges* and edges in T as *super-edges*. Let T be rooted at an arbitrary bag $r \in V'$. A tree bag is called a *leaf bag* if it has no children and is called an *internal bag* otherwise. The *level* of each bag is the number of super-edges on the path connecting it to r in T. We use T_b to denote the *subtree* rooted at the bag b, $V'(T_b)$ to denote the bag set of T_b, and $E'(T_b)$ to denote the super-edge set of T_b. For each bag $b \in V'$, let $V'_b = \cup_{i \in V'(T_b)} b_i$ denote the union of nodes in bags of $V'(T_b)$. Let $p : V' \to V'$ be a mapping of bags that maps each bag to its parent bag and maps r to itself.

We refer to the edges $\{(s,t) \in b \times p(b)\}$ as the *bridge-edges* with respect to the super-edge e_b. We use $e^b_{s,t}$ to refer to the bridge-edge that connects $s \in b$ to $t \in p(b)$. We add an edge between u and v in G with weight $d_G(u,v)$ (if it does not exist).

For any pair of points $u, v \in V$, one can verify that equivalent to the path $p_H(u,v)$ there exists a path on bags of T connecting u and v that consists of only the bridge-edges over the super-edges connecting B_u to B_v in T. We write $p_B(u,v)$ to refer to this path. See Figure 3.6.

The Min-Sum k-Clustering Problem on Bounded Treewidth Graphs. Given a graph $G = (V,E)$ with treewidth f', and a set $P \subseteq V$ of points, let $T = (V', E')$ be a binary decomposition tree of width at most $3f' + 2$ and height $O(\log n)$. Let f be the width of T. We assume that for each point $u \in P$, there is only one bag $b \in V'$ that contains u. Recall that each node $u \in V$ may appear in multiple bags of V', and the bags containing u form a subtree of T. To ensure that each point is covered exactly once, we mark only one of these bags as the bag containing the point: we place the point of a node at the bag containing the node which is closest to the root bag of T. Next, we further change the points so that only leaf bags of the decomposition tree contain points. To ensure this, repeatedly duplicate each internal bag that contains a point, mark the new copy as the bag containing the point, and then connect this copy to the original bag as a child. Finally, remove all leaf bags with no points. Notice that the tree remains binary, and tree height is still $O(\log n)$. See Figure 3.2. We refer to the resulting tree decomposition with all the properties mentioned above as the *proper tree decomposition* of the graph.

For each point u, we let B_u denote the bag that contains u. For each $C \subseteq P$, let $B_C = \{B_u : u \in C\}$. Let $d_G(u,v)$ denote the (shortest-path) distance in G between u and v. For a given set $C \subseteq P$ of points, we use $D(C)$ to denote *the sum of pairwise distances* of the points in C, $D(C) = \sum_{\{u,v\} \subseteq C} d_G(u,v)$. Also, For a given set $C \subseteq P$ of points and a given set of vertices $H \subseteq V$, called *hubs*, we denote *the sum of pairwise hub-distances* of the points in C by $D_H(C) = \sum_{\{u,v\} \subseteq C} d_H(u,v)$, where $d_H(u,v) = \min_{h_1,h_2 \in H} \big(d_G(u,h_1) + d_G(h_1,h_2) + d_G(h_2,v)\big)$. In other words, $d_H(u,v)$ is the shortest path between u, v that crosses two hubs, not necessarily distinct.

Definition 14 (graph k-MSC)**.** *Given an edge-weighted graph $G = (V,E)$, a*

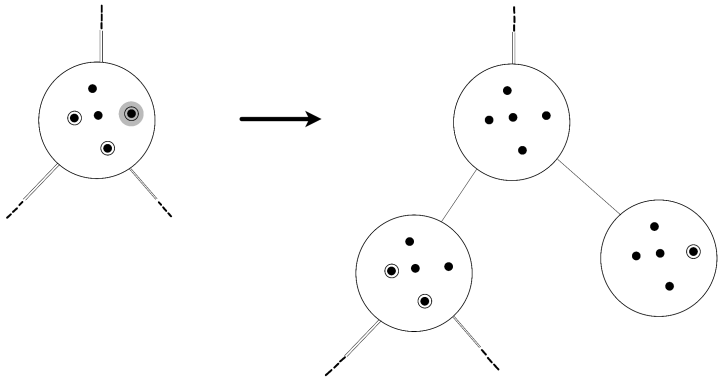


Figure 3.2: Each big circle represents a bag, the small points inside each bag are the nodes of the corresponding bag, a circle around a point shows that it is a token. By creating two duplicates of the bag on the left, the token with dark shade is placed in a leaf bag on. the right.

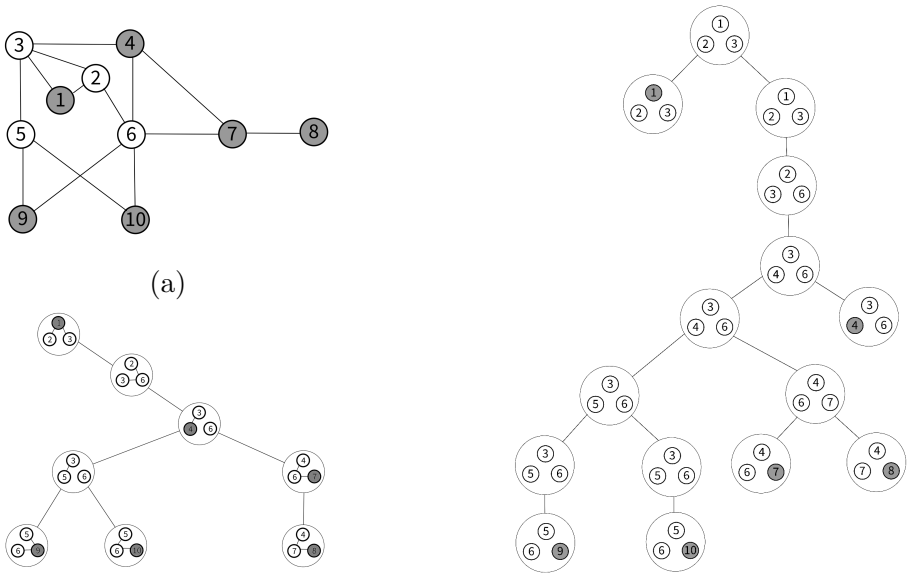


Figure 3.3: (a) A graph including a set of points (in grey). (b) A tree decomposition of the graph in which each point is placed in exactly one bag. (c) A proper tree decomposition of the graph in which points are placed only in leaf bags, each point is placed in exactly one bag.

set of n points $P \subseteq V$ and a positive integer k (such that $k \leq n$), the goal is to partition points P into k clusters $C_1, ..., C_k$ to minimize the sum of pairwise distances between points assigned to the same cluster: $\sum_{i=1}^{k} D(C_i)$.

Like trees, we formulate a closely related problem, called the Min β-Hub k-Clustering problem, **(k,β)-MHC**.

Definition 15 (graph Min β-Hub k-Clustering)**.** *Given a set of points $P \subseteq V$ and a weighted graph $G = (V, E)$, in **(k,β)-MHC**, we are asked to partition points P into k clusters $C_1, ..., C_k$ and associate with each cluster C_i a set of hubs $H_i \subseteq V$ of size at most β (that is $|H_i| \leq \beta$) to minimize $\sum_{i=1}^{k} D_{H_i}(C_i)$.*

3.2 Approximate Equivalence of k-MSC and (k,β)-MHC

Let $\epsilon > 0$. In this section, we show that finding a $(1+\epsilon)$-approximation for $(k,O(\frac{1}{\epsilon}))$-MHC will imply a $(1+O(\epsilon))$-approximation for k-MSC on graphs with bounded treewidth.

Analogous to the case of trees, employing the following lemma (together with Theorem 17), we will show how to find a constant-size set of "proper" hubs such that $D_H(C)$ would be almost equal to $D(C)$ for a given set C of points; this is: $D(C) \leq D_H(C) \leq (1+O(\epsilon))D(C)$.

For a tree decomposition $T = (V', E')$ and a subset of bags $\hat{V} \subseteq V'$, we use $\delta(\hat{V}) = \{b_i \in \hat{V} : \ b_i b_j \in E' \ \& \ b_j \notin \hat{V}\}$ to denote the *border bags* of $\hat{V}$.

Lemma 16. *Given a graph $G = (V, E)$ of bounded treewidth, a proper tree decomposition $T = (V', E')$ of G, a set of points $C \subseteq P$, for any $\epsilon > 0$, there exists a partition of V' into a set of groups $\mathbb{C}_\epsilon = \{g_1, \ldots, g_\sigma\}$ such that all of the following properties hold:*

- $\sigma = O(1/\epsilon)$.
- *The subgraph induced by each group $g \in \mathbb{C}_\epsilon$ is connected in T.*
- *For each group $g \in \mathbb{C}_\epsilon$, $|g \cap B_C| \in [1, \max\{1, \epsilon|C|\}]$.*
- $\forall g \in \mathbb{C}_\epsilon, \quad |\delta(g)| = O(1/\epsilon)$.

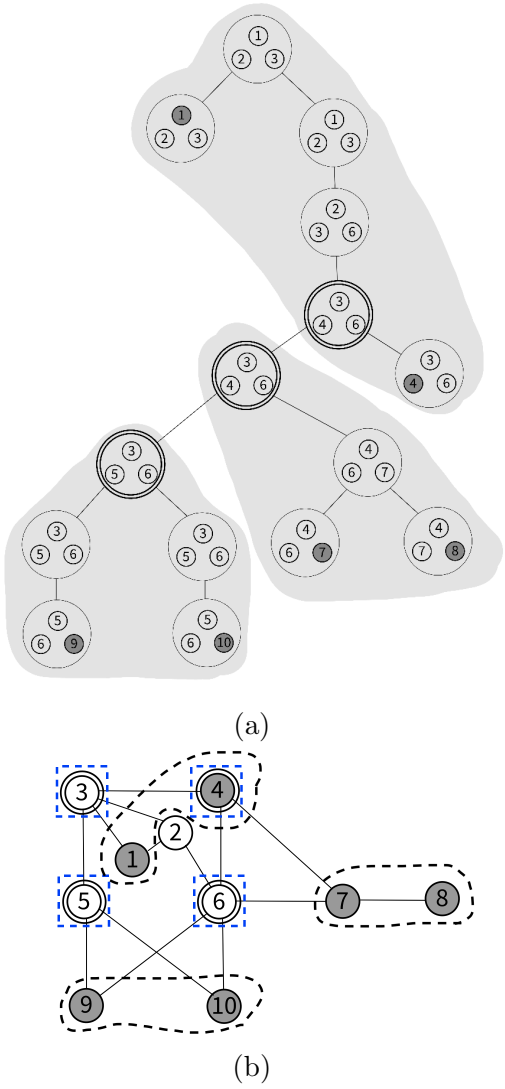


Figure 3.4: An illustration of the execution of Algorithm 1 on a proper tree decomposition of a graph. (a) Shaded regions highlight the three different groups obtained by the algorithm. Circles around bags specify the bags picked as hubs by the algorithm. (b) Circles around nodes specify the corresponding set of hubs in the graph. Observe that each path in the graph between any pair of points u and v that are in different groups (for example, all paths connecting v_1 to v_9) go through hubs, hence we have $d_G(u, v) = d_H(u, v)$ holds for such u and v as long as H includes all vertices in the border bags as picked by the algorithm.

Proof. Since there is no point in the internal bags and there is (exactly) one point in each leaf bag of the proper tree decomposition, the partitioning $\mathbb{C}_\epsilon$ of T can be computed using Algorithm 1 (by replacing C with B_C in the algorithm) and so the proof is exactly similar to the proof of Lemma 2.2. See Figure 3.4. □

Theorem 17. *Given an edge-weighted graph $G = (V, E)$, a proper three decomposition $T = (V', E')$ of width $f = O(1)$ for the graph G, a set of points $C \subseteq P$, for any $\epsilon > 0$, there exists a set of hubs $H \subseteq V$ of size $O(1/\epsilon)$ (this is, $|H| = O(\frac{f}{\epsilon})$) for which the following holds: $D(C) \leq D_H(C) \leq (1+O(\epsilon))D(C)$.*

Proof. Since the proof is very similar to that of Theorem 6, we shall sketch the proof, referring the reader to that proof for more details.

Let $\mathbb{C}_\epsilon = \{g_1, \ldots, g_\sigma\}$ be the groups of C by Lemma 16. For each group $g_i \in \mathbb{C}_\epsilon$, we let $V(g_i) = \cup_{j \in g_i} b_j$ denote the set of all vertices and points inside (the bags of) that group, and $m_i = \arg\min_{v \in V(g_i)} \left(\sum_{u \in V(g_i) \cap C} d_G(u, v) \right)$ be the median hub of the group, and $\hat{H}_i = \cup_{j \in \delta(g_i)} b_j$ be the border hubs of the group. Let $H = \cup_{i=1}^{\sigma} \left(\hat{H}_i \cup \{m_i\} \right)$ be the hub set of V (note that $H \subseteq V$). Observe that $|H| = O(\frac{f}{\epsilon})$ using Lemma 16 and the fact that the size of b_js is f. It is easy to verify that $D(C) \leq D_H(C)$ holds for each choice of H (by the definitions), so we need to show that $D_H(C) - D(C) \leq O(\epsilon)D(C)$ holds with respect to this choice of H.

Let $I = \cup_{i=1}^{\sigma} I_i$, where $I_i = \{(u, v) : u \in V(g_i) \ \& \ v \in V(g_i)\}$. Let $X = \cup_{i=1}^{\sigma} X_i$, where $X_i = \{(u, v) : u \in V(g_i) \ \& \ v \in C \setminus V(g_i)\}$. One can verify that

$$D_H(C) - D(C) = \sum_{(u,v) \in I} d_H(u, v) - \sum_{(u,v) \in I} d_G(u, v).$$

and

$$\sum_{(u,v) \in I} d_H(u, v) \leq 2 \sum_{(u,v) \in I} d_G(u, v)$$

and

$$\sum_{(u,v) \in I} d_T(u, v) \leq O(\epsilon) \sum_{(u,v) \in X} d_G(u, v).$$

hold for this choice of H (we refer the reader to the proof of Theorem 6 for details).

These three together imply $D_H(C) - D(C) \leq O(\epsilon)D(C)$ as desired.

□

Let $\epsilon > 0$. Consider a cluster $C \subseteq P$. Let $\mathbb{C}_\epsilon = \{g_1, \ldots, g_\sigma\}$ be the cluster groups. For each such cluster C and any constant $\epsilon > 0$, we let $H_\epsilon(C) = \cup_{i=1}^{\sigma} \cup_{j \in \delta(g_i)} b_j$ denote the border hubs of the cluster and $\text{cost}_{H_\epsilon}(C) = \sum_{i=1}^{\sigma} \sum_{j=i+1}^{\sigma} \sum_{u \in V(g_i) \cap C, v \in V(g_j) \cap C} d_{H_\epsilon(C)}(u, v)$ be the *ϵ-approximate cost* of the cluster. Similar to the case of trees, two immediate corollaries of Theorem 17 are the following.

Corollary 18. *Let $\epsilon > 0$. A $(1+\epsilon)$-approximation for $(k, O(1/\epsilon))$-MHC will imply a $(1 + O(\epsilon))$-approximation for k-MSC on bounded treewidth graphs.*

Definition 19 (graph Simplified Min-Hub k-Clustering problem (k-SMHC))**.** *Given a constant $\nu > 0$, a set of points $P \subseteq V$ and a weighted graph $G = (V, E)$, in k-SMHC, we are asked to partition points P into k clusters $C_1, ..., C_k$ to minimize $\sum_{i=1}^{k} cost_{H_\nu}(C_i)$.*

Corollary 20. *Given $\epsilon > 0$, for some $\nu = O(\epsilon)$, a $(1+\epsilon)$-approximation for k-SMHC, will imply a $(1 + O(\epsilon))$-approximation for k-MSC on bounded treewidth graphs.*

3.3 QPTAS Dynamic Program for k-MSC

Given $\nu > 0, G(V, E), P \subseteq V$, and a proper decomposition tree $T = (V', E')$ of graph G with treewidth f and height less than $c \log n$, for some constant c, let OPT be the value of partitioning P into k clusters $C_1, C_2, \cdots, C_k$ such that $\sum_{i=1}^{k} \text{cost}_{H_\nu}(C_i)$ is minimized. Let $\epsilon > 0$. We will present a dynamic program to find a $(1+\epsilon)$ approximation for OPT which, thanks to Theorem 17, implies a $(1 + O(\epsilon))$-approximation solution for k-MSC.

Cluster, Backbone Tree, and Partial Cluster Types. Consider a cluster $C \subseteq P$. Let $\mathbb{C}_\nu = \{g_1, \ldots, g_\sigma\}$ be the groups by Lemma 16. Each group g_i contains a subset of the bags of T. Similar to Section 2.4, we construct the backbone tree associated with C which is a tree with $O(1/\nu)$ nodes corresponding to groups of $\mathbb{C}_\nu$ and with edges between those nodes whose corresponding groups are connected by an edge in T.

Definition 21. *Given $\epsilon > 0$, let ϵ' be $\frac{\epsilon}{c \log n}$. Let **logarithmic threshold values** be $\Phi'_{\epsilon,n} = \{\phi'_1, \cdots, \phi'_{\tau'}\}$ where $\phi'_i = i$ for $1 \leq i \leq \lceil \frac{1}{\epsilon'} \rceil$, and for $i > \frac{1}{\epsilon'}$ we have $\phi'_i = \lceil \phi'_{i-1}(1+\epsilon') \rceil$, and $\phi'_{\tau'} = n$. So $\tau = O(\log n/\epsilon')$. We define*

*a **mapping** ϕ' which associates with each value $1 \leq i \leq n$ the minimum threshold value ϕ_j for which $i \leq \phi'_j$ holds.*

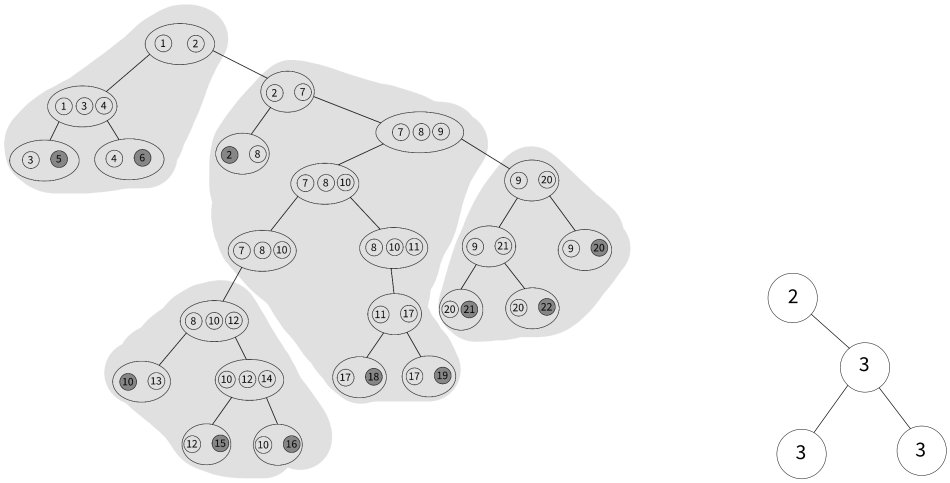
An *ϵ-restricted cluster type* (associated with a cluster) is a node-weighted backbone tree with weights vector $\vec{w} \in {\Phi'_{\epsilon,n}}^{\sigma}$ which for each group of the cluster (so for each node of the backbone tree) stores the number of points within the group's bags rounded up to the nearest threshold.

We associate to each cluster and each bag b of T, an *ϵ-restricted partial cluster type with respect to T_b* with a size vector $\vec{s}_b \in {\Phi'_{\epsilon,n}}^{\sigma}$ which for each group of the cluster stores the points at the intersection of the bags of the group and V'_b, rounded up to the nearest threshold. We use $\vec{s}_b \in {\Phi'_{\epsilon,n}}^{\sigma}$ to denote the size vector of the cluster with respect to T_b. We use a triple $(t_c, \gamma_b, \vec{s}_b)$ to denote a partial cluster with respect to b in which t_c indicates the type of the cluster, γ_b indicates the cut group of the cluster (the group of the cluster that contains the bag b), and $\vec{s}_b$ indicated the size vector of the cluster with respect to T_b. It is not hard to verify that the number of possible ϵ-restricted partial clusters is $O(\sigma^{\sigma-2} \log^{\sigma}_{(1+\epsilon')} n) = O(\log^{\sigma+1} n/\epsilon)$, where we fix $\sigma = O(1/\nu)$.

As before, we use $\ell \in \{1, 2, \ldots, O(\log^{\sigma+1} n/\epsilon)\}$ to refer to a specific ϵ-restricted partial cluster type. We say an ϵ-restricted partial cluster type ℓ, with respect to T_b, is *valid* if all of the conditions stated in Section 2.3.1 are satisfied (i.e. For each group i of ℓ, we have $0 \leq \vec{s}^{\ell}_v[i] \leq \vec{w}^{\ell}[i]$ and we have $\vec{w}^{\ell}[i] \leq \max\{\nu \sum_{i'} \vec{w}^{\ell}[i'], 1\}$ and if b is a leaf bag, then γ^{ℓ}_b is a leaf node of the backbone tree of ℓ). Parallel to the same section a **leaf partial cluster** is ℓ within T_b, such that b is a leaf and $\vec{s}^{\ell}_b[\gamma^{\ell}_b] = 1$.

Edge Loads, Cluster and Partial Cluster Costs. Let $\nu > 0$. Consider a cluster C and its groups $\mathbb{C}_{\nu} = \{g_1, \ldots, g_{\sigma}\}$. Let $H = H_{\nu}(C)$ be a set of hubs of C. For points u and v, we say path $p_H(u, v)$ goes via super-edge $e \in E'$ only if the path between B_u, B_v in T contains the super-edge e. For each cluster $C \subseteq P$ and each super-edge $e \in E'$, we define the *load of e with respect to C*, denoted by $load^C(e)$, to be the number of the paths $p_H(u, v)$ that goes via e over all $\{u, v\} \in X$ (recall that $X = \cup_{i=1}^{\sigma} X_i$, where $X_i = \{(u, v) : u \in V(g_i)\ \&\ v \in C \setminus V(g_i)\}$.

Consider a partial cluster type ℓ with respect to some bag b. Let e_b denote the super edge connecting b to its parent bag $p(b)$ in T. One can verify that the load of super-edge e_b with respect to this partial cluster can be obtained as follows (recall that γ^{ℓ}_b refers to the group that b belongs to and Γ_b is the set



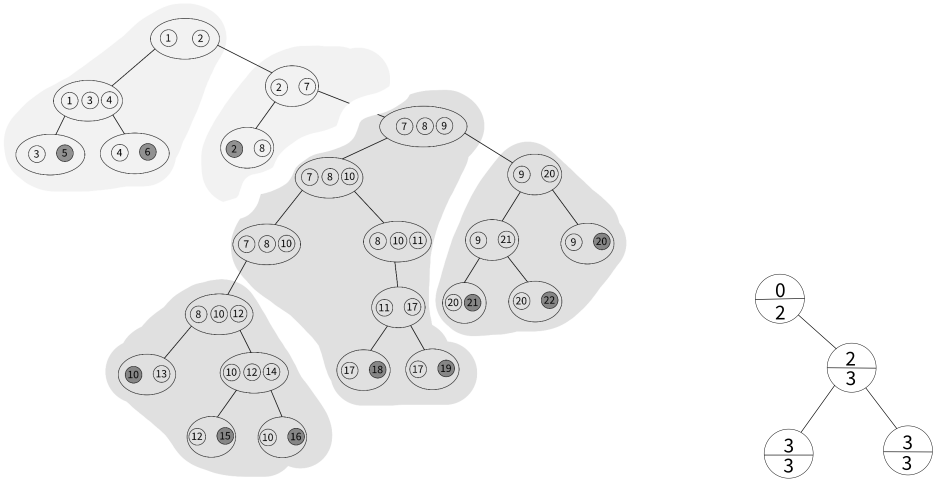
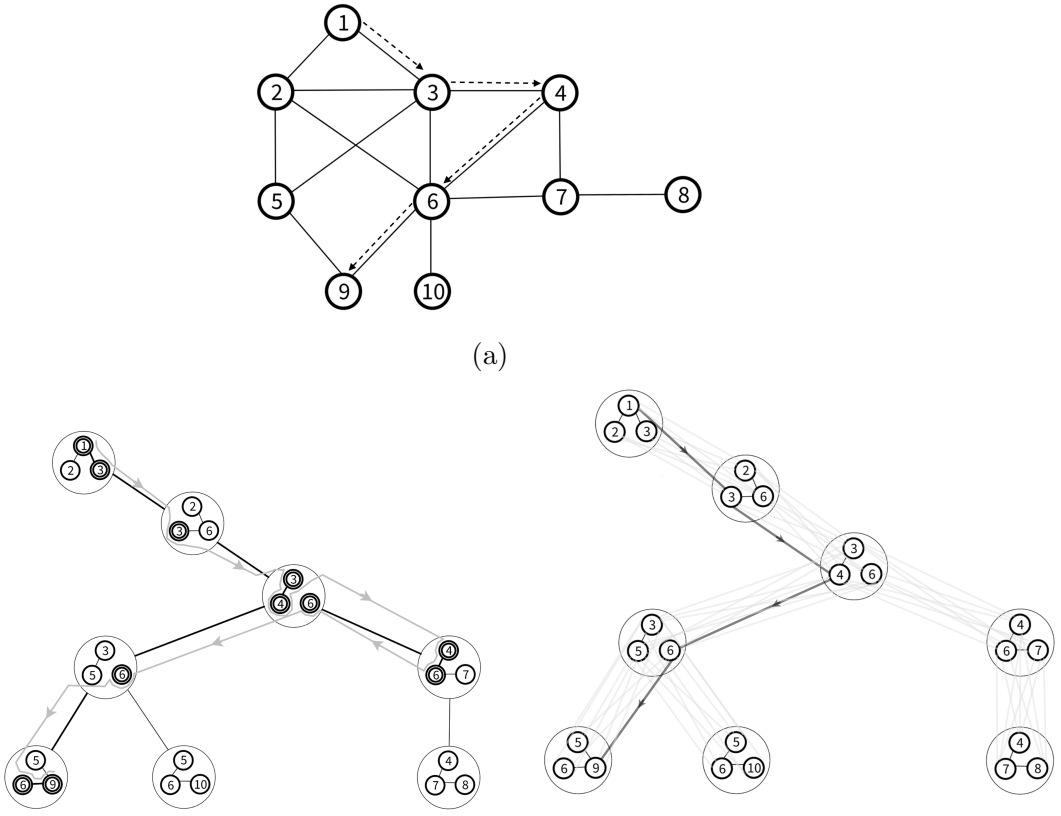


Figure 3.5: A cluster and its corresponding weighted backbone tree; the weight of each node of the backbone tree indicates the number of points within the group corresponding to that node (top figures). A partial cluster type at vertex v and its corresponding backbone tree whose nodes are associated with sizes and weights.



(a) (b) (c)

Figure 3.6: (a) shows path $< 1, 3, 4, 6, 9 >$ on a graph G. (b) shows the same path by traversing the edges inside bags and (c) shows it by using the bridge edges on the tree decomposition T for G.

of groups in T_b except γ_b):

$$load^{\ell}(e_b) = \underbrace{\Big(\sum_{i=1, i\neq\gamma_b}^{\sigma} \vec{s}_b^{\ell}[i]\Big) \times \Big(\sum_{i=1}^{\sigma} (\vec{w}^{\ell}[i] - \vec{s}_b^{\ell}[i])\Big) +}_{\#\text{pairs such that one is below } \gamma_b} \underbrace{\vec{s}_b^{\ell}[\gamma_b] \times \Big(\sum_{i\notin\Gamma_b}^{\sigma} \vec{w}^{\ell}[i]\Big)}_{\#\text{pairs such that none of them is below } \gamma_b}$$

Indeed $load^{\ell}(e_b)$ computes the number of the paths $p_H(u, v)$ that intersects the cut-set $(b, p(p))$ in G over all $(u, v) \in X$.

Consider a partial cluster C with respect to b, we define the *load of a bridge-edge* $e^b_{s,t}$ *with respect to* C, denoted by $load^C(e^b_{s,t})$, to be the number of paths $p_B(u, v)$ that contains this edge over all $\{u, v\} \in X$. Observe that

$$\text{cost}_{H_\nu}(C) = \sum_{b\in V'} \sum_{\{s,t\}\in b\times p(b)} load^C(e^b_{s,t}).w(e^b_{s,t}).$$

Also, for any bag b, we have $load^C(e_b) = \sum_{\{s,t\} \in b \times p(b)} load^C(e^b_{s,t})$. We let $load^\ell(e^b_{s,t})$ denote the the load of a bridge-edge $e^b_{s,t}$ with respect to the partial cluster type ℓ. We note that (unlike the load of the super-edges) the load of a bridge-edge can not be induced from the sizes and weights of the groups (we do not store any details on the nodes inside the bags).

Associated with each partial cluster type ℓ and each b, we define a vector ψ^ℓ_b with dimension f^2, that $\psi^\ell_b[e^b_{s,t}]$ specifies the load of each bridge-edge $e^b_{s,t}$ with respect to ℓ. One can now compute the cost of a partial cluster ℓ at bag b (we denote it by $cost^\ell_b$) recursively as follows. For the base case, $cost^\ell_b = 0$, if b is a leaf bag. For the recurrence, $cost^\ell_b = cost^\ell_{b_1} + cost^\ell_{b_2} + \sum_{\{s,t\} \in b_1 \times b} \psi^\ell_{b_1}(e^{b_1}_{s,t}) w(e^{b_1}_{s,t}) + \sum_{\{s,t\} \in b_2 \times b} \psi^\ell_{b_2}(e^{b_2}_{s,t}) w(e^{b_2}_{s,t})$, where b_1, b_2 are children of b.

3.3.1 Dynamic program

The Dynamic Program (DP) traverses T starting at the leaves and moving upward and considers all ways ϵ-restricted partial clusters can be made. A configuration $< b, \mathbb{P}_b, \psi_b >$ is defined at each bag b in which $\mathbb{P}$ specifies the number of ϵ-restricted partial cluster of each type covering points within T_b and ψ_b specifies the total load for each bridge-edge over all the partial cluster types ℓ specified in $\mathbb{P}_b$; namely, $\psi_b = \sum_\ell \mathbb{P}_b[\ell].\psi^\ell_b$.

Valid configuration In the validity check of a configuration, we ensure the feasibility of load distributions relative to partial clusters. Consider a bag b and configuration $(\mathbb{P}_b, \psi_b)$. Using the definition of the load of a cluster for super edges, we can write

$$\Psi_b = \sum_\ell \mathbb{P}_b[\ell] \psi^\ell_b load^\ell(e_b).$$

$\mathbb{P}_b$,ψ_b are consistent with respect together if and only if

$$\phi'(\Psi_b) = \phi'\Big(\sum_{e^b_{s,t} \in b \times p(b)} \psi[e^b_{s,t}] \Big).$$

Observe that for the case that b is a leaf, it implies that $\phi'(\sum_{e^b_{s,t} \in b \times p(b)} \psi[e^b_{s,t}]) = \phi'(\sum_i w[i] - 1)$.

Let $A[b, \mathbb{P}_b, \psi_b]$ be the minimum cost solution for subproblem $< b, \mathbb{P}_b, \psi_b >$

in which points in V_b' are covered by a set of partial clusters whose types (and loads) are consistent with the configuration $\mathbb{P}_b, \psi_b$ (recall that $V_b' = \cup_{i \in T_b} b_i$). Similar to before, assume that we have access to an inner table $\varphi[(\mathbb{P}, \psi), (\mathbb{P}_1, \psi_1), (\mathbb{P}_2, \psi_2)]$ that for every combination of configurations of $(\mathbb{P}, \psi)$ on b and $(\mathbb{P}_1, \psi_1), (\mathbb{P}_2, \psi_2)$ on its children, b_1, b_2, indicates whether they are *consistent* or not. In the case that b is a leaf or has one child we represent the empty configurations for its children by $\perp$.

Base Case. For each leaf vertex b:

$$A[b, \mathbb{P}_b, \psi_b] = \begin{cases} 0 & \varphi[(\mathbb{P}_b, \psi_b), \perp, \perp] = True \\ \infty & o.w. \end{cases} \tag{3.1}$$

Recurrence. For each internal vertex b and its children, b_1, b_2:

$$A[b, \mathbb{P}_b, \psi_b] = \min_{\varphi[(\mathbb{P}_b, \psi_b), (\mathbb{P}_{b_1}, \psi_{b_1}), (\mathbb{P}_{b_2}, \psi_{b_2})] = True} \left\{ \sum_{i=1,2} \left(A[b_i, \mathbb{P}_{b_i}, \psi_{b_i}] + \sum_{\{s,t\} \in b_i \times b} \psi_b[e_{s,t}^{b_i}] w(e_{s,t}^{b_i}) \right) \right\} \tag{3.2}$$

The case of b having one child is similar.

3.3.2 Consistency Constraints

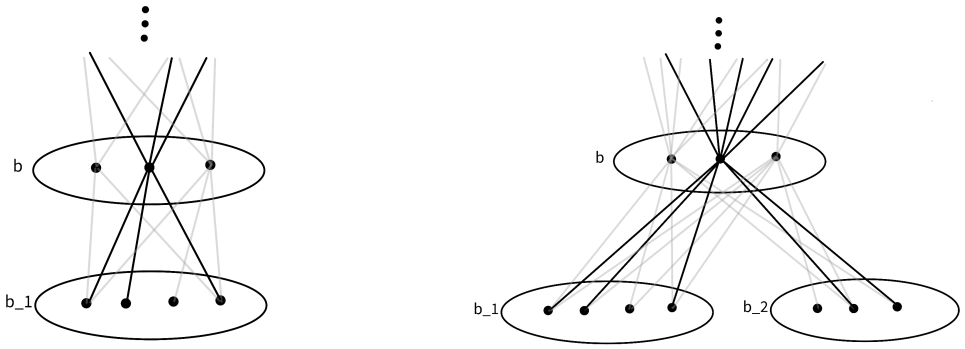
Consider a bag b and configuration $< b, \mathbb{P}_b, \psi_b >$ and for its possible children $< b_1, \mathbb{P}_{b_1}, \psi_{b_1} >, < b_2, \mathbb{P}_{b_2}, \psi_{b_2} >$. We break the consistency checking of it into two parts, the feasibility of *1) partial clusters* regarding to $(\mathbb{P}_b, \mathbb{P}_{b_1}, \mathbb{P}_{b_2})$ and feasibility of *2) load distributions* regarding to $(\psi_b, \psi_{b_1}, \psi_{b_2})$, which we will describe in the same order. We say a configuration is consistent with its children, if they are all valid configurations and all of the following constraints are satisfied ($\varphi[(\mathbb{P}_b, \psi_b), (\mathbb{P}_{b_1}, \psi_{b_1}), (\mathbb{P}_{b_2}, \psi_{b_2})] = True$). We can see $\mathbb{P}_b$ is used to check if a solution is possible,(i.e. it covers all of the points correctly), and we can view ψ_b as it is used to calculate the cost of a solution as a flow.

Feasibility of Partial Clusters There are three cases based on the degree of b.

- **(Leaf)** The following constraint must be satisfied: $\mathbb{P}_b[\ell] = 1$ for some ℓ which is a leaf partial cluster at b.

- **(One child)** Assume that b has one child. Consider $P_1 = (t_{c_1}, \gamma_{b_1}, \vec{s}_{b_1})$ be the snapshot of P at b_1. Since there is no point on internal bags, based on Algorithm 1, b and b_1 must belong to the same group. In this case, we must ensure that :
  - **(Type Consistency)** $t_b = t_{b_1}$
  - **(Group Consistency)** $\gamma_b = \gamma_{b_1}, \delta_b^{in} = \delta_{b_1}^{in}$
  - **(Size Consistency)** $\vec{s}_b = \vec{s}_{b_1}$
- **(Two children)** Assume that b has two children b_1, b_2. Let $P = (t_c, \gamma_b, \vec{s}_b), P_1 = (t_{c_1}, \gamma_{b_1}, \vec{s}_{b_1}), P_2 = (t_{c_2}, \gamma_{v_2}, \vec{s}_{b_2})$ be considered partial cluster types at b, b_1, b_2, respectively. Note that the type of a cluster is made up of backbone tree t_b and weights $\vec{w}$. Recall that similar to trees, $\delta(\{\gamma_b\})$ stands for the adjacent bags of γ_b and δ_b^{in} stands for the adjacent bags of γ_b inside T_b. We say the partial cluster type P (with respect to T_b) is consistent with the two partial clusters P_1 and P_2 (with respect to T_{b_1} and T_{b_2}, respectively) if the following holds:
  - **(Type Consistency)** $t_c = t_{c_1} = t_{c_2}$.
  - **(Group Consistency)**
    - If $\gamma_b = \gamma_{b_1} = \gamma_{b_2}$, then we ensure that $\delta_{b_1}^{in} \cup \delta_{b_2}^{in} = \delta_b^{in}$ and $\delta_{b_1}^{in} \cap \delta_{b_2}^{in} = \emptyset$.
    - If $\gamma_b = \gamma_{b_1}$ and $\gamma_{b_2} \in \delta_b^{in}$, then we ensure that $\delta_{b_1}^{in} = \delta_b^{in} \setminus \{\gamma_{b_2}\}$ and $\delta_{b_2}^{in} = \delta(\{\gamma_{b_2}\}) \setminus \{\gamma_b\}$.
    - If $\gamma_b = \gamma_{b_2}$ and $\gamma_{b_1} \in \delta_b^{in}$, then we ensure that $\delta_{b_2}^{in} = \delta_b^{in} \setminus \{\gamma_{b_1}\}$ and $\delta_{b_1}^{in} = \delta(\{\gamma_{b_1}\}) \setminus \{\gamma_b\}$.
  - **(Size Consistency)**
    - If $\gamma_b = \gamma_{b_1} = \gamma_{b_2}$, then we ensure that $\phi'\big(\vec{s}_{b_1}[\gamma_{b_1}] + \vec{s}_{b_2}[\gamma_{b_2}]\big) = \vec{s}_b[\gamma_b]$
    - If $\gamma_b = \gamma_{b_1}$ `and` $\gamma_{b_2} \in \delta_b^{in}$, then we ensure that $\vec{s}_{b_2}[\gamma_{b_2}] = w[\gamma_{b_2}]$ and $\vec{s}_{b_1}[\gamma_{b_1}] = \vec{s}_b[\gamma_b]$.
    - If $\gamma_b = \gamma_{b_2}$ `and` $\gamma_{b_1} \in \delta_b^{in}$, then we ensure that $\vec{s}_{b_1}[\gamma_{b_1}] = w[\gamma_{b_1}]$ and $\vec{s}_{b_2}[\gamma_{b_2}] = \vec{s}_b[\gamma_b]$

For every combination of configurations on b and its children, b_1, b_2, $\lambda[\mathbb{P}_b, \mathbb{P}_{b_1}, \mathbb{P}_{b_2}]$ is computed recursively as below. For the base case $\lambda[\vec{0}, \vec{0}, \vec{0}] =$ TRUE. For



(a) (b)

Figure 3.7: Consistency of load distribution per each vertex.

the recurrence, we consider all possible *consistent* ϵ-restricted partial cluster types P_b, P_{b_1} and P_{b_2}

$$\lambda[\mathbb{P}_b, \mathbb{P}_{b_1}, \mathbb{P}_{b_2}] = \bigvee_{\forall \text{ consistent } P_b, P_{v_1}, P_{v_2}} \lambda[\mathbb{P}_b - P_b, \mathbb{P}_{b_1} - P_{b_1}, \mathbb{P}_{b_2} - P_{b_1}] \tag{3.3}$$

where $\mathbb{P}_b - P_b$ indicates the configuration of $\mathbb{P}_b$ with one less partial cluster of type P_b.

Feasibility of Load Distributions There are two cases based on the number of children of b.

- **(Leaf)** Consider b is a leaf. Suppose $y \in b$ is the only point of bag b, we must ensure that: $\forall st : s \in b, t \in p(b), s \neq y, \psi[e^b_{s,t}] = 0$
- **(One child)** Consider b has one child b_1. Loads of configurations ψ_b, ψ_{b_1} are consistent if and only if, for each vertex of b, the load coming from b_1 into each vertex of b is equal to the load going upwards, formulated as following (See Figure 3.7a):

$$\forall t \in b. \quad \sum_{s \in b_1} \psi[e^{b_1}_{s,t}] = \sum_{u \in p(b)} \psi[e^b_{t,u}]$$

- **(Two children)** Consider b has two children b_1, b_2. For each $t \in b$ let L_t be $\sum_{s \in b_1} \psi[e^{b_1}_{s,t}]$, R_t be $\sum_{s \in b_1} \psi[e^{b_2}_{s,t}]$, U_t be $\sum_{s \in p(b)} \psi[e^b_{t,s}]$. Load vectors of configurations $\psi_b, \psi_{b_1}, \psi_{b_2}$ are consistent if and only if for each $u \in b_b$ one of the constraints below must hold (See Figure 3.7b):

- $L_b + R_b = U_b$
- $|L_b - R_b| = U_b$

3.3.3 Analysis

The number of the possible ϵ-restricted partial clusters is $O(\log^{\sigma+1} n/\epsilon)$, therefore the number of the possible subproblem configurations at bag b, $\mathbb{P}_b$ is $n^{O(\log^{\sigma+1} n/\epsilon)}$. The number of the possible values for ψ, is n^{f^2}, which implies that the number of the dynamic program table entries is $n^{O(\log^{\sigma+1} n/\epsilon f^2)}$. Deciding whether configurations $(\mathbb{P}_b, \psi_b), (\mathbb{P}_{v_b}, \psi_{b_1}), (\mathbb{P}_{b_2}, \psi_{b_2})$ are consistent requires iterating over all consistent configurations which are at most equal $n^{O(\log^{\sigma+1} \frac{nf^2}{\epsilon})}$. Therefore running time is $n^{O(\log^{\sigma+1} \frac{nf^2}{\epsilon})}$ which is quasi-polynomial in n. Observe that even if treewidth is poly-logarithmic, the running time stays quasi-polynomial.

At each level of recursion we lose a factor of $(1+\epsilon/\log n)$ calculating $A[b, \mathbb{P}_b]$. Since $c \log n$ is an upper bound for the height of the tree, the approximation factor of the final solution is, at most $(1 + \frac{\epsilon}{c \log n})^{c \log n} \leq 1 + \epsilon$. Also note that the problem of finding treewidth is NP but assuming the treewidth is bounded finding a tree decomposition takes linear time.

Theorem 22. *Given $\epsilon > 0$, there is an algorithm that for an instance of k-MSC on a f treewidth graph metric with cost OPT, finds a $(1+\epsilon)$ approximate solution in time $n^{O(\log^{\sigma+1} \frac{nf^2}{\epsilon})}$ where $\sigma = O(1/\epsilon)$.*

Note that the treewidth of trees is equal to 1, therefore this solution gives a QPTAS for trees. In the next section, we will extend this algorithm to other graph metrics (doubling metrics, highway dimensions) by embedding them into logarithmic tree width decomposition trees.

Chapter 4

Generalization to k-MSC on Doubling Metrics and Highway Dimension

This section shows how to extend our algorithm to metrics with bounded doubling dimension and bounded highway dimension via embedding to Bounded Treewidth graphs. Firstly we show that an embedding of a graph metric with $(1+\epsilon)$ distortion gives us a $(1+\epsilon)$ solution in base metric, and then we show that we can assume that the aspect ratio of the given metric is polynomially bounded and then we use our QPTAS on bounded treewidth graphs as a black box.

4.1 Embedding Lemma

Suppose G is an input graph for an instance of k-MSC. Suppose we have a probabilistic embedding $\kappa : G \to B$ where B is the host graph metric with distortion $(1+\epsilon)$ i.e for any $u, v \in V(G)$ we have

$$d_G(u,v) \leq \mathbb{E}[d_B(\kappa(u), \kappa(v))] \leq (1+\epsilon) d_G(u,v)$$

We will show that if we can find a near optimum solution in B we can convert an instance of k-MSC on input graph G to an instance of k-MSC on B and move the solution of B back to G with additional ϵ cost which is proved in the next lemma.

Lemma 23. *Given $\epsilon > 0$, let $\kappa : G \to B$ be a probabilistic embedding with $(1+\epsilon)$ distortion. For instance, I_G of k-MSC with optimal solution OPT_G we can create an instance I_B of k-MSC on B with optimal solution OPT_B and then move back OPT_B to a solution of I_G with the cost at most $(1+\epsilon)OPT_G$.*

Proof. Note that the vertices are the same in both graphs G, B, so a solution in G is a solution in B (and vice versa). With the help of mapping $\kappa : G \to B$, we abuse the notation and for a solution S on I_G, we use $\kappa(S)$ to show the same solution embedded on I_B. Suppose that $C_1, C_2, \cdots, C_k$ are the clusters of OPT_G and $C'_1, C'_2, \cdots, C'_k$ are the clusters of OPT_B. Let cost^i_G be the cost of C_i in OPT_G where $\text{cost}_G(C) = \sum_{\{u,v \in C\}} d_G(u,v)$. Let cost^i_B be the cost of C'_i in OPT_B where $\text{cost}_B(C) = \sum_{\{u,v \in C\}} d_B(u,v)$. We can write $\text{OPT}_G = \sum_i \text{cost}_G(C_i)$ and $\text{OPT}_B = \sum_i \text{cost}_B(C'_i)$. Let $\text{cost}_G(\kappa^{-1}(OPT_B))$ be the cost of clusters $C'_1, C'_2, \cdots, C'_k$ of solution OPT_B as a solution for I_G. Our goal is to show that $\text{cost}_G(\kappa^{-1}(OPT_B)) \leq (1+\epsilon)\text{OPT}_G$.

Since in expected we have $d_G(u,v) \leq d_B(\kappa(u), \kappa(v))$, we know that for any solution S, $\text{cost}_G(S) \leq \text{cost}_B(\kappa(S))$, therefore for any solution S_G of I_G, we have $\text{OPT}_B = \text{cost}_B(OPT_B) \leq \text{cost}_B(\kappa(S_G))$ and for any solution S_B of I_B, we have $\text{OPT}_G = \text{cost}_G(OPT_G) \leq \text{cost}_G(\kappa^{-1}(S_B))$.

$$\begin{aligned}
\text{cost}_G(\kappa^{-1}(OPT_B)) &\leq \text{cost}_B(OPT_B) \leq \text{cost}_B(\kappa(OPT_G)) \\
&= \sum_{C_i \in OPT_G} \text{cost}_B(C_i) \\
&= \sum_{C_i} \Big(\sum_{\{u,v \in C_i\}} d_B(u,v) \Big) \\
&\leq \sum_{C_i} \Big(\sum_{\{u,v \in C_i\}} (1+\epsilon) d_G(\kappa^{-1}(u), \kappa^{-1}(v)) \Big) \\
&= (1+\epsilon) \sum_{C_i} \text{cost}_G(\kappa^{-1}(C_i)) \\
&= (1+\epsilon)\text{OPT}_G
\end{aligned} \tag{4.1}$$

□

4.2 Polynomial Aspect Ratio

In this section, based on [10], we will prove the following standard result that one can assume the aspect ratio of the given metric of a k-MSC instance is polynomially bounded. Recall that aspect ratio of a metric (X,d) is $\frac{\max_{u,v \in X} d(u,v)}{\min_{u,v \in X} d(u,v)}$.

Lemma 24. *Given an $\rho(n)-$approximation algorithm for k-MSC on instances with polynomial bounded aspect ratio that runs in time T, we can obtain a $\rho(n)+o(1)$-approximation algorithm for k-MSC on all instances running in time $T+poly(n)$.*

Proof. Given an instance I of k-MSC, we find an upper bound estimate M for the optimal cost on I (denoted by $OPT(I)$) by using 2.611-approximation algorithm for k-median of [9], that runs in polynomial time for general metrics. Let $OPT_{\text{k-median}}$ be the cost of the optimal solution for $k-$median, one can verify that $n.OPT_{\text{k-median}}$ is an upper bound for k-BM which together with the 2 factor relation between k-MSC and k-BM implies that $OPT(I) \leq 6n.OPT_{\text{k-median}}$ and $OPT_{\text{k-median}} \leq OPT(I)$. (We can write $M/3 \leq OPT(I) \leq 6nM$). View the metric space (X,d) as a complete graph. For each edge with weight more than $12n.\rho(n)M$, reduce their weight to $12n.\rho(n)M$, and for short edges of length less than $\frac{M}{n^3}$ increase their length to $\frac{M}{n^3}$. Computing all-pairs shortest paths gives us a new metric (X,d'), let I' be the corresponding k-MSC instance on (X,d'). Note that the aspect ratio of (X,d') will be at most $12n^4.\rho(n)$.

Let S^* be the optimal solution of clusters for I and use the given algorithm to get an $\rho(n)$ approximate solution S'. We claim that S' is a $(\rho(n)+o(1))$-approximate solution for the original instance I. The cost of S^* in I' is greater than in I, by at most $n^2(\frac{M}{n^3})$, for the reason that $OPT(I) \leq M$, no long edges would be used in S^* and there is only the increase of short edges which implies that $OPT(I') \leq OPT(I) + n^2(\frac{M}{n^3}) = (1+o(1))OPT(I)$. Since S' is a $\rho(n)$-approximation for I', and the weight of long edges are reduced to $12n.\rho(n)M$, none of the clusters in S' will contain a long edge. Therefore

$$\text{COST}_I(S') \leq \text{COST}_{I'}(S') \leq \rho(n)OPT(I') \leq \rho(n)(1+o(1))OPT(I)$$

completes the proof.

□

4.3 k-MSC on Doubling Metrics

In this section, we will use the following embedding lemma from [24] of graphs of bounded Doubling Dimensions into Bounded Tree Width Graphs to present a QPTAS for k-MSC on metrics with constant doubling dimensions.

Lemma 25 (Theorem 9 in [24]). *Let (X, d) be a metric with doubling dimension D and aspect ratio Δ. For any $\epsilon > 0$, (X, d) can be probabilistically approximated by a family of treewidth k-metrics for $k \leq 2^{O(D)}\lceil(\frac{4D\log\Delta}{\epsilon}^{D})\rceil$.*

The following Theorem is a direct implication of Lemmas 23, 25 and Theorem 22.

Theorem 26. *Given $\epsilon, D > 0$, there is an algorithm that for an instance of k-MSC on a D doubling dimension graph, finds a $(1+\epsilon)$ approximate solution in time $n^{O(4^{\alpha}\frac{\log n}{\epsilon}(D\frac{\log n}{\epsilon})^{2D})}$ where $\alpha = O(D)$*

Proof. Based on Theorem 22 for a graph with treewidth f, we can find a $(1+\epsilon)$ approximate solution in time $n^{O(\log^{\sigma+1}\frac{nf^2}{\epsilon})}$, where $\sigma = O(1/\epsilon)$. Based on Lemma 25 there is an embedding of a doubling metric into a graph with treewidth f such that $f \leq 2^{O(D)}\lceil(\frac{4D\log\Delta}{\epsilon}^{D})\rceil$ with $(1+\epsilon)$ distortion. Finally, using Lemma 23, an approximation scheme on bounded treewidth graphs implies an approximation on doubling metrics. Thanks to Lemma 24, we have $\Delta = O(n^5)$, therefore the running time of algorithm will be $n^{O(4^{\alpha}\frac{\log n}{\epsilon}(D\frac{\log n}{\epsilon})^{2D})}$ where $\alpha = O(D)$. □

Euclidean space $\mathbb{R}^m$ has a doubling dimension $\Theta(m)$, which is constant for constant m. As an immediate corollary of Theorem 26 we can use this theorem for Euclidean metrics. Lemma 25 embeds $\mathbb{R}^m$ into a graph with treewidth $O(\log n)$.

Corollary 27. *There exist a QPTAS for k-MSC on Euclidean metrics $\mathbb{R}^m$ for any fixed m.*

4.4 k-MSC on Highway Dimensions

In this section, analogous to the previous section, we will use the embedding of graphs of bounded highway dimensions into bounded treewidth graphs from [14] to devise a QPTAS for k-MSC on constant highway dimensions.

Lemma 28 (Theorem 1.3 in [14]). *Let G be a graph with highway dimension D of violation $\lambda > 0$ and aspect ratio Δ. For any $\epsilon > 0$, there exists a polynomial time computable probabilistic embedding E of G with treewidth $(\log\Delta)^{O(\log^2(\frac{D}{\epsilon\lambda})/\lambda)}$ and expected distortion $(1+\epsilon)$.*

Theorem 29. *Given $\epsilon > 0, D > 0, \lambda > 0$, there exists a quasi-polynomial time algorithm that, for an instance of k-MSC on a D highway dimension graph with violation λ and aspect ratio Δ, finds a $(1+\epsilon)$ approximate solution in time $n^{O(\frac{\log n}{\epsilon}(\log n)^{2\alpha})}$ where $\alpha = O(\log^2(\frac{D}{\epsilon\lambda})/\lambda)$.*

Proof. Observe that Theorem 22 shows that for a graph with tree width f, there is a $(1+\epsilon)$ approximate algorithm running in time $n^{O(\log^{\sigma+1}\frac{nf^2}{\epsilon})}$. Based on lemma 28 there is an embedding of a highway dimension metric into a graph with treewidth f such that $f \leq (\log\Delta)^{O(\log^2(\frac{D}{\epsilon\Delta}/\lambda))}$ with $(1+\epsilon)$ distortion. Finally, using lemma 23, an approximation scheme on treewidth graphs implies an approximation on highway dimensions. Thanks to lemma 24, we have $\Delta = O(n^5)$, therefore the running time of this algorithm is $n^{O(\frac{\log n}{\epsilon}(\log n)^{2\alpha})}$ where $\alpha = O(\log^2(\frac{D}{\epsilon\lambda})/\lambda)$. □

Chapter 5

Conclusion and Future Problems

In this work, we gave a quasi-polynomial time approximation scheme for Min Sum k-Clustering Problem on graphs with bounded treewidth and bounded doubling metrics. An immediate question is to make it polynomial time. Since the number of cluster types is poly-logarithmic, it is not clear how to improve our algorithm to get a PTAS. Although [11] proved that Metric k-MSC is APX-HARD to a factor of 1.415.

One possible idea to find a PTAS for $k-$MSC on other metrics such as the Euclidean metric, is to use the idea of simplifying the $k-$MSC objective function. Actually this simplification is a generalization of corsets. Instead of storing $O(n^2)$ distances between all of the pairs of points, we show that a selection of $O(n)$ pairs is enough to approximate the k-min sum objective. One may use this idea on different problems to work with a candidate subset instead of the whole input.

References

[1] M. Adamczyk, J. Byrka, J. Marcinkowski, S. Meesum, and M. Włodarczyk, *Constant factor fpt approximation for capacitated k-median*, Sep. 2018.

[2] M. Aigner and G. M. Ziegler, "Cayley's formula for the number of trees," in *Proofs from THE BOOK*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 201–206, ISBN: 978-3-642-00856-6. DOI: 10.1007/978-3-642-00856-6_30. [Online]. Available: https://doi.org/10.1007/978-3-642-00856-6_30.

[3] S. Banerjee, R. Ostrovsky, and Y. Rabani, *Min-sum clustering (with outliers)*, Nov. 2020.

[4] Y. Bartal, M. Charikar, and D. Raz, "Approximating min-sum k-clustering in metric spaces," STOC '01, pp. 11–20, 2001. DOI: 10.1145/380752.380754. [Online]. Available: https://doi.org/10.1145/380752.380754.

[5] Y. Bartal, M. Charikar, and D. Raz, "Approximating min-sum k-clustering in metric spaces," in *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, ser. STOC '01, Hersonissos, Greece: Association for Computing Machinery, 2001, pp. 11–20, ISBN: 1581133499. DOI: 10.1145/380752.380754. [Online]. Available: https://doi.org/10.1145/380752.380754.

[6] B. Behsaz, Z. Friggstad, M. Salavatipour, and R. Sivakumar, "Approximation algorithms for min-sum k-clustering and balanced k-median," vol. 9134, Jul. 2015, pp. 116–128, ISBN: 978-3-662-47671-0. DOI: 10.1007/978-3-662-47672-7_10.

[7] H. L. Bodlaender and T. Hagerup, "Parallel algorithms with optimal speedup for bounded treewidth," *SIAM Journal on Computing*, vol. 27, no. 6, pp. 1725–1746, 1998. DOI: 10.1137/S0097539795289859. eprint: https://doi.org/10.1137/S0097539795289859. [Online]. Available: https://doi.org/10.1137/S0097539795289859.

[8] J. Bourgain, "On lipschitz embedding of finite metric spaces in hilbert space," *Israel Journal of Mathematics*, vol. 52, pp. 46–52, 1985.

[9] J. Byrka, T. Pensyl, B. Rybicki, A. Srinivasan, and K. Trinh, "An improved approximation for k-median and positive correlation in budgeted optimization," *ACM Trans. Algorithms*, vol. 13, no. 2, Mar. 2017, ISSN: 1549-6325. DOI: 10.1145/2981561. [Online]. Available: https://doi.org/10.1145/2981561.

[10] V. Cohen-Addad, A. Gupta, A. Kumar, E. Lee, and J. Li, "Tight FPT Approximations for k-Median and k-Means," in *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, C. Baier, I. Chatzigiannakis, P. Flocchini, and S. Leonardi, Eds., ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 132, Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 42:1–42:14, ISBN: 978-3-95977-109-2. DOI: 10.4230/LIPIcs.ICALP.2019.42. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2019/10618.

[11] V. Cohen-Addad, C. S. Karthik, and E. Lee, "On approximability of clustering problems without candidate centers," in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 2635–2648. DOI: 10.1137/1.9781611976465.156. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611976465.156. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611976465.156.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd. The MIT Press, 2001, ISBN: 0262032937.

[13] A. Czumaj and C. Sohler, "Small space representations for metric min-sum k-clustering and their applications," in *Proceedings of the 24th Annual Conference on Theoretical Aspects of Computer Science*, ser. STACS'07, Aachen, Germany: Springer-Verlag, 2007, pp. 536–548, ISBN: 9783540709176.

[14] A. E. Feldmann, W. S. Fung, J. Könemann, and I. Post, "A $(1+\varepsilon)$-embedding of low highway dimension graphs into bounded treewidth graphs," *SIAM Journal on Computing*, vol. 47, no. 4, pp. 1667–1704, 2018. DOI: 10.1137/16M1067196. eprint: https://doi.org/10.1137/16M1067196. [Online]. Available: https://doi.org/10.1137/16M1067196.

[15] A. Frieze and M. Jerrum, "Jerrum, m.: Improved approximation algorithms for max k-cut and max bisection. algorithmica 18(1), 67-81," *Algorithmica*, vol. 18, pp. 67–81, May 1997. DOI: 10.1007/BF02523688.

[16] N. Guttmann-Beck and R. Hassin, "Approximation algorithms for min-sum p-clustering," *Discrete Applied Mathematics*, vol. 89, no. 1-3, pp. 125–142, 1998.

[17] R. Hassin and E. Or, "Min sum clustering with penalties," *European Journal of Operational Research*, vol. 206, no. 3, pp. 547–554, 2010, ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2010.03.004.

[Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221710001724.

[18] J. Heinonen, *Lectures on Analysis on Metric Spaces*. Jan. 2001, ISBN: 978-1-4612-6525-2. DOI: 10.1007/978-1-4613-0131-8.

[19] V. Kann, S. Khanna, J. Lagergren, and A. Panconesi, "On the hardness of approximating max k-cut and its dual," *Chicago J Theoret Comput Sci*, May 1997.

[20] J. Kleinberg and É. Tardos, *Algorithm Design*. Addison Wesley, 2006.

[21] "Metric structures for riemannian and non-riemannian spaces," 2007, Zapis avtomatsko prevzet s spletne strani založnika Springer, ISSN: 978-0-8176-4583-0. [Online]. Available: doi:10.1007/978-0-8176-4583-0.

[22] S. Sahni and T. Gonzalez, "P-complete approximation problems," *J. ACM*, vol. 23, no. 3, pp. 555–565, Jul. 1976, ISSN: 0004-5411. DOI: 10.1145/321958.321975. [Online]. Available: https://doi.org/10.1145/321958.321975.

[23] S. Sahni and T. F. Gonzalez, "P-complete approximation problems," *Journal of the ACM (JACM)*, vol. 23, pp. 555–565, 1976.

[24] K. Talwar, "Bypassing the embedding: Algorithms for low dimensional metrics," in *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, ser. STOC '04, Chicago, IL, USA: Association for Computing Machinery, 2004, pp. 281–290, ISBN: 1581138520. DOI: 10.1145/1007352.1007399. [Online]. Available: https://doi.org/10.1145/1007352.1007399.

[25] W. F. de la Vega, M. Karpinski, C. Mathieu, and Y. Rabani, "Approximation schemes for clustering problems," in *STOC '03*, 2003.

[26] D. B. West, *Introduction to Graph Theory*, 2nd ed. Prentice Hall, Sep. 2000, ISBN: 0130144002.

[27] Wikipedia contributors, *Apx — Wikipedia, the free encyclopedia*, [Online; accessed 11-November-2022], 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=APX&oldid=1040375348.

[28] Wikipedia contributors, *Ptas reduction — Wikipedia, the free encyclopedia*, [Online; accessed 23-November-2022], 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=PTAS_reduction&oldid=1102026234.