Approximation Schemes for Min-Sum k-Clustering

Ismail Naderi

Department of Computer Science, University of Alberta, Edmonton, Canada

Mohsen Rezapour

Department of Computer Science, University of Alberta, Edmonton, Canada
Department of Computer Science and Statistics, K. N. Toosi University of Technology, Iran

Mohammad R. Salavatipour[1]

Department of Computer Science, University of Alberta, Edmonton, Canada

Abstract

We consider the Min-Sum k-Clustering (k-MSC) problem. Given a set of points in a metric which is represented by an edge-weighted graph $G = (V, E)$ and a parameter k, the goal is to partition the points V into k clusters such that the sum of distances between all pairs of the points within the same cluster is minimized.

The k-MSC problem is known to be APX-hard on general metrics. The best known approximation algorithms for the problem obtained by Behsaz, Friggstad, Salavatipour and Sivakumar [Algorithmica 2019] achieve an approximation ratio of $O(\log |V|)$ in polynomial time for general metrics and an approximation ratio $2 + \epsilon$ in quasi-polynomial time for metrics with bounded doubling dimension. No approximation schemes for k-MSC (when k is part of the input) is known for any non-trivial metrics prior to our work. In fact, most of the previous works rely on the simple fact that there is a 2-approximate reduction from k-MSC to the balanced k-median problem and design approximation algorithms for the latter to obtain an approximation for k-MSC.

In this paper, we obtain the first Quasi-Polynomial Time Approximation Schemes (QPTAS) for the problem on metrics induced by graphs of bounded treewidth, graphs of bounded highway dimension, graphs of bounded doubling dimensions (including fixed dimensional Euclidean metrics), and planar and minor-free graphs. We bypass the barrier of 2 for k-MSC by introducing a new clustering problem, which we call min-hub clustering, which is a generalization of balanced k-median and is a trade off between center-based clustering problems (such as balanced k-median) and pair-wise clustering (such as Min-Sum k-clustering). We then show how one can find approximation schemes for Min-hub clustering on certain classes of metrics.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Approximation Algorithms, Clustering, Dynamic Programming

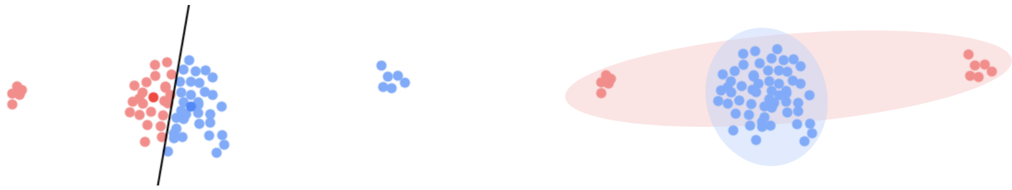
Digital Object Identifier 10.4230/LIPIcs.ESA.2023.45

1 Introduction

Clustering is a fundamental problem in many areas of data analysis and machine learning and has many applications across various fields. 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 cluster.

In the well-known *center-based* k-clustering problems (such as 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

* This research project was supported by the last authors NSERC Discovery grant.



(a) (b)

Figure 1 Clustering of a set of points: (a) a possible center-based clustering induced by a Voronoi diagram of two cluster centers , and (b) a min-sum k-clustering solution for $k = 2$. Observe that the min-sum k-clustering solution in (b) places all outliers into a separate cluster.

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 efficient for clustering large datasets as the main task reduces to selecting k centers; once we decided on the set of centers, points that are closest to a particular center are considered to be part of the cluster represented by that center. Center-based clustering algorithms are not always precise because they heavily rely on the assumption that each cluster has a *spherical* shape and hence can be represented by one center.

In *pair-wise* k-clustering, on the other hand, the goal of partitioning is to minimize the dissimilarity between pairs of points that are in the same cluster. For example, in the case of the *k-diameter* problem, the goal is to minimize the maximum distance between any two points in a cluster; or in the *min-sum k-clustering* problem, the goal is to minimize the sum of distances between all pairs of the points within the same cluster.

Unlike center-based clustering problems, min-sum k-clustering (which is the main focus of this paper) is less sensitive to the shape of clusters because it forms clusters based on the pair-wise distances between points rather than the distances of points to their cluster center. Also, as observed in [8], min-sum k-clustering can handle (detect) noises (outliers) in an effective way: in scenarios where data include well-defined clusters and a limited number of scattered noises (outliers), assigning an outlier to one of the clusters would be more costly than placing it in an outlier cluster that holds all the outliers. This results in a solution with a separate cluster specifically for outliers, avoiding the limitations of center-based clustering algorithms, which rely on Voronoi partitioning to divide the data space into clusters and are unable to handle overlapping cluster spaces. See Figure 1.

We now formally define the min-sum k-clustering problem. Given a metric space over a set of n points V with metric distances $d(u, v)$ between any two $u, v \in V$. We assume the metric is induced by an edge-weighted graph $G = (V, E)$. In the **Min-Sum k-Clustering** problem (**k-MSC**), 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: $\sum_{i=1}^{k} \sum_{\{u,v\} \subseteq C_i} d(u, v)$. This problem is closely related to the **Balanced k-Median** problem (**k-BM**), with the same input as in **k-MSC**. Here, 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 $\sum_{i=1}^{k} |C_i| \sum_{v \in C_i} d(v, c_i)$.

Related Works

Sahni and Gonzalez introduced k-MSC in 1976 [13]. They showed the problem is NP-hard and provided a polynomial time k-approximation algorithm for the k-Max Cut problem, which is the dual of **k-MSC** and involves partitioning points into k clusters to maximize the distance between points in different clusters. Kann et al. [12] showed it is NP-Hard to approximate non-metric k-**MSC** within $O(n^{2-\epsilon})$ for any $\epsilon > 0$ and $k > 3$. Later, Cohen-Addad et al. [6] proved that it is NP-hard to approximate metric k-**MSC** within 1.415.

Guttman-Beck and Hassin [11] showed that k-**BM** and k-**MSC** are closely related. They showed an algorithm with ρ approximation for one of these problems implies a 2ρ approximation for the other. In the literature, most of the previous work (with a guaranteed approximation factor) for k-**MSC** make use of this reduction. Guttman-Beck and Hassin [11] showed that **k-BM** can be solved in time $n^{O(k)}$ by guessing the cluster centers and sizes and finding the minimum-cost assignment from clients to these centers. This results in a 2-approximation solution for the min-sum k-clustering problem when k is fixed. Bartal et al. [3] introduced the first polynomial time approximation algorithm for both k-**MSC** and k-**BM** in metric spaces. They devised an algorithm with an approximation factor of $O(\frac{1}{\epsilon}\log^{1+\epsilon} n)$ and running time of $n^{\frac{1}{\epsilon}}$ for k-**BM**. The algorithm is based on the embedding of metric spaces into hierarchically separated trees (HSTs). They also provided a bi-criteria approximation algorithm with a constant approximation factor with $O(k)$ clusters. Later, Behsaz et al. [4] improved the result by utilizing the properties of HSTs through a direct dynamic programming approach, leading to a $O(\log n)$ approximation algorithm for both k-**MSC** and k-**BM**. This is the current best result for general metrics. They also present a quasi-polynomial time approximation scheme for k-**BM** in metrics with constant doubling dimensions, leading to a $(2+\epsilon)$-approximation algorithm for the min-sum k-clustering problem that runs in quasi-polynomial time. More recently Banerjee et al. [2] gave a bicriteria approximation for k-**MSC** with outliers: for any $\epsilon > 0$, given an instance with n points and any integer $n' \leq n$, their algorithm finds a solution that clusters at least $(1-\epsilon)n'$ points whose cost is poly$(1/\epsilon)$ times the optimum clustering of n' points.

For small values of k, Vega et al. [9] introduced the first polynomial time approximation scheme for k-**MSC** in metric spaces. The running time of their algorithm is $O(n^{3k}2^{\epsilon^{-k^2}})$. Czumaj and Sohler [8], presented a $(4+\epsilon)$ approximation algorithm for k-**MSC** in metric spaces with a running time of linear for $k = o(\log n/\log\log n)$.

Our Results and Techniques

As mentioned earlier, the previous methods for designing approximation for k-**MSC** attempt to approximate the cost using a center-based clustering objective (such as k-**BM** [3, 4] or a capacitated version of k-median [2]). Such methods have a barrier of 2 (even for tree metrics). A key challenge in extending the framework of [4] to work directly for k-**MSC** is to develop a compact representation of the cluster types in a near-optimal solution that can capture the essence of the cluster without relying on a center.

Here we introduce a new clustering objective that is in between the pair-wise distances objective of k-**MSC** and the center-based objective of k-**BM**, which we call min-hub clustering. We show that for metrics with a nice hierarchical decomposition (such as graphs of bounded treewidth, or bounded doubling dimension), the objective of min-hub clustering is a good (namely $(1+\epsilon)$) approximation of k-**MSC** and how one can obtain an approximation scheme for the new objective (and hence one for k-**MSC**).

In center-based clustering, a cluster is represented by a single center. However, as demonstrated in Figure 1 (see the outlier cluster in red), not all k-**MSC** clusters can be represented by a single center. To address this, we explore the possibility of using multiple centers to represent a cluster. Our results show that a cluster in the k-**MSC** solution can be represented by $O_\epsilon(1)$ centers, which we refer to as *hubs*, while incurring an error of $(1+\epsilon)$. Specifically, let H be a set of hubs. The *hub-distance* between two points u and v in a cluster C is defined as the shortest path between the points that passes through hub points in H.

Our results show that there exists a set of H of constant size (depending on ϵ) such that the sum of distances between all pairs of points within C is "almost" equal to the sum of hub-distances between pairs of points in C. This suggests that the network interconnecting the hubs, called the *backbone structure*, carries the majority of the connection flow in the cluster. We represent a cluster by the type of its backbone structure and the distribution of points around its hubs.

In Section 2, we consider the special case of tree metrics. We construct a dynamic program for k-**MSC** on tree metrics that have a logarithmic height. In Section 3, we extend our approach to cover metrics with bounded treewidth, thereby covering general trees as well.

▶ **Theorem 1.** *There is a quasi-polynomial time algorithm that, given an instance of k-**MSC** on a metric of treewidth f, for any $\epsilon > 0$ finds a $(1+\epsilon)$-approximate solution in time $n^{O(f^2+(\frac{\log n}{\epsilon})^{\sigma+1})}$, where σ depends on ϵ.*

It is worth pointing out that, if one tries to extend the result from trees to graphs with treewidth f in a natural way, the algorithm will have a run time of the form $n^{(\frac{\log n}{\epsilon})^{f^2+\sigma+1}}$ (instead of $n^{O(f^2+(\frac{\log n}{\epsilon})^{\sigma+1})}$), which is still quasi-polynomial for fixed f, but will not be quasi-polynomial if $f = \text{Polylog}(n)$. This is essential to obtain the next three theorems, as we use embeddings into graphs with treewidths $f = \text{Polylog}(n)$.

In Section 4, using frameworks from [14], [10], and [7], we expand our results to three additional metric classes: bounded doubling metrics, bounded highway dimension metrics, and minor-free metrics, respectively.

▶ **Theorem 2.** *There is a quasi-polynomial time algorithm that, given an instance of k-**MSC** on a metric of doubling dimension D, for any $\epsilon > 0$ finds a $(1+\epsilon)$ approximate solution in time $n^{O((\frac{D\log n}{\epsilon})^{2D}+(\frac{\log n}{\epsilon})^{\sigma+1})}$.*

▶ **Theorem 3.** *There is a quasi-polynomial time algorithm that, given an instance of k-**MSC** on a metric highway dimension D and violation λ, for any $\epsilon > 0$ finds a $(1+\epsilon)$ approximate solution in time $n^{O((\log n)^{\alpha}+(\frac{\log n}{\epsilon})^{\sigma+1})}$, where $\alpha = O(\log^2(\frac{D}{\epsilon\lambda})/\lambda)$.*

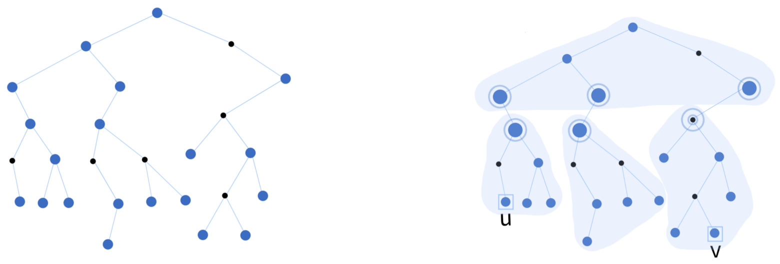
▶ **Theorem 4.** *There is a quasi-polynomial time algorithm that, given an instance of k-**MSC** in minor-free metrics, for any $1/2 > \epsilon > 0$ finds a $(1+\epsilon)$ approximate solution in time $n^{\epsilon^{-O(1)}\log^{O(1)} n}$.*

2 The k-MSC Problem in Tree Metrics

In this section, we construct a dynamic program for k-**MSC** on trees. Consider metric (V,d) induced by an edge-weighted tree $T = (V,E)$. Let $w(e)$ denote the weight of edge e in E.

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 *level* of each node is the number of edges on the path from it to r. The *height* of the tree is the level of the leaf node with the highest level. We use T_v to denote the *subtree* rooted at v, $V(T_v)$ and $E(T_v)$ to denote the vertex set and the edge set of T_v, respectively. By introducing zero-weight edges and nodes, we convert the tree into an equivalent binary tree. Note that the resulting binary tree has at most $2|V|$ nodes.

We use $C \subseteq V$ to denote a cluster and $D(C)$ to denote the total sum of the distances between all pairs of points in C; i.e., $D(C) = \sum_{\{u,v\}\subseteq C} d(u,v)$. We use $H \subseteq V$ to indicate a set of points referred to as *hubs*. The distance between any two points u and v in C, when measured through hubs in H, is called the *hub-distance* and is denoted by $d_H(u,v)$. This



(a) **(b)**

Figure 2 (a) A cluster on the tree, where the blue circles specify points of this cluster. (b) Shaded regions highlight the resulting groups by applying Lemma 5 on the cluster. Notice that the distance between any two points of the cluster that belong to different groups (such as u and v) is equal to their hub-distance, $d_H(u, v)$, as long as H contains the border nodes of the groups. The larger circles around the nodes depict the border nodes of the groups (so the proper hubs of the cluster).

Algorithm 1 Tree Partitioning Algorithm

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

is the length of the shortest path between the two points that goes through hub points in H; i.e., $d_H(u, v) = \min_{h_1,h_2\in H} \big(d(u, h_1) + d(h_1, h_2) + d(h_2, v)\big)$. Let $p_H(u, v)$ represent the path between points u and v that passes through hub points in H and has the length of $d_H(u, v)$. The *sum of pairwise hub-distances* for the points in C is represented by $D_H(C)$ and is equal to the total sum of the hub-distances between all pairs of points in C; i.e., $D_H(C) = \sum_{\{u,v\}\subseteq C} d_H(u, v)$. Note that $D_V(C)$=$D(C)$.

The following lemma shows how to find a (constant-size) set of hubs that represents a given cluster in metrics induced by a tree metric. See Figure 2. For a subset of nodes $g \subseteq V$, we use $\delta(g) = \{v \in g : \ uv \in E \ \& \ u \notin g\}$ to denote the *border nodes* of g.

▶ **Lemma 5.** *Let $C \subseteq V$ be a cluster and let $T = (V, E)$ be a given binary tree. For any $\nu > 0$, there exists a partition of V into a set of groups $\mathbb{C}_\nu = \{g_1, \ldots, g_\sigma\}$ such that all of the following properties hold: (i) the subgraph induced by each group $g \in \mathbb{C}_\nu$ is connected. (ii) for each group $g \in \mathbb{C}_\nu$, $|g \cap C| \in [1, \max\{1, \nu|C|\}]$. (iii) $|\mathbb{C}_\nu| = O(1/\nu)$. (iv) $\forall g \in \mathbb{C}_\nu, \quad |\delta(g)| = O(1/\nu)$.*

Proof. We use Algorithm 1 to compute $\mathbb{C}_\nu$. The algorithm iteratively selects a subtree $T_{\hat{v}}$, with approximately $\frac{\nu}{2}$ of the total number of points $|C|$, adds the vertex set $V(T_{\hat{v}})$ to $\mathbb{C}_\nu$, and removes $T_{\hat{v}}$ from T. The number of iterations (i.e. the number of groups made by the algorithm) is at most $2/\nu$, and every vertex of V belongs to one group.

Note that there is at most one edge between any two groups, so $|\delta(g)| = O(1/\nu)$, $\forall g \in \mathbb{C}_\nu$.

The subgraphs induced by g_i's are connected by construction. Thus, the algorithm has constructed a partition with the desired properties, as shown in Figure 2. ◀

Note that each cluster covers only a subset of points, however, the groups of the cluster always include all the nodes of V. Given a cluster $C \subseteq V$ and a constant $\nu > 0$, let $\mathbb{C}_\nu = \{g_1, \ldots, g_\sigma\}$ be the groups obtained by applying Lemma 5 on C with the given value of ν. We let $H_\nu(C) = \cup_{i=1}^{\sigma} \delta(g_i)$ denote the **ν-proper hubs** of the cluster. Notice that the size of $|H_\nu(C)|$ is constant, depending on ν.

Given a cluster $C \subseteq V$ and a constant $\nu > 0$, consider the ν-proper hubs of the cluster, $H_\nu(C)$. We refer to $\text{cost}_{H_\nu}(C) = \sum_{i=1}^{\sigma} \sum_{j=i+1}^{\sigma} \sum_{u \in g_i \cap C, v \in g_j \cap C} d_{H_\nu(C)}(u, v)$ as the **ν-approximate cost** of the cluster. This represents the sum of hub-distances between all pairs of points of C belonging to different groups. The following lemma shows that $\text{cost}_{H_\nu}(C)$ is "almost" equal to $D(C)$, when the value of ν is sufficiently small.

▶ **Lemma 6.** *For each such cluster C and any $\nu > 0$, $cost_{H_\nu}(C) \leq D(C) \leq (1+O(\nu))cost_{H_\nu}(C)$.*

The proof is omitted due to page limitations.

To make the presentation of our dynamic programming algorithm simpler, we formulate a problem with the same input and objective as the min-sum k-clustering problem, but the cost of clusters is evaluated by $\text{cost}_{H\nu}(C)$ instead of $D(C)$: Given a constant $\nu > 0$ and an edge-weighted tree $T = (V, E)$. In the **Min-Hub k-Clustering** problem (**k-MHC**), we are asked to partition points V into k clusters $C_1, ..., C_k$ to minimize $\sum_{i=1}^{k} \text{cost}_{H_\nu}(C_i)$.

▶ **Theorem 7.** *Let $\epsilon > 0$. A $(1+\epsilon)$-approximation for k-**MHC** will imply a $(1+O(\epsilon))$-approximation for k-MSC on tree metrics.*

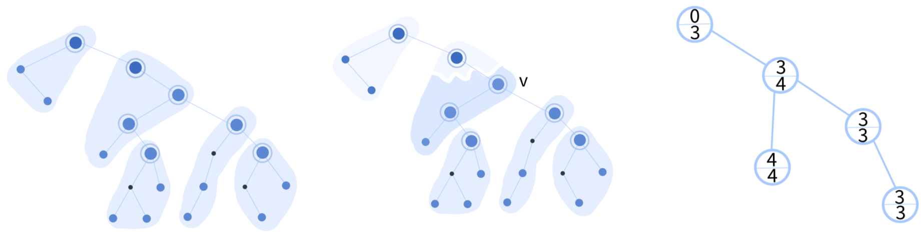
The proof is omitted due to page limitations.

2.1 QPTAS for k-MHC on Trees with Logarithmic Heights

Theorem 7 tells us that if we try to find a clustering which optimizes the objective of k-**MHC**, then the same clustering has a good value for the objective of k-**MSC**. Suppose we are given a tree $T = (V, E)$ that has a logarithmic height and a constant $\nu > 0$. Let OPT be the minimum cost of partitioning V into k clusters $C_1, C_2, \cdots, C_k$ with the total cost being $\sum_{i=1}^{k} \text{cost}_{H_\nu}(C_i)$. Given $\epsilon > 0$, we will present a dynamic program that finds a $(1+\epsilon)$-approximation of OPT. This, as a result of Theorem 7, leads to a $(1+O(\epsilon))$ approximation solution for k-**MSC** on trees with logarithmic heights. Then, in the next section, we will extend the dynamic program to cover metrics with bounded treewidth, thereby covering general trees as well.

Preprocessing. We assume each node of the tree has a *token* on it and our goal is to cluster the tokens. We may modify the tree by adding dummy edges (with zero weight) and dummy nodes (that do not have tokens). Throughout this section, we refer to a node with a token as a *point* and a node without a token as a *vertex*. By introducing zero-weight edges and nodes, we convert the tree into an equivalent binary tree in which the points are only located on distinct leaves. We repeatedly remove leaves with no tokens until there is no such leaf in the tree. We also repeatedly remove internal vertices (with no token) of degree two by consolidating their incident edges into one edge of the total weight.

Cluster, Backbone Tree, and Partial Cluster Types. Let $\nu > 0$ and consider a cluster $C \subseteq V$. Suppose $\mathbb{C}_\nu = \{g_1, \ldots, g_\sigma\}$ are the groups by Lemma 5. We define a tree called the **backbone tree** of C, with nodes corresponding to groups $g_1, \ldots, g_\sigma$. This tree has edges between nodes whose corresponding groups are connected by an edge. We use g_i



(a) **(b)** **(c)**

Figure 3 (a) A cluster and its corresponding groups. (b) The partial cluster with respect to T_v. (c) The corresponding backbone tree whose nodes are labelled according to their sizes/weights.

to refer to both the group and the corresponding node in the backbone tree. According to Cayley's formula [1], the number of different trees that can be formed by $\tilde{n}$ labeled nodes is $\tilde{n}^{\tilde{n}-2}$. Hence the cluster's backbone tree has one of the types $1, 2, \cdots, \sigma^{\sigma-2}$.

Each cluster C is associated with a pair $(t_b, \vec{w})$ (referred to as the **cluster type** of C), where t_b is an integer between 1 and $\sigma^{\sigma-2}$ and represents the type of the cluster's backbone tree, and $\vec{w}$ is a vector representing the weights of each node in the backbone tree, with $\vec{w}[i] = |g_i \cap C|$ being the number of points in the i-th group of the cluster; see Figure 3.

The maximum number of ways to assign weights to nodes of a backbone tree is n^σ, where $n = |V|$. To keep the number of different cluster types manageable, we store the group weights approximately by rounding them to the nearest *threshold value*. This reduces the number of possible ways to assign weights to nodes of a backbone tree to a poly-logarithmic number and so allows for a more compact representation of the cluster types.

▶ **Definition 8.** *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 $\tau = O(\frac{\log n}{\epsilon})$. 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.*

By rounding the weights of groups to the nearest threshold value, the number of different cluster types is reduced to $O(\sigma^{\sigma-2}(\frac{\log n}{\epsilon'})^\sigma))$, where $\sigma = O(1/\nu)$. We will show that, by choosing the number of thresholds appropriately large, the DP solution will have a multiplicative error of at most $1 + O(\epsilon)$ (provided that the tree has a logarithmic height).

For every cluster $C \subset V$ and every node $v \in V$, the part of cluster that falls into T_v is referred to as the *partial cluster of C with respect to v*. To represent such a partial cluster, we associate it with a triple $(t_c, \gamma_v, \vec{s}_v)$, where t_c is an integer between 1 and $O(\sigma^{\sigma-2}(\frac{\log n}{\epsilon'})^\sigma))$ and represents the type of the cluster, γ_v is the *split group* of the partial cluster and specifies the group that includes the node v, and $\vec{s}_v$ is a vector representing the sizes of each group of the partial cluster that intersects with the tree T_v, with $\vec{s}_v[i] = |(g_i \cap C) \cap V(T_v)|$ being the number of points in the i-th group that intersect with $V(T_v)$; see Figure 3. Similar to the group weights, the group sizes are stored approximately by rounding them to the nearest threshold value. This results in a reduction of the number of partial cluster types to $O(\sigma^{\sigma-2}(\frac{\log n}{\epsilon'})^\sigma))$. Observe that a partial cluster C with respect to root r is actually the full cluster C. This means that for every group i in the cluster, the value $\vec{s}_r[i]$ is equal to $\vec{w}[i]$.

We let $\Gamma_v \subseteq \mathbb{C}_\nu$ indicate the groups, called the *inner groups*, of the partial cluster whose nodes are completely contained within the tree node T_v. For a specific partial cluster type ℓ at v, we use the notation $\gamma_v^\ell, \Gamma_v^\ell, \vec{s}_v^\ell$, and $\vec{w}^\ell$ to refer to its split group, inner groups, size, and weight vectors, respectively. It is important to note that both the weight vector $\vec{w}^\ell$ and the inner groups Γ_v^ℓ can be obtained from the triple $(t_c, \gamma_v, \vec{s}_v)$ that defines ℓ.

A partial cluster type ℓ with respect to a node v is considered **valid** if the following conditions are met: (i) the values of $\vec{s}_v^\ell[i]$ for each group i of ℓ are between 0 and $\vec{w}^\ell[i]$, (ii) the value of $\vec{w}^\ell[i]$ for each group i of ℓ is less than or equal to $\max\{\nu. \sum_{i'} \vec{w}^\ell[i'], 1\}$ (see Lemma 5), (iii) if v is a leaf node of T, then γ_v^ℓ is a leaf node of the backbone tree of ℓ (from the definition of the backbone tree). A partial cluster type ℓ is considered a **leaf partial cluster type** at a node 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, Partial Cluster Cost, and Cluster Cost. Consider a cluster C together with its groups $\mathbb{C}_\nu = \{g_1, \ldots, g_\sigma\}$ and hubs $H_\nu(C)$, and let ℓ be the type of this cluster with respect to v. Recall that, vectors $\vec{w}^\ell$ and $\vec{s}_v^\ell$ are used to show the weight and the size (with respect to the tree T_v) of the groups within the cluster C, and γ_v is used to specify the group of the cluster that includes the node v. Here, we explain how to compute the ν-approximate cost of the cluster, $\mathrm{cost}_{H_\nu}(C)$, by utilizing the information provided by these vectors.

We define the load of edge e with respect to the cluster C, its groups $\mathbb{C}_\nu$, and hubs $H_\nu(C)$ to be the number of paths $p_H(u,v)$ that include edge e over all $(u,v) \in X$, where $X = \cup_{i=1}^{\hat{\sigma}} X_i$ and $X_i = \{(u,v) : u \in \hat{g}_i \;\; v \in C \setminus \hat{g}_i\}$. Let e_v denote the edge connecting v to its parent in T. The load of edge e_v with respect to ℓ can be calculated using the following formula, represented as $load^\ell(e_v)$:

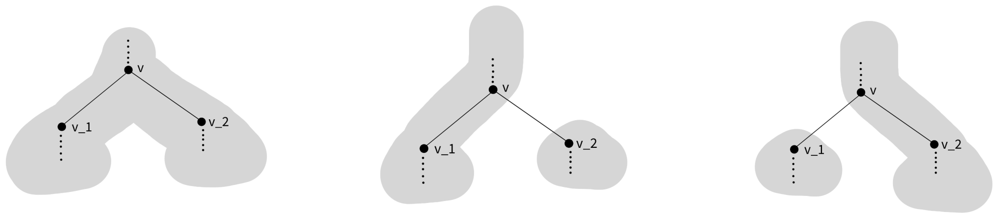
$$load^\ell(e_v) = \underbrace{\Big(\sum_{i=1, i\neq\gamma_v}^{\sigma} \vec{s}_v^\ell[i]\Big) \times \Big(\sum_{i=1}^{\sigma} (\vec{w}^\ell[i] - \vec{s}_v^\ell[i])\Big)}_{\#\text{paths crossing } e_v \text{ s.t. one of its ends is below } \gamma_v} + \underbrace{\vec{s}_v^\ell[\gamma_v] \times \Big(\sum_{i \notin \Gamma_v}^{\sigma} \vec{w}^\ell[i]\Big)}_{\#\text{paths crossing } e_v \text{ s.t. one of its ends is in } \gamma_v}$$

We define and compute the cost of a partial cluster type ℓ with respect to a node v (we denote it by $cost_v^\ell$) recursively as follows. For the base case, $cost_v^\ell = 0$, if v is a leaf node. 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. Note that the union of groups of each cluster always includes the root node r (see Algorithm 1). One can verify that $cost_r^\ell = \mathrm{cost}_{H_\nu}(C)$, if ℓ stores the exact weights and sizes of the groups of the cluster. However, here, ℓ stores weights and sizes approximately and therefore the edge load $load^\ell(e_v)$ might be overestimated by a factor of $(1+\epsilon')$ (by choosing the number of thresholds appropriately large). In the next section, we will see how this affects our approximation solution and results in a multiplicative error of at most $1 + O(\epsilon)$ (provided that the tree has a logarithmic height).

Dynamic Program

The Dynamic Program (DP) starts at the leaves of T and works its way up, exploring all possible ways to form clusters. For each node v and each possible configuration $\mathbb{P}_v$ of partial clusters with respect to v, there is an entry in the DP table. A configuration $\mathbb{P}_v \in [k]^{O(\sigma^{\sigma-2}(\frac{\log n}{\epsilon'})^\sigma)\}}$ at node v lists the number of each type of partial cluster covering points within subtree T_v. We let $A[v, \mathbb{P}_v]$ store the minimum cost to form a set of partial clusters, which match the configuration $\mathbb{P}_v$, and cover all points in T_v. Observe that the number of such subproblems is at most $n^{O(\sigma^{\sigma-2}(\frac{\log n}{\epsilon'})^\sigma)}$.

Consider a node v in the tree T. Assume for now that we have access to a table $\lambda[\mathbb{P}_v, \mathbb{P}_{v_1}, \mathbb{P}_{v_2}]$, where $\mathbb{P}_v$ is the configuration at node v, and $\mathbb{P}_{v_1}$ and $\mathbb{P}_{v_2}$ are the configurations at its children nodes v_1 and v_2, respectively. The table λ indicates whether the configurations $\mathbb{P}_v$, $\mathbb{P}_{v_1}$, and $\mathbb{P}_{v_2}$ are *consistent*, meaning that there is a solution where the descriptions of partial clusters below nodes v, v_1, and v_2 match the configurations $\mathbb{P}_v$, $\mathbb{P}_{v_1}$, and $\mathbb{P}_{v_2}$,



(a) (b) (c)

Figure 4 Consider a node v and its children v_1, v_2. There are three possible scenarios in which v, v_1, and v_2 may belong to one or two groups of a cluster. (a) is depicting the case where all three nodes are in the same group, (b) is depicting the case that v and v_1 are in the same group, (c) is depicting the case that v and v_2 are in the same group. Note that the case where all three nodes belong to different groups does not happen due to Algorithm 1.

respectively. We shall describe how to compute λ. We will compute the subproblems $A[v, \mathbb{P}_v]$ in a bottom-up manner: We will compute $A[v, \mathbb{P}_v]$ after we have computed the subproblems $A[v_1, \mathbb{P}_{v_1}]$ and $A[v_2, \mathbb{P}_{v_2}]$ for the children of v. The subproblems are computed as follows:

Base Case. For every leaf node v and every configuration $\mathbb{P}_v$, set: $A[v, \mathbb{P}_v] = 0$ if there exists a type ℓ such that $\mathbb{P}_v[\ell] = 1$ and ℓ is a leaf partial cluster at v. Otherwise, set $A[v, \mathbb{P}_v] = \infty$.

Recurrence. Let $load(v) = \sum_\ell \mathbb{P}_v[\ell] load^\ell(e_v)$. For each internal node v and its children, v_1, v_2 and every combination of configurations of $\mathbb{P}_v$ on v and $\mathbb{P}_{v_1}, \mathbb{P}_{v_2}$:

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

The final solution is obtained by finding the minimum value of $A[r, \mathbb{P}_r]$ over all configurations $\mathbb{P}_r$ such that the sum of all $\mathbb{P}_r[\ell]$ values equals k; and $\vec{s}^{\,\ell}_r[i] = \vec{w}^{\ell}[i]$ holds, for each partial cluster type ℓ with $\mathbb{P}_r[\ell] > 0$, and for all i.

Consistency Constraints. Consider a node v and its 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 some valid partial cluster types at v, v_1, v_2, respectively. We say P_v is consistent with P_{v_1} and P_{v_2} if the following conditions are met:

- **Type Consistency**. The types of P_v, P_{v_1}, and P_{v_2} must be the same, i.e. $t_c = t_{c_1} = t_{c_2}$.
- **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 δ^{in}_v be the inner groups adjacent to γ_v in the backbone tree; $\delta^{in}_v = \delta(\{\gamma_v\}) \cap \Gamma_v$, where $\delta(\{\gamma_v\})$ indicates groups adjacent 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 $\gamma_v = \gamma_{v_1} = \gamma_{v_2}$ (Figure 4a), then $\delta^{in}_{v_1} \cup \delta^{in}_{v_2} = \delta^{in}_v, \quad \delta^{in}_{v_1} \cap \delta^{in}_{v_2} = \emptyset$.
  - If $\gamma_v = \gamma_{v_1}$ and $\gamma_{v_2} \in \delta^{in}_v$ (Figure 4b), then $\delta^{in}_{v_1} = \delta^{in}_v \setminus \{\gamma_{v_2}\}, \quad \delta^{in}_{v_2} = \delta(\{\gamma_{v_2}\}) \setminus \{\gamma_v\}$.
  - If $\gamma_v = \gamma_{v_2}$ and $\gamma_{v_1} \in \delta^{in}_v$ (Figure 4c), then $\delta^{in}_{v_2} = \delta^{in}_v \setminus \{\gamma_{v_1}\}, \quad \delta^{in}_{v_1} = \delta(\{\gamma_{v_1}\}) \setminus \{\gamma_v\}$.
- **Size Consistency**. The group sizes of P_1 and P_2 are consistent with those of P. 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 $\gamma_{v_2} \in \delta^{in}_v$, 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 $\gamma_{v_1} \in \delta^{in}_v$, then we ensure that $\vec{s}_{v_1}[\gamma_{v_1}] = w[\gamma_{v_1}]$ and $\vec{s}_{v_2}[\gamma_{v_2}] = \vec{s}_v[\gamma_v]$.

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 the preprocessing step).

The value of $\lambda[\mathbb{P}_v, \mathbb{P}_{v_1}, \mathbb{P}_{v_2}]$ is calculated recursively for every combination of configurations of v and its children, v_1, v_2. For the base case $\lambda[\vec{0}, \vec{0}, \vec{0}] = \text{True}$. Let $\mathbb{P}_v - P_v$ indicate the configuration of $\mathbb{P}_v$ with one less partial cluster of type P_v. For the recurrence, we consider all possible *consistent* valid partial cluster types P_v, P_{v_1} and P_{v_2}

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

Analysis

In our DP, configurations store the rounded sizes (and weights) of the partial clusters' groups. To ensure consistency between the sizes of the groups at node v and its children v_1 and v_2, we allow the size of the group at v to be a $(1 + \epsilon')$ upper bound for the combined size of the groups at v_1 and v_2. This results in a multiplicative error of at most $(1 + \epsilon')$ in the calculation of the edges' loads and so the cost of the partial clusters at each node of the tree when the sizes (weights) of merged partial clusters are rounded. Given that the height of the tree is h, it is not difficult to see that our dynamic programming approach finds a solution that is an $(1 + \epsilon')^h$-approximation to the problem.

The number of possible configurations $\mathbb{P}_v$ for each node v is at most $n^{O(\sigma^{\sigma-2}(\frac{\log n}{\epsilon'})^{\sigma}))}$, resulting in $n^{O(\sigma^{\sigma-2}(\frac{\log n}{\epsilon'})^{\sigma}))}$ dynamic program table entries. To compute each entry in the DP table, we iterate over all consistent configurations at v, v_1, and v_2, which takes $n^{O(\sigma^{\sigma-2}(\frac{\log n}{\epsilon'})^{\sigma}))}$ time. Hence, the overall running time of the algorithm is $n^{O(\sigma^{\sigma-2}(\frac{\log n}{\epsilon'})^{\sigma}))}$, which is still a quasi-polynomial time complexity in n. By setting $\epsilon' = \frac{\epsilon}{\log n}$ in the threshold mapping, the algorithm finds a $(1 + \epsilon)$ approximation solution in time $n^{O(\sigma^{\sigma-2}(\frac{\log n}{\epsilon})^{\sigma+1}))}$.

▶ **Theorem 9.** *There is a QPTAS for the k-**MSC** problem on trees with logarithmic heights.*

3 The k-MSC Problem in Metrics of Bounded Treewidth

In this section, we extend our algorithm from Section 2 to metrics of bounded treewidth. A *tree decomposition* of a graph $G = (V, E)$ 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: (i) $\cup_{i \in V'} b_i = V$, (ii) for every edge $uv \in E$, there exists a bag t of T such that b_t contains both u and v, (iii) if b_i, b_j contain vertex v then every bag on the path between i and j in T contains v. 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. The authors of [5] 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: (i) T is binary, (ii) the height of T is $O(\log |V|)$.

Given a graph $G = (V, E)$ with a treewidth of f', we create a binary decomposition tree $T = (V', E')$ with a width of no more than $3f' + 2$ and a height of logarithmic in $|V|$ (see [5]). Let f be the width of T. 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 will refer to edges in G as *edges* and edges in T as *super-edges*. Let T be rooted at an arbitrary bag $r \in V'$. 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. Each node $u \in V$ can appear in multiple bags of V', and these bags form a subtree of T. To ensure that each point is covered only once, we consider the point as a *token* placed at the node. We place the token of a node at the bag closest to the root of T that contains the node. This bag is marked as the one containing the point.

We further modify the tree to make sure that (i) only the leaf bags contain the tokens and (ii) each bag contains at most one token: for any bag A that violates these two rules, create two new bags B and C that are identical copies of A. Move one of the tokens from the original bag A to bag C and place any remaining tokens in bag B. Connect the children of the original bag A to the newly created bag B. Connect both bags B and C to A. Finally, we remove all leaf bags without any tokens. This process results in a binary tree decomposition with a height of $O(\log n)$. We call this tree decomposition with these properties the **proper tree decomposition** of the graph. For each point $u \in V$, we let $B_u \in V'$ denote the bag that contains point u. For each $C \subseteq V$, let $B_C = \{B_u : u \in C\}$.

Consider a mapping $p : V' \to V'$ that maps each bag to its parent bag and maps r to itself. Let e_b be the super-edge between b and $p(b)$ in T. The edges (s,t) where $s \in b$ and $t \in p(b)$ are referred to as the **bridge-edges** with respect to the super-edge e_b. We use the notation $e_b^{s,t}$ to refer to these edges. An edge between such vertices $s \in b$ and $t \in p(b)$ is added in G with a weight of $d(s,t)$ if it does not already exist. For any pair of points u and v in V, one can verify that there exists a path between $u \in B_u$ and $v \in B_v$ in the tree T consisting only of bridge-edges over the super-edges which is equivalent to the shortest path between u and v in the graph. This path connects the bags B_u and B_v in T and only uses the bridge-edges over the super-edges of the unique path connecting these bags in the tree. The length of this path is equal to $d(u,v)$, the distance between u and v in the graph G. This path is referred to as $p_B(u,v)$.

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 $V'(T_b)$. 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 $\hat{V}$. The proof of the following lemma is analogous to that of Lemma 5.

▶ **Lemma 10.** *Given a graph $G = (V,E)$ of bounded treewidth, a proper tree decomposition $T = (V', E')$ of G, a set of points $C \subseteq V$, for any $\nu > 0$, there exists a partition of V' into a set of groups $\mathbb{C}_\nu = \{g_1, \ldots, g_\sigma\}$ such that all of the following properties hold: (i) The subgraph induced by each group $g \in \mathbb{C}_\nu$ is connected in T. (ii) For each group $g \in \mathbb{C}_\nu$, $|g \cap B_C| \in [1, \max\{1, \nu|C|\}]$. (iii) $\sigma = O(1/\nu)$. (iv) $\forall g \in \mathbb{C}_\nu, \quad |\delta(g)| = O(1/\nu)$.*

Let $\nu > 0$. Consider a cluster $C \subseteq V$. Let $\mathbb{C}_\nu = \{g_1, \ldots, g_\sigma\}$ be the groups obtained by Lemma 10. For each such cluster C and any constant $\nu > 0$, we let $H_\nu(C) = \cup_{i=1}^{\sigma} \cup_{j \in \delta(g_i)} b_j$ denote the *border hubs* of the cluster and $\text{cost}_{H_\nu}(C) = \sum_{i=1}^{\sigma} \sum_{j=i+1}^{\sigma} \sum_{u \in V(g_i) \cap C, v \in V(g_j) \cap C} d_{H_\nu(C)}(u,v)$ be the *ν-approximate cost* of the cluster. Notice that for any two points u and v in C that belong to different groups of $\mathbb{C}_\nu$, the path $p_B(u,v)$ passes through the hubs $H_\nu(C)$, implying $d(u,v) = d_{H_\nu(C)}(u,v)$. The proof of the following is analogous to that of Theorem 7.

▶ **Theorem 11.** *Given $\epsilon > 0$, a $(1+\epsilon)$-approximation for k-**MHC**, will imply a $(1+O(\epsilon))$-approximation for k-**MSC** on bounded treewidth graphs.*

3.1 QPTAS for k-MHC on Graphs of Bounded Treewidth

Given $\nu > 0$ and a graph $G(V,E)$ that has a *proper decomposition tree* $T = (V', E')$ with a logarithmic height and a treewidth of f. Let OPT be the minimum cost of partitioning V into k clusters $C_1, C_2, \cdots, C_k$ with the total cost being $\sum_{i=1}^{k} \text{cost}_{H_\nu}(C_i)$. Given $\epsilon > 0$, we will present a dynamic program that finds a $(1+\epsilon)$ approximation of OPT. This, as a result of Theorem 11, leads to a $(1+O(\epsilon))$ approximation solution for the k-**MSC** problem.

Consider a cluster $C \subseteq V$. Let $\mathbb{C}_\nu = \{g_1, \dots, g_\sigma\}$ be the groups obtained by Lemma 10 on C. We define a *backbone tree* associated with the cluster C. This tree is made up of $O(1/\nu)$ nodes that correspond to the groups of $\mathbb{C}_\nu$ and there are edges between the nodes in the tree if the corresponding groups in $\mathbb{C}_\nu$ are connected by a super-edge in the tree T. A *cluster type* is defined as a node-weighted backbone tree where each node in the tree is assigned a weight from the threshold values $\Phi_{\epsilon,n}$ (see Definition 8) which represents the number of points in the corresponding group rounded up to the nearest threshold value.

For each cluster C and bag b in tree T, we associate a partial cluster type to it. This is represented by a triple $(t_c, \gamma_b, \vec{s}_b)$ and includes: the type of the cluster, t_c; the group of the cluster that has bag b, γ_b; and a vector $\vec{s}_b$, where $\vec{s}_b[i]$ denotes the number of points in the ith group located in tree T_b. It is not hard to verify that the number of possible partial clusters is $O(\sigma^{\sigma-2} \log^{\sigma}_{(1+\epsilon')} n) = O((\frac{\log n}{\epsilon})^{\sigma+1})$, where we fix $\sigma = O(1/\nu)$.

We use $\ell \in \{1, 2, \dots, O((\frac{\log n}{\epsilon})^{\sigma+1})\}$ to refer to a specific partial cluster type. A partial cluster type ℓ with respect to a vertex b is considered **valid** if: the values of $\vec{s}^{\ell}_v[i]$ for each group i of ℓ are between 0 and $\vec{w}^{\ell}[i]$, the value of $\vec{w}^{\ell}[i]$ for each group i of ℓ is less than or equal to $\max\{\nu. \sum_{i'} \vec{w}^{\ell}[i'], 1\}$, and if v is a leaf vertex of T, then γ^{ℓ}_v is a leaf node of the backbone tree of ℓ. A partial cluster type ℓ is considered a **leaf partial cluster type** at a vertex b if γ^{ℓ}_b is a leaf node of the backbone tree of ℓ and $\vec{s}^{\ell}_b[\gamma^{\ell}_v] = 1$.

Consider a cluster C together with its groups $\mathbb{C}_\nu = \{g_1, \dots, g_\sigma\}$ and hubs $H_\nu(C)$, and let ℓ be the type of this cluster with respect to bag b. Here, we explain how to compute the ν-approximate cost of the cluster, $\text{cost}_{H_\nu}(C)$. Let $X = \cup_{i=1}^{\sigma} X_i$ and $X_i = \{(u, v) : u \in V(g_i) \;\; v \in C \setminus V(g_i)\}$. Let e_b denote the super edge connecting b to its parent bag $p(b)$ in T. We define load of a bridge-edge $e^{s,t}_b$ with respect to the cluster C, its groups $\mathbb{C}_\nu$, and hubs $H_\nu(C)$ to be the number of paths $p_B(u, v)$ that contain this edge over all $\{u, v\} \in X$. We use $load^{\ell}(e^{s,t}_b)$ to represent the load of bridge-edge $e^{s,t}_b$ with respect to partial cluster type ℓ and bag b. Similarly, we use $load^{\ell}(e_b)$ to represent the load of super-edge e_b with respect to partial cluster type ℓ and bag b.

Similarly to the case of the tree, the load of the super-edge e_b with respect to ℓ can be calculated using the following formula: $load^{\ell}(e_b) = \big(\sum_{i=1, i\neq\gamma_b}^{\sigma} \vec{s}^{\ell}_b[i]\big) \times \big(\sum_{i=1}^{\sigma} (\vec{w}^{\ell}[i] - \vec{s}^{\ell}_b[i])\big) + \vec{s}^{\ell}_b[\gamma_b] \times \big(\sum_{i \notin \Gamma_b}^{\sigma} \vec{w}^{\ell}[i]\big)$. Note that $load^{\ell}(e_b)$ computes the number of paths $p_{H_\nu(C)}(u, v)$ in G that cross the cut-set $(b, p(b))$ for all pairs of points (u, v) in the set X.

When computing the cost of a cluster type, it is necessary to take into account the load among the bridge-edges. However, the load of a bridge-edge cannot be calculated simply from the sizes and weights of the groups within the cluster, unlike the load of the super-edges.

To address this issue, for each partial cluster type ℓ and each b, we have defined a vector ψ^{ℓ}_b with a dimension of f^2 (where f is the treewidth of the graph), that $\psi^{\ell}_b[e^{s,t}_b]$ specifies the load of each bridge-edge $e^{s,t}_b$ with respect to ℓ. One can now compute the cost of a partial cluster ℓ at bag b, denoted 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.

We could attach ψ^{ℓ}_b (with a dimension of f^2 which approximately stores the flow of the bridge edges) to the vectores we store for each cluster type ℓ to obtain a QPTAS for the problem on graphs with bounded treewidth. However, this QPTAS cannot be extended to include graphs with bounded highway dimension or graphs with bounded doubling dimensions (as f becomes logarithmic in these cases). To address this issue, in the next section we propose that at each bag v, it is sufficient to store information about the total flow of the partial clusters that passes through the bridge edges, in addition to the information about the type of partial cluster covering the points within the subtree. This eliminates the need to separately store the flow of each partial cluster.

Dynamic program

The Dynamic Program (DP) traverses T starting at the leaves and moving upward and considers all ways partial clusters can be made. At each bag b, a configuration $< b, \mathbb{P}_b, \psi_b >$ is defined. In this configuration, $\mathbb{P}_b$ specifies the number of partial clusters 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, $\psi_b = \sum_\ell \mathbb{P}_b[\ell].\psi_b^\ell$.

Valid Configuration. The validity check of a configuration involves ensuring the feasibility of the load distributions among partial clusters. For a given bag b and configuration $(\mathbb{P}_b, \psi_b)$, we can use the loads of super edges to get the total loads crossing b: $\Psi_b = \sum_\ell \mathbb{P}_b[\ell] load^\ell(e_b)$. We say the configuration $(\mathbb{P}_b, \psi_b)$ is *valid* if the following holds: $\phi(\Psi_b) = \phi\left(\sum_{e^b_{s,t} \in b \times p(b)} \psi[e^b_{s,t}]\right)$; this is, the total load of the partial clusters crossing super-edge e_b (this can be obtained via $\mathbb{P}_b$ as described in the previous section) must be equal to the total load of the partial clusters crossing all the bridge-edges with respect to the super-edge e_b. Note that when b is a leaf, this condition implies that, $\phi(\sum_{e^b_{s,t} \in b \times p(b)} \psi[e^b_{s,t}]) = \phi(\sum_i w[i] - 1)$.

Assume for now 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. The representation of $\perp$ is used to indicate the empty configurations for handling the cases when b is a leaf or has one child.

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 $V'_b = \cup_{i \in T_b} b_i$).

We will compute the subproblems $A[b, \mathbb{P}_b, \psi_b]$ in a bottom-up manner:

Base Case. For each leaf vertex b: $A[b, \mathbb{P}_b, \psi_b] = 0$ if $\varphi[(\mathbb{P}_b, \psi_b), \perp, \perp] =$ True and otherwise it is ∞.

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

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

The case of b having one child is similar. The final solution is obtained by finding the minimum value of $A[b, \mathbb{P}_b, \psi_b]$ over all valid configurations $< \mathbb{P}_b, \psi_b >$ such that the sum of all $\mathbb{P}_b[\ell]$ values equals k.

Consistency Constraints

Consider a bag b and its two children b_1 and b_2. Let $< \mathbb{P}_b, \psi_b >, < \mathbb{P}_{b_1}, \psi_{b_1} >, and < \mathbb{P}_{b_2}, \psi_{b_2} >$ be some configurations at b, b_1, and b_2, respectively. To check the consistency of them, there are two steps to follow: (1) verify the *feasibility of partial cluster types*; if the types of the partial clusters in $\mathbb{P}_b$ match those in $\mathbb{P}_{b_1}$ and $\mathbb{P}_{b_2}$. (2) ensure the *feasibility of load distributions*; if the load distribution of the clusters in ψ_b aligns with the load distributions of the clusters in ψ_{b_1} and ψ_{b_2}. If these two conditions are met, $\varphi[(\mathbb{P}_b, \psi_b), (\mathbb{P}_{b_1}, \psi_{b_1}), (\mathbb{P}_{b_2}, \psi_{b_2})]$ will be set to True. Otherwise, it will be set to False.

Feasibility of Partial Cluster Types. Here we check if there is a solution where the descriptions of partial clusters below nodes b, b_1, and b_2 match the configurations $\mathbb{P}_b$, $\mathbb{P}_{b_1}$, and $\mathbb{P}_{b_2}$, respectively. This step guarantees that the final clustering covers all the points and is therefore a valid solution. This check is very similar to the consistency verification we performed in the case of the tree. There are three cases, depending on whether b is a *leaf*, a bag with *one child*, or a bag with *two children*:

- when b is a leaf: $\mathbb{P}_b[\ell] = 1$ must hold for some ℓ, where ℓ is a leaf partial cluster at b.
- when b has one child, say bag b_1: since there is no point (token) on internal bags, b and b_1 must belong to the same group. In this case, we must ensure the following: $t_b = t_{b_1}$ (*type consistency*); $\gamma_b = \gamma_{b_1}, \delta_b^{in} = \delta_{b_1}^{in}$ (*group consistency*); and $\vec{s}_b = \vec{s}_{b_1}$ (*size consistency*).
- when 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: (i) (*type consistency*) $t_c = t_{c_1} = t_{c_2}$. (ii)(*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 $\delta_{b_1}^{in} = \delta(\{\gamma_{b_1}\}) \setminus \{\gamma_b\}$. (iii) (*size consistency*) If $\gamma_b = \gamma_{b_1} = \gamma_{b_2}$, then we ensure that $\phi(\vec{s}_{b_1}[\gamma_{b_1}] + \vec{s}_{b_2}[\gamma_{b_2}]) = \vec{s}_b[\gamma_b]$. 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 the recurrence, we consider all possible *consistent* partial cluster types P_b, P_{b_1} and P_{b_2}

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

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. This ensures that the sum of all flows through the bridge edges into bag b and the sum of all flows out of it are consistent, and that the flow originates only from points that have tokens. This confirms the accuracy of the solution cost calculated using these bridge-edge load distributions. There are three cases, depending on whether b is a leaf, a bag with one child, or a bag with two children:

- when b is a leaf. Suppose $y \in b$ is the only point of bag b, we must ensure that: $\forall st : s \in b, t \in p(b), s \neq y, \psi[e_{s,t}^b] = 0$
- when b has one child, say 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: $\forall t \in b.\quad \sum_{s \in b_1} \psi[e_{s,t}^{b_1}] = \sum_{u \in p(b)} \psi[e_{t,u}^b]$
- when b has two children, b_1, b_2. For each $t \in b$ let L_t be $\sum_{s \in b_1} \psi[e_{s,t}^{b_1}]$, R_t be $\sum_{s \in b_1} \psi[e_{s,t}^{b_2}]$, U_t be $\sum_{s \in p(b)} \psi[e_{t,s}^b]$. 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 following constraints must hold: $L_b + R_b = U_b$ or $|L_b - R_b| = U_b$.

Proof of Theorem 1. There are $O((\frac{\log n}{\epsilon})^{\sigma+1})$ possible partial clusters, so the number of subproblem configurations, $\mathbb{P}_b$, at bag b is $n^{O((\frac{\log n}{\epsilon})^{\sigma+1})}$. The number of the possible values for ψ, is n^{f^2}, resulting in a number of DP table entries of $n^{O(f^2 + (\frac{\log n}{\epsilon})^{\sigma+1})}$.

Deciding 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 $n^{O(f^2 + (\frac{\log n}{\epsilon})^{\sigma+1})}$. Therefore the running time is $n^{O(f^2+(\frac{\log n}{\epsilon})^{\sigma+1})}$, which is quasi-polynomial in n. Notice that even if treewidth is poly-logarithmic, the running time stays quasi-polynomial.

We lose a factor of $(1+\epsilon/\log n)$ when computing $A[b, \mathbb{P}_b]$ at each level of recursion. Since the height of the tree is at most $c \log n$, the approximation factor of the solution is $1+\epsilon$. ◀

4 Bounded Doubling, Highway Dimension, and Minor-Free Metrics

We assume that the aspect ratio of a given metric in a k-MSC instance is polynomially bounded (the details are omitted). We use our QPTAS for k-MSC on graphs with bounded treewidth as a black box and combine it with embeddings into polylogarithmic-treewidth graphs [7, 10, 14] to develop QPTASs for k-MSC on metric spaces with bounded doubling dimension[2], bounded highway dimension, and minor-free metrics. The details are omitted in this version of the paper.

References

1. Martin Aigner and Günter M. Ziegler. Cayley's formula for the number of trees. *Proofs from THE BOOK*, pages 201–206, 2010. doi:10.1007/978-3-642-00856-6_30.
2. Sandip Banerjee, Rafail Ostrovsky, and Yuval Rabani. Min-Sum Clustering (With Outliers). In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, volume 207 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:16, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2021/14709, doi:10.4230/LIPIcs.APPROX/RANDOM.2021.16.
3. Yair Bartal, Moses Charikar, and Danny Raz. Approximating min-sum k-clustering in metric spaces. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 11–20, 2001.
4. Babak Behsaz, Zachary Friggstad, Mohammad R Salavatipour, and Rohit Sivakumar. Approximation algorithms for min-sum k-clustering and balanced k-median. *Algorithmica*, 81:1006–1030, 2019.
5. Hans L. Bodlaender and Torben Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. *SIAM Journal on Computing*, 27(6):1725–1746, 1998. arXiv:https://doi.org/10.1137/S0097539795289859, doi:10.1137/S0097539795289859.
6. Vincent Cohen-Addad, CS Karthik, and Euiwoong Lee. On approximability of clustering problems without candidate centers. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2635–2648. SIAM, 2021.
7. Vincent Cohen-Addad, Hung Le, Marcin Pilipczuk, and Michał Pilipczuk. Planar and minor-free metrics embed into metrics of polylogarithmic treewidth with expected multiplicative distortion arbitrarily close to 1. *arXiv preprint arXiv:2304.07268*, 2023. URL: https://arxiv.org/abs/2304.07268.
8. Artur Czumaj and Christian 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*, STACS'07, pages 536–548, Berlin, Heidelberg, 2007. Springer-Verlag.

[2] To obtain a QPTAS for Euclidean Min-Sum clustering, we could adopt the approach we proposed for tree-like metrics. This involves using a $(1+\epsilon)$-reduction from Euclidean Min-Sum clustering to Euclidean Min-Hub clustering, achieved by placing hubs of constant size in suitable locations for each cluster. We could then apply Arora's scheme to get a QPTAS for Euclidean Min-Hub clustering, with cluster types determined by the backbone structure of the hubs. We skip the details since this can be derived from the reduction from doubling metrics to bounded treewidth metrics.

9 Wenceslas Fernandez de la Vega, Marek Karpinski, Claire Mathieu, and Yuval Rabani. Approximation schemes for clustering problems. In *STOC '03*, 2003.

10 Andreas Emil Feldmann, Wai Shing Fung, Jochen Könemann, and Ian Post. A $(1+\varepsilon)$-embedding of low highway dimension graphs into bounded treewidth graphs. *SIAM Journal on Computing*, 47(4):1667–1704, 2018. doi:10.1137/16M1067196.

11 Nili Guttmann-Beck and Refael Hassin. Approximation algorithms for min-sum p-clustering. *Discrete Applied Mathematics*, 89(1-3):125–142, 1998.

12 Viggo Kann, Sanjeev Khanna, Jens Lagergren, and Alessandro Panconesi. On the hardness of approximating max k-cut and its dual. *Chicago J Theoret Comput Sci*, 05 1997.

13 Sartaj Sahni and Teofilo F. Gonzalez. P-complete approximation problems. *Journal of the ACM (JACM)*, 23:555 – 565, 1976.

14 Kunal Talwar. Bypassing the embedding: Algorithms for low dimensional metrics. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 281–290, New York, NY, USA, 2004. Association for Computing Machinery. doi:10.1145/1007352.1007399.