Nearly-Optimal Hierarchical Clustering for Well-Clustered Graphs

Steinar Laenen [*1] **Bogdan-Adrian Manghiuc** [*1] **He Sun** [*1]

Abstract

This paper presents two efficient hierarchical clustering (HC) algorithms with respect to Dasgupta's cost function. For any input graph G with a clear cluster-structure, our designed algorithms run in nearly-linear time in the input size of G, and return an $O(1)$-approximate HC tree with respect to Dasgupta's cost function. We compare the performance of our algorithm against the previous state-of-the-art on synthetic and real-world datasets and show that our designed algorithm produces comparable or better HC trees with much lower running time.

1. Introduction

Hierarchical clustering (HC) is the recursive partitioning of a dataset into increasingly smaller clusters via an effective binary tree representation, and has been employed as a standard package in data analysis with widespread applications in practice. Traditional HC algorithms are typically based on agglomerative heuristics and, due to the lack of a clear objective function, there was limited work on their analysis. Dasgupta (2016) introduced a simple cost function for hierarchical clustering, and this work has inspired a number of algorithmic studies on hierarchical clustering.

In this paper we study efficient hierarchical clustering for graphs with a clear structure of clusters. We prove that, under two different conditions of an input graph G that characterise its cluster-structure, one can construct in nearly-linear time[1] an $O(1)$-approximate HC tree $\mathcal{T}$ of G with respect to Dasgupta's cost. Our results show that, while it's NP-hard to construct an $O(1)$-approximate HC tree for general graphs assuming the Small Set Expansion Hypothesis (Charikar & Chatziafratis, 2017), an $O(1)$-approximate HC tree can be constructed in nearly-linear time for a wide range of graph instances occurring in practice. This nearly-linear time complexity of our designed algorithms represents a significant improvement over the previous state-of-the-art on the same problem (Manghiuc & Sun, 2021).

Our designed two algorithms share the same framework at the high level: we apply spectral clustering (Ng et al., 2001) to partition an input graph G into k clusters $P_1, \ldots, P_k$, and further partition each cluster by grouping the vertices in every P_i $(1 \leqslant i \leqslant k)$ with respect to their degrees. We call the resulting vertex sets *degree buckets*, and show that the Dasgupta cost of HC trees constructed on degree buckets is low. These intermediate trees constructed on every bucket can therefore form the basis of our final tree. To construct the final HC tree on G, we merge the degree bucket trees based on the following two approaches:

- Our first approach treats every degree bucket of vertices in G as a single vertex of another "contracted" graph H with much fewer vertices. Thanks to the small size of H, we apply the recursive sparsest cut algorithm and construct a tree on H in a top-down fashion. The structure of this tree on H determines how the degree bucket trees are merged when constructing our final tree of G.
- For our second approach, we show that under a regularity assumption on the vertex degrees, it suffices to merge the clusters in a "caterpillar" style. This simpler construction allows our second algorithm to run in nearly-linear time for a larger number of clusters k compared with the first algorithm.

To demonstrate the significance of our work, we experimentally compare our first algorithm against the previous state-of-the-art and a well-known linkage heuristic (`AverageLinkage`) on both synthetic and real-world datasets. Our experimental results show that the trees constructed from our algorithm and `AverageLinkage` achieve similar cost values, which are much lower than the ones constructed from Cohen-Addad et al. (2017) and Manghiuc & Sun (2021). Moreover, our algorithm runs significantly faster than the other three tested algorithms.

[*]Equal contribution [1]School of Informatics, University of Edinburgh, United Kingdom. Correspondence to: Steinar Laenen <steinar.laenen@ed.ac.uk>, Bogdan-Adrian Manghiuc <b.a.manghiuc@sms.ed.ac.uk>, He Sun <h.sun@ed.ac.uk>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]We say that, for an input graph G with n vertices and m edges, an algorithm runs in nearly-linear time if the algorithm's running time is $O(m \cdot \log^c n)$ for some constant c. For simplicity we use $\widetilde{O}(\cdot)$ to hide this $\log^c n$ factor.

2. Related Work

Addressing the lack of an objective function for hierarchical clustering, Dasgupta (2016) introduced a simple cost function to measure the quality of an HC tree, and proved several properties of the cost function. Dasgupta further showed that a recursive sparsest cut algorithm can be applied to construct an $O(\log^{3/2} n)$-approximate HC tree. Charikar & Chatziafratis (2017) improved the analysis of constructing HC trees based on the sparsest cut problem, and proved that an α-approximate algorithm for the sparsest cut problem can be employed to construct an $O(\alpha)$-approximate HC tree. This implies that, by applying the state-of-the-art for the sparsest cut problem (Arora et al., 2009), an $O(\sqrt{\log n})$-approximate HC tree can be constructed in polynomial-time.

It is known that, assuming the Small Set Expansion Hypothesis, it is NP-hard to construct an $O(1)$-approximate HC tree for general graphs (Charikar & Chatziafratis, 2017). Hence, it is natural to examine the conditions of input graphs under which an $O(1)$-approximate HC tree can be constructed in polynomial-time. Cohen-Addad et al. (2017) studied a hierarchical extension of the classical stochastic block model (SBM) and showed that, for graphs randomly generated from this model, there is an SVD projection-based algorithm (McSherry, 2001) that, together with linkage heuristics, constructs a $(1 + o(1))$-approximate HC tree with high probability. Manghiuc & Sun (2021) studied hierarchical clustering for well-clustered graphs and proved that, when there is a cluster-structure of an input graph G, an $O(1)$-approximate HC tree of G can be constructed in polynomial-time; their designed algorithm is based on the graph decomposition algorithm by Oveis Gharan & Trevisan (2014), and has high time complexity.

There are recent studies of hierarchical clustering in different models of computation. For instance, Kapralov et al. (2022) studied the problem of learning the hierarchical cluster structure of graphs in a semi-supervised setting. Their presented algorithm runs in sub-linear time and, under some clusterability conditions of an input graph G with k clusters, their algorithm $O(\sqrt{\log k})$-approximates Dasgupta's cost of an optimal HC tree. This work is incomparable to ours: the objective of their work is to approximate Dasgupta's cost of an HC tree, while the output of our algorithms is a complete HC tree.

Finally, there are studies of hierarchical clustering under different objective functions. Moseley & Wang (2017) studied the dual of Dasgupta's cost function. This objective, and a dissimilarity objective by Cohen-Addad et al. (2019), have received considerable attention (Alon et al., 2020; Charikar et al., 2019; Chatziafratis et al., 2020). It is important to notice that an $O(1)$-approximate HC tree can be constructed efficiently for general graphs under these objectives, suggesting the fundamental difference in the computational complexity of constructing an HC tree under different objectives. This is the main reason for us to entirely focus on the Dasgupta's cost function in this work.

3. Preliminaries

This section lists the background knowledge used in our paper, and is organised as follows: In Section 3.1 we list the basic notation and facts in spectral graph theory. Section 3.2 discusses hierarchical clustering and Dasgupta's cost function. In Section 3.3 we introduce the notion of contracted graphs, and we finish the section with a brief introduction to spectral clustering in Section 3.4.

3.1. Notation

We always assume that $G = (V, E, w)$ is an undirected graph with $|V| = n$ vertices, $|E| = m$ edges, and weight function $w : V \times V \to \mathbb{R}_{\geqslant 0}$. For any edge $e = \{u, v\} \in E$, we write w_e or w_{uv} to express the weight of e. Let $w_{\min}$ and $w_{\max}$ be the minimum and maximum edge weight of G, respectively; we further assume that $w_{\max}/w_{\min} \leqslant c \cdot n^{\gamma}$ for some constants $\gamma > 0$ and c, which are independent of the input size.

For a vertex $u \in V$, we denote its *degree* by $d_u \triangleq \sum_{v \in V} w_{uv}$. We use δ_G, Δ_G, and d_G for the minimum, maximum and average degrees in G respectively, where $d_G \triangleq \sum_{u \in V} d_u/n$. For any $S \subset V$, we use $\delta_G(S), \Delta_G(S)$, and $d_G(S)$ to represent the minimum, maximum, and average degrees of the vertices of S in G.

For any two subsets $S, T \subset V$, we define the *cut value* between S and T by $w(S, T) \triangleq \sum_{e \in E(S,T)} w_e$, where $E(S, T)$ is the set of edges between S and T. For any $G = (V, E, w)$ and set $S \subseteq V$, the *volume* of S is $\mathrm{vol}_G(S) \triangleq \sum_{u \in S} d_u$, and we write $\mathrm{vol}(G)$ when referring to $\mathrm{vol}_G(V)$. Sometimes we drop the subscript G when it is clear from the context. For any non-empty subset $S \subset V$, we define $G[S]$ to be the induced subgraph on S.

For any input graph $G = (V, E, w)$ and any $S \subset V$, let the conductance of S in G be

$$\Phi_G(S) \triangleq \frac{w(S, V \setminus S)}{\mathrm{vol}(S)}.$$

We define the conductance of G by

$$\Phi_G \triangleq \min_{\substack{S \subset V \\ \mathrm{vol}(S) \leqslant \mathrm{vol}(V)/2}} \Phi_G(S).$$

For any non-empty subset $S \subset V$ we refer to $\Phi_G(S)$ as the *outer conductance* of S with respect to G, and $\Phi_{G[S]}$ as the *inner conductance* of S.

Our analysis is based on the spectral properties of graphs, and here we list the basics of spectral graph theory. For

a graph $G = (V, E, w)$, let $\mathbf{D} \in \mathbb{R}^{n\times n}$ be the diagonal matrix defined by $\mathbf{D}_{uu} = d_u$ for all $u \in V$. We denote by $\mathbf{A} \in \mathbb{R}^{n\times n}$ the *adjacency matrix* of G, where $\mathbf{A}_{uv} = w_{uv}$ for all $u, v \in V$. The *normalised Laplacian matrix* of G is defined as $\mathcal{L} \triangleq \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{I}$ is the $n \times n$ identity matrix. The normalised Laplacian $\mathcal{L}$ is symmetric and real-valued, and has n real eigenvalues which we write as $\lambda_1 \leqslant \ldots \leqslant \lambda_n$. It is known that $\lambda_1 = 0$ and $\lambda_n \leqslant 2$ (Chung, 1997).

For any integer $k \geqslant 2$, we call subsets of vertices $A_1, \ldots, A_k$ a k-way partition of G if $\bigcup_{i=1}^{k} A_i = V$ and $A_i \cap A_j = \emptyset$ for different i and j. We define the *k-way expansion* of G by

$$\rho(k) \triangleq \min_{\text{partitions } A_1,\ldots,A_k} \max_{1\leqslant i\leqslant k} \Phi_G(A_i).$$

The celebrated higher-order Cheeger inequality (Lee et al., 2014) states that it holds for any graph G and $k \geqslant 2$ that

$$\frac{\lambda_k}{2} \leqslant \rho(k) \leqslant O(k^3)\sqrt{\lambda_k}. \tag{1}$$

3.2. Hierarchical Clustering

A *hierarchical clustering (HC) tree* of a given graph G is a binary tree $\mathcal{T}$ with n leaf nodes such that each leaf corresponds to exactly one vertex $v \in V(G)$. Let $\mathcal{T}$ be an HC tree of a graph $G = (V, E, w)$, and $N \in \mathcal{T}$ be an arbitrary internal node[2] of $\mathcal{T}$. We denote $\mathcal{T}[N]$ to be the subtree of $\mathcal{T}$ rooted at N, $\mathsf{leaves}(\mathcal{T}[N])$ to be the set of leaf nodes of $\mathcal{T}[N]$, and $\mathrm{parent}_{\mathcal{T}}(N)$ to be the parent of node N in $\mathcal{T}$. In addition, each internal node $N \in \mathcal{T}$ induces a unique vertex set $C \subseteq V$ formed by the vertices corresponding to $\mathsf{leaves}(\mathcal{T}[N])$. For ease of presentation, we sometimes abuse the notation and write $N \in \mathcal{T}$ for both the internal node of $\mathcal{T}$ and the corresponding subset of vertices in V.

To measure the quality of an HC tree $\mathcal{T}$ with similarity weights, Dasgupta (2016) introduced the cost function defined by

$$\mathsf{COST}_G(\mathcal{T}) \triangleq \sum_{e=\{u,v\}\in E} w_e \cdot |\mathsf{leaves}(\mathcal{T}[u \vee v])|,$$

where $u \vee v$ is the lowest common ancestor of u and v in $\mathcal{T}$. Sometimes, it is convenient to consider the cost of an edge $e = \{u, v\} \in E$ in $\mathcal{T}$ as $\mathsf{cost}_G(e) \triangleq w_e \cdot |\mathsf{leaves}(\mathcal{T}[u \vee v])|$. Trees that achieve a better hierarchical clustering have a lower cost, and the objective of HC is to construct trees with the lowest cost based on the following consideration: for any pair of vertices $u, v \in V$ that corresponds to an edge of high weight w_{uv} (i.e., u and v are highly similar), a "good" HC tree would separate u and v lower in the tree, thus reflected in a small size of $|\mathsf{leaves}(\mathcal{T}[u \vee v])|$. We denote by OPT_G the minimum cost of any HC tree of G, i.e., $\mathsf{OPT}_G = \min_{\mathcal{T}} \mathsf{COST}_G(\mathcal{T})$, and use the notation $\mathcal{T}^*$ to refer to an *optimal* tree achieving the minimum cost. We say that an HC tree $\mathcal{T}$ is an *α-approximate* tree if $\mathsf{COST}_G(\mathcal{T}) \leqslant \alpha \cdot \mathsf{OPT}_G$ for some $\alpha \geqslant 1$.

[2] We consider any non-leaf node of $\mathcal{T}$ an *internal node*. We always use the term *node(s)* for the nodes of $\mathcal{T}$ and the term *vertices* for the elements of the vertex set V.

3.3. Contracted Graphs

Our work is based on *contracted* graphs, which were introduced by Kapralov et al. (2022) in the context of hierarchical clustering.

Definition 3.1 (Contracted Graph (Kapralov et al., 2022))**.** *Let $G = (V, E, w)$ be a weighted graph, and $\mathcal{A} = \{A\}_{i=1}^{k}$ be a partition of V. We say that the vertex and edge-weighted graph $H = ([k], \binom{[k]}{2}, W^*, w^*)$ is a contraction of G with respect to $\mathcal{A}$ if for every $i, j \in [k]$ we have that $W^*(i, j) = w(A_i, A_j)$ and for every $i \in [k]$ we have $w^*(i) = |A_i|$. We denote the contraction of G with respect to $\mathcal{A}$ as $G/\mathcal{A}$.*

Note that contracted graphs are *vertex-weighted*, i.e., every vertex $u \in V(H)$ has a corresponding weight. To measure the quality of an HC tree $\mathcal{T}$ on a vertex-weighted graph $H = (V, E, W, w)$, we define the *weighted Dasgupta's cost* of $\mathcal{T}$ on H as

$$\mathsf{WCOST}_H(\mathcal{T}) \triangleq \sum_{e=\{u,v\}\in E} W(u, v) \sum_{z\in\mathsf{leaves}(\mathcal{T}[u\vee v])} w(z).$$

We denote by WOPT_H the minimum cost of any HC tree of H, i.e., $\mathsf{WOPT}_H = \min_{\mathcal{T}} \mathsf{WCOST}_H(\mathcal{T})$.

For any set $S \subset V(H)$ we define the sparsity of the cut $(S, V \setminus S)$ in H as

$$\mathsf{sparsity}_H(S) \triangleq \frac{W(S, V \setminus S)}{w(S) \cdot w(V \setminus S)}, \tag{2}$$

where $w(S) \triangleq \sum_{v\in S} w(v)$. The vertex-weighted sparsest cut of G is the cut with the minimum sparsity.

We call the vertex-weighted variant of the recursive sparsest cut algorithm the WRSC algorithm, which is described as follows: Let $\alpha \geqslant 1$, and $H = (V, E, W, w)$ be a vertex and edge-weighted graph. Let $(S, V \setminus S)$ be a vertex-weighted sparsest cut of H. The WRSC algorithm on H is a recursive algorithm that finds a cut $(T, V \setminus T)$ satisfying $\mathsf{sparsity}_H(T) \leqslant \alpha \cdot \mathsf{sparsity}_H(S)$, and recurs on the induced subgraphs $H[T]$ and $H[V \setminus T]$. Kapralov et al. (2022) showed that the approximation guarantee of this algorithm for constructing HC trees follows from the one for non-vertex-weighted graphs (Charikar & Chatziafratis, 2017), and their result is summarised as follows:

Lemma 3.2 (Kapralov et al. (2022)). *Let $H = (V, E, W, w)$ be a vertex and edge-weighted graph. Then, the* WRSC *algorithm achieves an $O(\alpha)$-approximation for the weighted Dasgupta's cost of H, where α is the approximation ratio of the sparsest cut algorithm used in* WRSC.

We emphasise that we only use the *combinatorial* properties of vertex-weighted graphs. As such we don't consider their Laplacian matrices and the corresponding spectra.

3.4. Spectral Clustering

Another key component used in our analysis is spectral clustering, which is one of the most popular clustering algorithms used in practice (Ng et al., 2001; Spielman & Teng, 1996). For any input graph $G = (V, E, w)$ and $k \in \mathbb{N}$, spectral clustering consists of the following three steps: (1) compute the eigenvectors $f_1 \dots f_k$ of $\mathcal{L}_G$, and embed each $u \in V(G)$ to the point $F(u) \in \mathbb{R}^k$ based on $f_1 \dots f_k$; (2) apply k-means on the embedded points $\{F(u)\}_{u \in V(G)}$; (3) partition V into k clusters $P_1 \dots P_k$ based on the output of k-means.

To analyse the performance of spectral clustering, one can examine the scenario in which there is a large gap between λ_{k+1} and $\rho(k)$. By the higher-order Cheeger inequality (1), we know that a low value of $\rho(k)$ ensures that the vertex set V of G can be partitioned into k subsets (clusters), each of which has conductance upper bounded by $\rho(k)$; on the other hand, a high value of λ_{k+1} implies that any $(k+1)$-way partition of V would introduce some $A \subset V$ with conductance $\Phi_G(A) \geqslant \rho(k+1) \geqslant \lambda_{k+1}/2$.

Based on this observation, a sequence of works (Macgregor & Sun, 2022; Mizutani, 2021; Peng et al., 2017) showed that, assuming the presence of a large gap between λ_{k+1} and $\rho(k)$, spectral clustering returns clusters $P_1, \dots P_k$ of low outer conductance $\Phi_G(P_i)$ for each $1 \leqslant i \leqslant k$. We remark that spectral clustering can be implemented in nearly-linear time (Peng et al., 2017).

4. Hierarchical Clustering for Well-Clustered Graphs: Previous Approach

Our presented new algorithms are based on the work of Manghiuc & Sun (2021) on the same problem, and this section gives a brief overview of their approach. We consider a graph $G = (V, E, w)$ to have k well-defined clusters if $V(G)$ can be partitioned into disjoint subsets $\{A_i\}_{i=1}^k$ such that (i) there's a sparse cut between A_i and $V \setminus A_i$, formulated as $\Phi_G(A_i) \leqslant \Phi_{\text{out}}$ for any $1 \leqslant i \leqslant k$, and (ii) each $G[A_i]$ has high inner conductance $\Phi_{G[A_i]} \geqslant \Phi_{\text{in}}$. Then, for an input graph G with a clear cluster-structure, the algorithm by Manghiuc & Sun (2021), which we call the MS algorithm in the following, constructs an $O(1)$-approximate HC tree of G in polynomial-time. At a very high level, the MS algorithm is based on the following two components:

- They first show that, when an input graph G of n vertices and m edges has high conductance, an $O(1)$-approximate HC tree of G can be constructed based on the degree sequence of G, and the algorithm runs in $O(m + n \log n)$ time. We use $\mathbb{T}_{\deg}$ to express such trees constructed from the degree sequence of G.
- They combine the first result with the algorithm proposed by Oveis Gharan & Trevisan (2014), which decomposes an input graph into a set of clusters $A_1, \dots, A_\ell$, where every A_i has low outer-conductance $\Phi_G(A_i)$ and high inner-conductance $\Phi_{G[A_i]}$ for any $1 \leqslant i \leqslant \ell$.

With these two components, one might intuitively think that an $O(1)$-approximate HC tree of G can be easily constructed by (i) finding clusters $\{A_i\}$ of high conductance, (ii) constructing an $O(1)$-approximate HC tree $\mathbb{T}_i = \mathbb{T}_{\deg}(G[A_i])$ for every $G[A_i]$, and (iii) merging the constructed $\{\mathbb{T}_i\}$ in an "optimal" way. However, Manghiuc & Sun (2021) show by counterexample that this is not sufficient and, in order to achieve an $O(1)$-approximation, further decomposition of every $\{A_i\}$ would be necessary. Specifically, they adjust the algorithm of Oveis Gharan & Trevisan (2014), and further decompose every vertex set A_i of high-conductance into smaller subsets, which they call *critical nodes*. They show that these critical nodes can be carefully merged to construct an $O(1)$-approximate HC tree for well-clustered graphs. On the downside, as the MS algorithm is heavily based Oveis Gharan & Trevisan (2014), the time complexity of their algorithm is $\widetilde{O}\left(k^3 m^2 n \cdot (w_{\max}/w_{\min})\right)$, which limits the application of their algorithm on large-scale datasets.

5. Algorithms

This section presents our hierarchical clustering algorithms for well-clustered graphs. It consists of two subsections, each of which corresponds to one algorithm.

5.1. The First Algorithm

In this subsection we present a nearly-linear time algorithm that, given a well-clustered graph G with a constant number of clusters as input, constructs an $O(1)$-approximate HC tree of G with respect to Dasgupta's cost. Our result is as follows:

Theorem 5.1. *Given a connected graph $G = (V, E, w)$ and some constant k as input such that $\lambda_{k+1} = \Omega(1)$, $\lambda_k = O(k^{-12})$, and $w_{\max}/w_{\min} = O(n^\gamma)$ for a constant $\gamma > 1$, there is a nearly-linear time algorithm that constructs an HC tree $\mathcal{T}$ of G satisfying $\mathsf{COST}_G(\mathcal{T}) = O(1) \cdot \mathsf{OPT}_G$.*

In comparison to the previous algorithms for hierarchical clustering on well-structured graphs (Cohen-Addad et al., 2017; Manghiuc & Sun, 2021), the advantages of our algorithm are its simplicity and nearly-linear time complexity, which is optimal up to a poly-logarithmic factor.

Overview of the Algorithm. We first describe the algorithm behind Theorem 5.1. Our algorithm consists of four steps. For any input graph $G = (V, E, w)$ and parameter k, our algorithm first runs spectral clustering and obtains clusters $P_1, \ldots, P_k$.

The second step of our algorithm is a degree-based bucketing procedure. We set $\beta \triangleq 2^{k(\gamma+1)}$, and define for any P_i returned from spectral clustering and $u \in P_i$ the set

$$B(u) \triangleq \{v \in P_i : d_u \leqslant d_v < \beta \cdot d_u\}. \tag{3}$$

Moreover, as $B(u)$ consists of all the vertices $v \in P_i$ with $d_u \leqslant d_v < \beta \cdot d_u$, we generalise the definition of $B(u)$ and define for any $j \in \mathbb{Z}$ that

$$B^j(u) = \left\{v \in P_i : \beta^j \cdot d_u \leqslant d_v < \beta^{j+1} \cdot d_u\right\}.$$

By definition, it holds that $B(u) = B^0(u)$, and $\{B^j(u)\}_j$ forms a partition of P_i for any $u \in P_i$. We illustrate the bucketing procedure in Figure 1. In our algorithm, we set $u^{(i)}$ $(1 \leqslant i \leqslant k)$ to be a vertex in P_i with minimum degree and focus on $\left\{B^j\left(u^{(i)}\right)\right\}_j$, the partition of P_i which is only based on non-negative values of j. Let $\mathcal{B}_i \triangleq \left\{B^j\left(u^{(i)}\right)\right\}_j$, and $\mathcal{B} \triangleq \bigcup_{i=1}^{k} \mathcal{B}_i$.

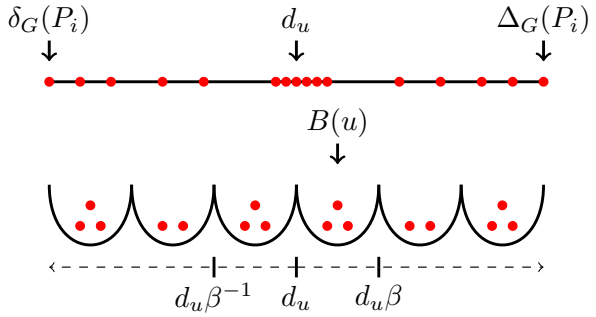

Figure 1. The illustration of our bucketing procedure induced by a vertex u.

In the third step, our algorithm constructs an arbitrary balanced binary tree $\mathcal{T}_B$ for every bucket $B \in \mathcal{B}$, and sets $\mathbb{T}$ to be the collection of our constructed balanced trees $\mathcal{T}_B$.

For the fourth step, our algorithm constructs the contracted graph H defined by $H \triangleq G/\mathcal{B}$, and applies the WRSC algorithm to construct an HC tree $\mathcal{T}_{G/\mathcal{B}}$ of H. Our algorithm finally replaces every leaf node of $\mathcal{T}_{G/\mathcal{B}}$ corresponding to a set $B \in \mathcal{B}$ by an arbitrary balanced tree $\mathcal{T}_B$ on $G[B]$. See Algorithm 1 for the formal description of our algorithm.

Algorithm 1 Spectral Clustering with Degree Bucketing and WRSC (SpecWRSC)

Input: $G = (V, E, w)$, $k \in \mathbb{N}$ such that $\lambda_{k+1} > 0$
$\{P_1, \ldots, P_k\} \leftarrow \mathsf{SpectralClustering}(G, k)$
$\beta \leftarrow 2^{k(\gamma+1)}$
$\mathbb{T} \leftarrow \emptyset$
for every P_i **do**
  Order the vertices of P_i increasingly with respect to their degrees
  $u^{(i)} \leftarrow$ vertex $u \in P_i$ with minimum degree
  $\mathcal{B}_i \leftarrow \left\{B^j\left(u^{(i)}\right)\right\}_j$
  for every $B \in \mathcal{B}_i$ **do**
    Construct an arbitrary balanced $\mathcal{T}_B$ of $G[B]$
    $\mathbb{T} \leftarrow \mathbb{T} \cup \mathcal{T}_B$
  end for
end for
$\mathcal{B} = \bigcup_{i=1}^{k} \mathcal{B}_i$
$H \leftarrow G/\mathcal{B}$
$\mathcal{T} \leftarrow \mathsf{WRSC}(H)$
for every $\mathcal{T}_B \in \mathbb{T}$ **do**
  Replace the leaf node of $\mathcal{T}$ corresponding to B by $\mathcal{T}_B$
end for
Return $\mathcal{T}$

It is important to notice that, as every output P_i $(1 \leqslant i \leqslant k)$ from spectral clustering doesn't necessarily have high inner-conductance, our work shows that in general the inner-conductance condition of vertex sets is not needed in order to construct an $O(1)$-approximate HC tree. This is significant in our point of view, since ensuring the high inner-conductance of certain vertex sets is the main reason behind the high time complexity of the MS algorithm. Moreover, the notion of critical nodes introduced in Manghiuc & Sun (2021), which are subsets of a cluster with high conductance, is replaced by the degree-based bucketing $\mathcal{B}$ of every output cluster P_i from spectral clustering; this $\mathcal{B}$ can be constructed in nearly-linear time.

Proof Sketch of Theorem 5.1. We first analyse the approximation ratio of our constructed tree, and show why these simple four steps suffice to construct an $O(1)$-approximate HC tree for well-clustered graphs. By the algorithm description, our constructed $\mathcal{T}$ is based on merging different $\mathcal{T}_B$, and it holds that

$$\mathsf{COST}_G(\mathcal{T}) = \sum_{i=1}^{k} \sum_{B \in \mathcal{B}_i} \mathsf{COST}_{G[B]}(\mathcal{T}_B) + \sum_{\substack{B, B' \in \mathcal{B} \\ B \neq B'}} \sum_{\substack{e = \{u,v\} \\ u \in B, v \in B'}} \mathsf{cost}_G(e). \tag{4}$$

The first step of our analysis is to upper bound the total contribution of the internal edges of all the buckets, and our result is as follows:

Lemma 5.2. *It holds that*

$$\sum_{i=1}^{k}\sum_{B\in\mathcal{B}_i}\mathsf{COST}_{G[B]}\left(\mathcal{T}_B\right)=O\left(\beta\cdot k^{23}/\lambda_{k+1}^{10}\right)\cdot\mathsf{OPT}_G.$$

By Lemma 5.2, the total cost induced from the edges inside every bucket can be *directly* used when constructing the final tree $\mathcal{T}$. This is crucial for our analysis since our defined buckets can be constructed in nearly-linear time. In comparison to this step, the MS algorithm relies on finding critical nodes, which is computationally much more expensive.

Next, we analyse the second term of (4). For ease of presentation, let $\widetilde{E}$ be the set of edges crossing different buckets, and $\mathcal{T}_{G/\mathcal{B}}$ the tree returned from the WRSC algorithm; remember that every leaf of $\mathcal{T}_{G/\mathcal{B}}$ corresponds to a vertex set in $\mathcal{B}$. By construction, we know that the weight of every edge $e\in\widetilde{E}$ contributes to the edge weight in $G/\mathcal{B}$, and it holds that

$$\sum_{e\in\widetilde{E}}\mathsf{cost}_G(e)=\mathsf{WCOST}_{G/\mathcal{B}}(\mathcal{T}_{G/\mathcal{B}}). \tag{5}$$

Moreover, since $\mathcal{T}_{G\setminus\mathcal{B}}$ is constructed by performing the WRSC algorithm on $G\setminus\mathcal{B}$, we have by Lemma 3.2 that

$$\mathsf{WCOST}_{G/\mathcal{B}}(\mathcal{T}_{G/\mathcal{B}})=O\left(\alpha\right)\cdot\mathsf{WOPT}_{G/\mathcal{B}}, \tag{6}$$

where α is the approximation ratio achieved by the WRSC algorithm.

Our next lemma is the key to the overall analysis, and upper bounds $\mathsf{WOPT}_{G/\mathcal{B}}$ with respect to OPT_G.

Lemma 5.3. *It holds that*

$$\mathsf{WOPT}_{G/\mathcal{B}}=O\left(\beta\cdot k^{23}/\lambda_{k+1}^{10}\right)\cdot\mathsf{OPT}_G.$$

Combining (5), (6) with Lemma 5.3, we have

$$\sum_{e\in\widetilde{E}}\mathsf{cost}_G(e)=O\left(\alpha\cdot\beta\cdot k^{23}/\lambda_{k+1}^{10}\right)\cdot\mathsf{OPT}_G. \tag{7}$$

Next, we study α, which is the only term in the approximation ratio of (7) that is not necessarily a constant. Recall that our choice of γ and β satisfies

$$\frac{w_{\max}}{w_{\min}}=O(n^{\gamma})$$

and $\beta=2^{k(\gamma+1)}$. Hence, it holds that

$$\frac{\Delta_G}{\delta_G}=O\left(n^{\gamma+1}\right),$$

and the total number of buckets in $\mathcal{B}$ is upper bounded by

$$k\cdot\max\{1,\log_\beta n^{\gamma+1}\}\leqslant k+\frac{k(\gamma+1)}{\log\beta}\cdot\log n=k+\log n.$$

Thanks to this, there are at most $k+\log n$ vertices in $G/\mathcal{B}$, and a sparsest cut of $G/\mathcal{B}$ can be found in $O(n)$ time by enumerating all of its possible subsets; as such we can set $\alpha=1$ in the analysis. We highlight that this is another advantage of our bucketing step: with careful choice of parameters, we only need to study a contracted graph with $O(\log n)$ vertices, whose sparsest cut can be found in linear time. Since the WRSC algorithm only computes the vertex-weighted sparsest cut $O(\log n)$ times, the overall running time of WRSC is $O(n\cdot\log n)$. We remark that we don't need to deal with the general sparsest cut problem, for which most approximation algorithms are based on complicated optimisation techniques (e.g., Arora et al. (2009)).

Combining (4) with Lemmas 5.2, 5.3 as well as the fact that $\alpha=1$ proves the approximation guarantee of Theorem 5.1.

Next, we sketch the running time analysis of Algorithm 1. The first step (Line 2) applies spectral clustering, and takes $\widetilde{O}(m)$ time (Peng et al., 2017). The second step (Lines 6–8) is a degree-based bucketing procedure for all the clusters, the time complexity of which is

$$\sum_{i=1}^{k}O\left(|P_i|\log|P_i|\right)=O(n\log n).$$

The third step (Lines 9–12) of the algorithm constructs an arbitrary binary tree $\mathcal{T}_B$ for every bucket $B\in\mathcal{B}$, and we show that this step takes $\widetilde{O}(m)$ time. The actual construction of the trees in this step isn't difficult, but we need to store several attributes for each internal node as we build the tree, such that we can compute Dasgupta's cost in nearly-linear time as well. We refer the reader to the appendix for the detailed discussion of this step. In the last step (Lines 14–19), the algorithm merges the trees $\mathcal{T}_B$ based on the output of WRSC, which can be implemented in $O(n\log n)$ time as discussed above. Combining all these steps gives us the nearly-linear time complexity of Algorithm 1.

Proof Sketch for Lemma 5.2 and Lemma 5.3. Our key approach to breaking the running time barrier of Manghiuc & Sun (2021) is that, instead of approximating the optimal tree $\mathcal{T}^*$, we approximate the tree $\mathcal{T}_{\mathsf{MS}}$ constructed by the algorithm of Manghiuc & Sun (2021). We know from their result that one can upper bound the cost of $\mathcal{T}_{\mathsf{MS}}$ with respect to OPT_G, i.e.,

$$\mathsf{COST}_G(\mathcal{T}_{\mathsf{MS}})=O\left(k^{22}/\lambda_{k+1}^{10}\right)\cdot\mathsf{OPT}_G.$$

We take the existence of the tree $\mathcal{T}_{\mathsf{MS}}$ for granted, and perform two transformations to construct a tree $\mathcal{T}''_{\mathsf{MS}}$, while

ensuring that the total introduced cost from the two transformations can be upper bounded. In the first step, we identify how every bucket $B \in \mathcal{B}$ is spread throughout $\mathcal{T}_{\mathsf{MS}}$, and make sure that all the separate components that make up B are isolated in our new tree. We call the resulting tree $\mathcal{T}'_{\mathsf{MS}}$, and prove that

$$\mathsf{COST}_G\left(\mathcal{T}'_{\mathsf{MS}}\right) \leqslant \mathsf{COST}_G\left(\mathcal{T}_{\mathsf{MS}}\right) + O(k^{21}/\lambda_{k+1}^{10}) \cdot \mathsf{OPT}_G.$$

For the second transformation, we carefully adjust the tree $\mathcal{T}'_{\mathsf{MS}}$, such that the currently isolated components that make up every bucket $B \in \mathcal{B}$ get grouped together into the same subtree. We call the resulting tree $\mathcal{T}''_{\mathsf{MS}}$, and bound its cost by

$$\mathsf{COST}_G\left(\mathcal{T}''_{\mathsf{MS}}\right) \leqslant \mathsf{COST}_G\left(\mathcal{T}'_{\mathsf{MS}}\right) + O(\beta \cdot k^{23}/\lambda_{k+1}^{10}) \cdot \mathsf{OPT}_G.$$

Taking these transformations together, we get that

$$\mathsf{COST}_G\left(\mathcal{T}''_{\mathsf{MS}}\right) \leqslant O(\beta \cdot k^{23}/\lambda_{k+1}^{10}) \cdot \mathsf{OPT}_G. \tag{8}$$

Notice that we can perform these two transformations without an explicit construction of $\mathcal{T}_{\mathsf{MS}}$, but we still end up with a tree $\mathcal{T}''_{\mathsf{MS}}$ with bounded Dasgupta cost. Moreover, since every bucket $B \in \mathcal{B}$ in $\mathcal{T}''_{\mathsf{MS}}$ is separated, $\mathcal{T}''_{\mathsf{MS}}$ is also a tree on the contracted graph $G/\mathcal{B}$; this allows us to upper bound $\mathsf{WOPT}_{G/\mathcal{B}}$ with $\mathsf{COST}_G(\mathcal{T}''_{\mathsf{MS}})$. Combining this with (8) proves Lemma 5.3. With the same approach, we prove that Lemma 5.2 holds as well.

We remark that this is another conceptual contribution of our paper: although Manghiuc & Sun (2021) show that a similar proof technique can be generalised to construct $O(1)$-approximate HC trees for *expander* graphs, we show that such proof technique can be used to obtain $O(1)$-approximate HC trees for *well-clustered* graphs.

5.2. The Second Algorithm

In this subsection, we present another nearly-linear time hierarchical clustering algorithm for well-clustered graphs. Here, we assume that the degrees of vertices inside every cluster are *almost balanced*, i.e., it holds for an optimal partitioning $S_1, \ldots S_k$ corresponding to $\rho(k)$ that the degrees inside each S_i are upper bounded by a parameter $\eta_S \in \mathbb{R}^+$. With this condition, our second algorithm achieves the same approximation guarantee as Algorithm 1 even with $k = O(\log^c n)$ for some constant c. This result is summarised as follows:

Theorem 5.4. *Let $G = (V, E, w)$ be a graph with k clusters $\{S_i\}_{i=1}^k$ such that $\max_i(\Delta_G(S_i)/\delta_G(S_i)) \leqslant \eta_S$, $\lambda_{k+1} = \Omega(1)$, $\rho(k) \leqslant k^{-4}$, and $\Phi_{G[S_i]} \geqslant \Omega(1)$ for $1 \leqslant i \leqslant k$. Then, there is a nearly-linear time algorithm that constructs an HC tree $\mathcal{T}_{\mathrm{SCB}}$ of G satisfying $\mathsf{COST}_G(\mathcal{T}_{\mathrm{SCB}}) = O(1) \cdot \mathsf{OPT}_G$.*

Algorithm 2 Spectral Clustering with Degree Bucketing and Caterpillar Construction

Input: $G = (V, E, w)$, $k \in \mathbb{N}$, $\eta_S \in \mathbb{R}^+$
$P = \{P_1, \ldots, P_k\} \leftarrow \mathsf{SpectralClustering}(G, k)$
$\beta \leftarrow \eta_S$
Initialize $\mathbb{T} \leftarrow \emptyset$
for $P_i \in P$ **do**
  Order the vertices of P_i increasingly with respect to their degrees
  $B(u) \leftarrow \{v \in P_i : d_u \leqslant d_v < \beta \cdot d_u\}$
  $u_i^* \leftarrow \operatorname{argmax}_{u \in P_i} \mathrm{vol}(B(u))$
  $\mathcal{B}_{u_i^*} \leftarrow \left\{B^j\left(u_i^*\right)\right\}_j$
  for $B \in \mathcal{B}_{u_i^*}$ **do**
    Let $\mathcal{T}_B$ be any balanced HC tree on $G[B]$
    $\mathbb{T} \leftarrow \mathbb{T} \cup \mathcal{T}_B$
  end for
end for
Let $\mathbb{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_{|\mathbb{T}|}\}$ be such that $|\mathcal{T}_i| \leqslant |\mathcal{T}_{i+1}|$, for all $1 \leqslant i < |\mathbb{T}|$
Initialize $\mathcal{T}_{\mathrm{SCB}} \leftarrow \mathcal{T}_1$
for $i \leftarrow 2, \ldots, |\mathbb{T}|$ **do**
  $\mathcal{T}_{\mathrm{SCB}} \leftarrow \mathcal{T}_{\mathrm{SCB}} \vee \mathcal{T}_i$
end for
Return $\mathcal{T}_{\mathrm{SCB}}$

The algorithm behind Theorem 5.4 is similar with Algorithm 1, and is described in Algorithm 2. However, our second algorithm has two significant changes compared with Algorithm 1.

1. We adjust the bucketing step by setting the bucketing parameter $\beta = \eta_S$, where η_S is an upper bound for $\max_i(\Delta_G(S_i)/\delta_G(S_i))$. Moreover, instead of bucketing the vertices starting at the vertex $u^{(1)}$ of minimum degree inside P_i, we carefully choose for each P_i the vertex $u_i^* \triangleq \operatorname{argmax}_{u \in P_i} \mathrm{vol}(B(u))$ whose induced bucket $B(u_i^*)$ has the highest volume inside P_i. Based on this, we set $\mathcal{B}_{u_i^*} \triangleq \left\{B^j\left(u_i^*\right)\right\}_j$ and $\mathcal{B} \triangleq \bigcup_{i=1}^k \mathcal{B}_{u_i^*}$.
2. After constructing the balanced trees $\mathcal{T}_B$ for every $B \in \mathcal{B}$, instead of applying WRSC, we concatenate the trees $\mathcal{T}_B$ in a simple "caterpillar" style according to the sizes $|B|$ of the buckets $B \in \mathcal{B}$.

To explain the first change, notice that every cluster P_i returned by spectral clustering has a large overlap with its corresponding optimal cluster S_i. Moreover, the degrees of the vertices inside S_i are within a factor of η_S of each other. Therefore, if an arbitrary vertex $u \in P_i$ is chosen as representative, the bucketing $B(u)$ of P_i might equally divide the vertices in $P_i \cap S_i$ into two consecutive buckets, which is undesirable due to the high induced cost of the crossing edges. To circumvent this issue, we choose u_i^* as

the vertex to induce the bucketing and prove that this specific choice of u_i^* guarantees that the bucketing $\mathcal{B}_{u_i^*}$ contains one bucket $B \in \mathcal{B}_{u_i^*}$ largely overlapping with S_i. This greatly reduces the number of crossing edges in our constructed HC tree and hence improves the approximation guarantee.

To explain the second change, we notice that the nearly-linear running time of WRSC relies on the condition that $k = O(1)$, which might not hold necessarily for the new setting. We prove that, as long as $\max_i (\Delta_G(S_i)/\delta_G(S_i))$ is upper bounded by a constant, it is sufficient to construct a caterpillar tree based on the sizes $|B|$ of the buckets $B \in \mathcal{B}$. We prove that our algorithm runs in nearly-linear time, and the output tree achieves $O(1)$-approximation. See the appendix for the detailed analysis of Algorithm 2 and the proof of Theorem 5.4

6. Experiments

We experimentally evaluate the performance of our designed `SpecWRSC` algorithm, and compare it against the previous state-of-the-art and a well-known linkage algorithm. Specifically, the `SpecWRSC` algorithm is compared against the following three algorithms:

1. the `Linkage++` algorithm proposed in (Cohen-Addad et al., 2017);
2. the `MS` algorithm proposed in (Manghiuc & Sun, 2021);
3. the `AverageLinkage` algorithm.

Even though there are no theoretical approximation guarantees of `AverageLinkage` with respect to Dasgupta's cost function, we include it in our evaluation for reference due to its excellent performance in practice.

All the tested algorithms were implemented in Python 3.9 and experiments were performed using a Lenovo ThinkPad T15G, with an Intel(R) Xeon(R) W-10855M CPU@2.80GHz processor and 126 GB RAM. All of the reported costs and running time below are averaged over 5 independent runs. Our code can be downloaded from https://github.com/steinarlaenen/nearly-optimal-hierarchical-clustering-for-well-clustered-graphs.

6.1. Results on Synthetic Data

We first compare the performance of our algorithm against the others described before on synthetic data.

Stochastic Block Model We look at graphs generated according to the standard Stochastic Block Model (SBM). We set the number of clusters as $k = 5$, and the number of vertices in each cluster $\{P_i\}_{i=1}^5$ as n_k. For each pair of vertices $u \in P_i$ and $v \in P_j$, we add an edge $\{u, v\}$ with probability p if $i = j$, and add an edge with probability q if $i \neq j$.

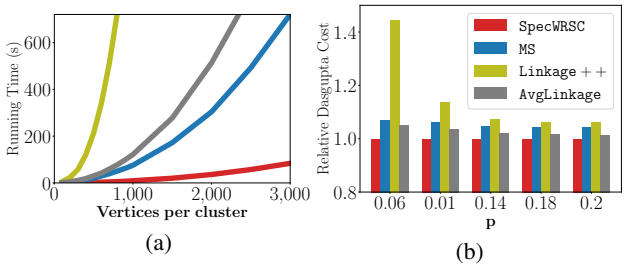

Figure 2. Results for SBMs. In Figure (a) the x-axis represents the number of vertices inside each cluster, and the y-axis represents the algorithms' running time in seconds. In Figure (b), the x-axis represents different values of p, while the y-axis represents the cost of the algorithms' returned HC trees normalised by the cost of `SpecWRSC`.

To compare the running time, we incrementally increase the number of vertices in each cluster $\{P_i\}_{i=1}^5$ from $n_k = 100$ to $n_k = 3{,}000$, in increments of 200, and we fix $p = 0.1$ and $q = 0.002$. For each algorithm, we measure the running time in seconds, and the results are plotted in Figure 2(a), confirming our algorithm's low time complexity compared with our competitors. To further highlight the fast running time of our algorithm, we increase the size of each cluster to 20,000, which results in the total number of nodes $n = 100{,}000$. We find that SpecWRSC returns an HC tree in 14,437 seconds (~ 4 hours), while all the other algorithms timeout after 12 hours of compute time.

To compare the quality of the constructed HC trees on SBMs, we fix the number of vertices inside each cluster $\{P_i\}_{i=1}^5$ as $n_k = 1{,}000$, the value $q = 0.002$, and consider different values of $p \in [0.06, 0.2]$. As shown in Figure 2(b), `SpecWRSC` performs better than all the other algorithms.

Hierarchical Stochastic Block Model. Next, we consider graphs generated from a hierarchical stochastic block model (HSBM) (Cohen-Addad et al., 2019); HSBM is similar to the SBM but assumes the existence of a hierarchical structure between the clusters. We set the number of vertices in each cluster $\{P_i\}_{i=1}^5$ as $n_k = 600$. For each pair of vertices $u \in P_i$ and $v \in P_j$, we assume that u and v are connected by an edge with probability p if $i = j$; otherwise u and v are connected by an edge with probability $q_{i,j}$ defined as follows: (i) for all $i \in \{1, 2, 3\}$ and $j \in \{4, 5\}$, $q_{i,j} = q_{j,i} = q_{\min}$; (ii) for $i \in \{1, 2\}$, $q_{i,3} = q_{3,i} = 2 \cdot q_{\min}$; (iii) $q_{4,5} = q_{5,4} = 2 \cdot q_{\min}$; (iv) $q_{1,2} = q_{2,1} = 3 \cdot q_{\min}$. We fix the value $q_{\min} = 0.0005$ and consider different values of $p \in [0.04, 0.2]$. This choice of hyperparameters bears similarity with the setting tested in (Cohen-Addad et al., 2017; Manghiuc & Sun, 2021). The result of our experiments is plotted in Figure 3(a). Again, we

	Ir	Wi	Ca	Ng
Parameters				
n	50	180	558	3,516
k	3	5	5	5
σ	0.3	0.88	0.88	130
λ_{k+1}/λ_k	5.27	4.96	1.45	1.01
Running Time				
`AvgLinkage`	0.14	0.22	2.49	198.34
`MS`	0.88	2.00	20.36	1,088.87
`Linkage++`	0.61	0.90	9.18	944.05
`SpecWRSC`	**0.136**	**0.16**	**1.69**	**128.17**

Table 1. Setup of the parameters for the tested real-world datasets, and the running time of the tested algorithms. For each dataset, we list the number of nodes n, the number of clusters k, the σ-value used to construct the similarity graph based on the Gaussian kernel, and the spectral gap λ_{k+1}/λ_k of the constructed normalised Laplacian. The running time is reported in seconds, and the lowest one is highlighted.

find that our algorithm outperforms `MS` and `Linkage++`, and has similar performance as `AverageLinkage`.

6.2. Result on Real-World Data

To evaluate the performance of our algorithm on real-world datasets, we first follow the sequence of recent work on hierarchical clustering (Abboud et al., 2019; Cohen-Addad et al., 2017; Manghiuc & Sun, 2021; Menon et al., 2019; Roy & Pokutta, 2017), all of which are based on the following 4 datasets from the Scikit-learn library (Pedregosa et al., 2011) and the UCI ML repository (Dua & Graff, 2017): Iris (Ir), Wine (Wi), Cancer (Ca), and Newsgroup (Ng)[3]. For each dataset, we construct the similarity graph based on the Gaussian kernel, in which the σ-value is chosen according to the standard heuristic (Ng et al., 2001). The setup of parameters for each dataset is listed in Table 1.

The cost of the HC trees returned by each algorithm is reported in Figure 3(b), and the figure shows that our algorithm performs better than `MS` and `Linkage++` and matches the performance of `AverageLinkage`. We further report the running time of the tested algorithms in Table 1, and this shows that `SpecWRSC` has the lowest running time among the four tested algorithms. Moreover, we observe that `SpecWRSC` performs better when the spectral gap λ_{k+1}/λ_k is large, which is in line with our theoretical analysis.

[3]Due to the large size of NewsGroup, we consider only a subset consisting of "comp.graphics", "comp.os.ms-windows.misc", "comp.sys.ibm.pc.hardware", "comp.sys.mac.hardware", "rec.sport.baseball", and "rec.sport.hockey".

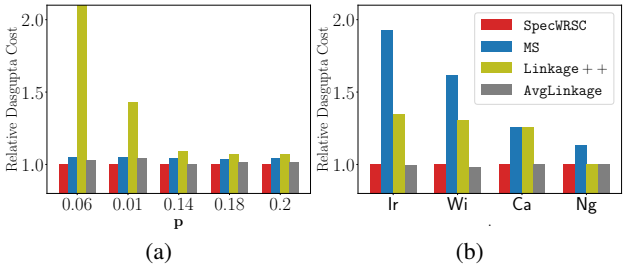

Figure 3. (a) Results for HSBMs, and (b) results on the real-world datasets. In Figure (a) the x-axis represents different values of p, while the y-axis represents the cost of the algorithms' returned HC trees normalised by the cost of `SpecWRSC`. In Figure (b) the x-axis represents the different real-world datasets we evaluate on.

To further demonstrate the effectiveness of our algorithm on larger real-world datasets, we evaluate our algorithm on the GEMSEC Facebook dataset (Rozemberczki et al., 2019). This dataset represents a page-page graph of verified Facebook pages, in which every vertex represents an official Facebook page, and every edge represents the mutual likes between pages. We focus on the largest subset of this dataset, i.e., the artist category. This graph contains 50,515 nodes and 819,306 edges. Out of all the tested algorithms, `SpecWRSC` is the only one that terminates within 12 hours of compute time. Setting $k = 20$, `SpecWRSC` returns a tree in 1,223 seconds, with a Dasgupta cost of $1.62 \cdot 10^{10}$.

These experimental results together demonstrate that our designed algorithm not only has excellent running time but also constructs hierarchical clustering with a cost lower than or similar to the ones constructed by the previous algorithms.

Acknowledgements

This work is supported by an EPSRC Early Career Fellowship (EP/T00729X/1).

References

Abboud, A., Cohen-Addad, V., and Houdrouge, H. Subquadratic high-dimensional hierarchical clustering. In *Advances in Neural Information Processing Systems 33 (NeurIPS'19)*, pp. 11576–11586, 2019.

Alon, N. Eigenvalues and expanders. *Combinatorica*, 6(2): 83–96, 1986.

Alon, N., Azar, Y., and Vainstein, D. Hierarchical clustering: a 0.585 revenue approximation. In *33rd Annual Conference on Learning Theory (COLT'20)*, pp. 153–162, 2020.

Arora, S., Rao, S., and Vazirani, U. Expander flows, geo-

metric embeddings and graph partitioning. *Journal of the ACM*, 56(2):1–37, 2009.

Blum, M., Karp, R. M., Vornberger, O., Papadimitriu, C. H., and Yannakakis, M. The complexity of testing whether a graph is a superconcentrator. *Information Processing Letters*, 13(4-5):164–167, 1981.

Charikar, M. and Chatziafratis, V. Approximate hierarchical clustering via sparsest cut and spreading metrics. In *28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*, pp. 841–854, 2017.

Charikar, M., Chatziafratis, V., and Niazadeh, R. Hierarchical clustering better than average-linkage. In *30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'19)*, pp. 2291–2304, 2019.

Chatziafratis, V., Yaroslavtsev, G., Lee, E., Makarychev, K., Ahmadian, S., Epasto, A., and Mahdian, M. Bisect and conquer: Hierarchical clustering via max-uncut bisection. In *23rd International Conference on Artificial Intelligence and Statistics (AISTATS'20)*, pp. 3121–3132, 2020.

Chung, F. R. K. *Spectral Graph Theory*. American Mathematical Society, 1997.

Cohen-Addad, V., Kanade, V., and Mallmann-Trenn, F. Hierarchical clustering beyond the worst-case. In *Advances in Neural Information Processing Systems 31 (NeurIPS'17)*, pp. 6201–6209, 2017.

Cohen-Addad, V., Kanade, V., Mallmann-Trenn, F., and Mathieu, C. Hierarchical clustering: Objective functions and algorithms. *Journal of the ACM*, 66(4):1–42, 2019.

Dasgupta, S. A cost function for similarity-based hierarchical clustering. In *48th Annual ACM Symposium on Theory of Computing (STOC'16)*, pp. 118–127, 2016.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Kapralov, M., Kumar, A., Lattanzi, S., and Mousavifar, A. Learning hierarchical structure of clusterable graphs. *Arxiv*, 2207.02581, 2022.

Lee, J. R., Oveis Gharan, S., and Trevisan, L. Multiway spectral partitioning and higher-order Cheeger inequalities. *Journal of the ACM*, 61(6):1–30, 2014.

Macgregor, P. and Sun, H. A tighter analysis of spectral clustering, and beyond. In *39th International Conference on Machine Learning (ICML'22)*, pp. 14717–14742, 2022.

Manghiuc, B.-A. and Sun, H. Hierarchical Clustering: O(1)-Approximation for Well-Clustered Graphs. In *Advances in Neural Information Processing Systems 35 (NeurIPS'21)*, pp. 9278–9289, 2021.

McSherry, F. Spectral partitioning of random graphs. In *42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS'01)*, pp. 529–537, 2001.

Menon, A. K., Rajagopalan, A., Sumengen, B., Citovsky, G., Cao, Q., and Kumar, S. Online hierarchical clustering approximations. *Arxiv*, :1909.09667, 2019.

Mizutani, T. Improved analysis of spectral algorithm for clustering. *Optimization Letters*, 15(4):1303–1325, 2021.

Moseley, B. and Wang, J. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. In *Advances in Neural Information Processing Systems 31 (NeurIPS'17)*, pp. 3094–3103, 2017.

Ng, A. Y., Jordan, M. I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 15 (NeurIPS'01)*, pp. 849–856, 2001.

Oveis Gharan, S. and Trevisan, L. Partitioning into expanders. In *25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, pp. 1256–1266, 2014.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Peng, R., Sun, H., and Zanetti, L. Partitioning Well-Clustered Graphs: Spectral Clustering Works! *SIAM Journal on Computing*, 46(2):710–743, 2017.

Roy, A. and Pokutta, S. Hierarchical clustering via spreading metrics. *The Journal of Machine Learning Research*, 18(1):3077–3111, 2017.

Rozemberczki, B., Davies, R., Sarkar, R., and Sutton, C. Gemsec: Graph embedding with self clustering. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2019*, pp. 65–72. ACM, 2019.

Spielman, D. A. and Teng, S.-H. Spectral partitioning works: Planar graphs and finite element meshes. In *37th Annual IEEE Symposium on Foundations of Computer Science (FOCS'96)*, pp. 96–105, 1996.

A. Additional Background

In this section we list additional background notion and facts used in our analysis, and the section is organised as follows: In Section A.1 we introduce additional background on hierarchical clustering. In Section A.2 we provide a more detailed discussion on spectral clustering, and finally in Section A.3 we present and prove a new decomposition lemma of well-clustered graphs.

A.1. Hierarchical Clustering

We first examine the upper and lower bounds of the cost of HC trees. The first fact holds trivially by the fact that every edge's contribution towards the cost function is at most n.

Fact A.1. *It holds for any HC tree $\mathcal{T}$ of G that*

$$\mathsf{COST}_G(\mathcal{T}) \leqslant \frac{n \cdot \mathrm{vol}(G)}{2}.$$

Secondly, it is known that OPT_G can be lower bounded with respect to the degree distribution of G.

Lemma A.1 ((Manghiuc & Sun, 2021))**.** *It holds for any graph G that*

$$\mathsf{OPT}_G \geqslant \frac{2\Phi_G}{9} \cdot \max\left\{\frac{\mathrm{vol}(G)^2}{\Delta_G}, \delta_G \cdot n^2\right\} = \frac{2\Phi_G}{9} \cdot n \cdot \mathrm{vol}(G) \cdot \max\left\{\frac{d_G}{\Delta_G}, \frac{\delta_G}{d_G}\right\}.$$

Next, we introduce the notion of a *dense branch*, which is a useful tool to analyse the cost of HC trees.

Definition A.2 (Dense branch, (Manghiuc & Sun, 2021))**.** *Given a graph G with an HC tree $\mathcal{T}$, the* dense branch *is the path $(A_0, A_1, \ldots, A_k)$ in $\mathcal{T}$ for some $k \in \mathbb{Z}_{\geqslant 0}$, such that the following hold:*

1. *A_0 is the root of $\mathcal{T}$;*
2. *A_k is the node such that $\mathrm{vol}(A_k) > \mathrm{vol}(G)/2$ and both of its children have volume at most $\mathrm{vol}(G)/2$.*

It is important to note that the dense branch of $\mathcal{T}$ is unique, and consists of all the nodes A_i with $\mathrm{vol}(A_i) > \mathrm{vol}(G)/2$. Moreover, for every pair of consecutive nodes A_i, A_{i+1} on the dense branch, A_{i+1} is the child of A_i of the *higher* volume.

The next lemma presents a lower bound on the cost of a tree $\mathcal{T}$ using the dense branch, which will be extensively used in our analysis.

Lemma A.3 (Lower bound of $\mathsf{COST}_G(\mathcal{T})$ based on the dense branch (Manghiuc & Sun, 2021))**.** *Let G be a graph of conductance Φ_G, and $\mathcal{T}$ an arbitrary HC tree of G. Suppose $(A_0, \ldots, A_k)$ is the dense branch of $\mathcal{T}$ for some $k \in \mathbb{Z}_{\geqslant 0}$, and suppose each node A_i has sibling B_i, for all $1 \leqslant i \leqslant k$. Then, the following lower bounds of $\mathsf{COST}_G(\mathcal{T})$ hold:*

1. $\mathsf{COST}_G(\mathcal{T}) \geqslant \frac{\Phi_G}{2} \sum_{i=1}^{k} |A_{i-1}| \cdot \mathrm{vol}(B_i)$;
2. $\mathsf{COST}_G(\mathcal{T}) \geqslant \frac{\Phi_G}{2} \cdot |A_k| \cdot \mathrm{vol}(A_k)$.

One of the two main results from Manghiuc & Sun (2021) is an $O(1)$-approximation on Dasgupta's cost for expander graphs. Their result is stated in the following theorem.

Theorem A.4 (Manghiuc & Sun (2021))**.** *Given any graph $G = (V, E, w)$ with conductance Φ_G as input, there is an algorithm that runs in $O(m + n \log n)$ time, and returns an HC tree $\mathbb{T}_{\deg}(G)$ of G that satisfies $\mathsf{COST}_G(\mathbb{T}_{\deg}) = O\left(1/\Phi_G^4\right) \cdot \mathsf{OPT}_G$.*

Throughout our discussion, we always use $\mathbb{T}_{\deg}(G)$ to represent the HC tree of G that is constructed from Theorem A.4, i.e., the tree constructed based on the degree sequence of G. In particular, for any partition $\{C_1, \ldots, C_\ell\}$, one can apply Theorem A.4 to every induced graph $G[C_i]$ for any $1 \leqslant i \leqslant \ell$, and for simplicity we write $\mathbb{T}_i \triangleq \mathbb{T}_{\deg}(G[C_i])$ in the following discussion. We next define critical nodes with respect to every $\mathbb{T}_i$.

Definition A.5 (Critical nodes (Manghiuc & Sun, 2021)). *Let $\mathbb{T}_i = \mathbb{T}_{\deg}(G[C_i])$ be defined as above for any non-empty subset $C_i \subset V$. Suppose $(A_0, \ldots, A_{r_i})$ is the dense branch of $\mathbb{T}_i$ for some $r_i \in \mathbb{Z}_+$, B_j is the sibling of A_j, and let A_{r_i+1}, B_{r_i+1} be the two children of A_{r_i}. We define $\mathcal{S}_i \triangleq \{B_1, \ldots, B_{r_i+1}, A_{r_i+1}\}$ to be the set of* critical nodes *of C_i. Each node $N \in \mathcal{S}_i$ is a* critical node.

We remark that each critical node $N \in \mathcal{S}_i$ $(1 \leqslant i \leqslant \ell)$ is an internal node of maximum size in $\mathbb{T}_i$ that is not in the dense branch. Moreover, every $\mathcal{S}_i$ is a partition of C_i.

The following lemma lists some facts of the critical nodes of the trees constructed from Theorem A.4.

Lemma A.6 ((Manghiuc & Sun, 2021)). *Let $\mathcal{S}_i = \{B_1, \ldots, B_{r_i+1}, A_{r_i+1}\}$ be the set of critical nodes of $\mathbb{T}_i$. Then the following statements hold:*

(Q1) $|B_j| = 2 \cdot |B_{j+1}|$ *for all* $j \geqslant 2$,

(Q2) $\mathrm{vol}_{G[C_i]}(B_j) \leqslant 2 \cdot \mathrm{vol}_{G[C_i]}(B_{j+1})$ *for all* $j \geqslant 1$,

(Q3) $|A_{i_{\max}}| = |B_{i_{\max}}|$.

For convenience in our notation, we also define a *critical sibling* node of critical nodes.

Definition A.7 (Critical Sibling Node). *Let $\mathcal{S}_i = \{B_1, \ldots, B_{r_i+1}, A_{r_i+1}\}$ be the set of critical nodes of $\mathbb{T}_i$. We define a* sibling node *of a critical node B_j as $\mathrm{sib}_{\mathbb{T}_i}(B_j) \triangleq B_{j+1}$ for $1 \leqslant j \leqslant r_i$, and $\mathrm{sib}_{\mathbb{T}_i}(B_{r_i+1}) \triangleq A_{r_i+1}$ otherwise.*

We remark that the only critical node without a sibling is A_{r_i+1}. Furthermore, due to the degree based construction in Theorem A.4, it holds for all the other critical nodes $N \in \mathcal{S}_i \setminus A_{r_i+1}$ that $\mathrm{vol}(N) \leqslant 2 \cdot \mathrm{vol}(\mathrm{sib}_{\mathbb{T}_i}(N))$ and $|N| \leqslant 2 \cdot |\mathrm{sib}_{\mathbb{T}_i}(N)|$.

Finally, the following inequality will be extensively in our proof of Theorem 5.4:

Lemma A.8. *Let $G = (V, E, w)$ be a graph with a partition $S_1, \ldots, S_k$, such that $\Phi_{\mathrm{out}} \leqslant 1/2$ and $\Phi_{G[S_i]} \geqslant \Phi_{\mathrm{in}}$ for any $1 \leqslant i \leqslant k$. Then it holds that*

$$\sum_{i=1}^{k} |S_i| \cdot \mathrm{vol}(S_i) \leqslant \frac{18 \cdot \eta_S}{\Phi_{\mathrm{in}}} \cdot \mathsf{OPT}_G,$$

where η_S is an upper bound of $\max_i(\Delta(S_i)/\delta(S_i))$.

Proof. We can trivially lower bound OPT_G by only considering the internal edges $e \in S_i$ of the individual clusters $G[S_i]$, and then applying the lower bound from Lemma A.1:

$$\begin{aligned}
\mathsf{OPT}_G &\geqslant \sum_{i=1}^{k} \sum_{e \in E[G[S_i]]} \mathsf{cost}_{G[S_i]}(e) \geqslant \sum_{i=1}^{k} \mathsf{OPT}_{G[S_i]} \\
&\geqslant \sum_{i=1}^{k} \frac{2 \cdot \Phi_{G[S_i]}}{9} \cdot |S_i| \cdot \mathrm{vol}(G[S_i]) \cdot \frac{d_{G[S_i]}}{\Delta_{G[S_i]}} \\
&\geqslant \frac{2 \cdot \Phi_{\mathrm{in}}}{9} \cdot \sum_{i=1}^{k} |S_i| \cdot \frac{\mathrm{vol}(S_i)}{2} \cdot \frac{1}{2 \cdot \eta_S}.
\end{aligned}$$

Here, the second inequality follows by Lemma A.1 and the last inequality holds because of the assumptions in the Lemma and the fact that

$$\mathrm{vol}(S_i) = \mathrm{vol}(G[S_i]) + w(S_i, V \setminus S_i) \leqslant 2 \cdot \mathrm{vol}(G[S_i])$$

when $\Phi_{\mathrm{out}} \leqslant 1/2$, and because

$$\frac{d_{G[S_i]}}{\Delta_{G[S_i]}} \geqslant \frac{d_{G[S_i]}}{\Delta(S_i)} \geqslant \frac{\delta(S_i)}{\Delta(S_i) \cdot 2} \geqslant \frac{1}{2 \cdot \eta_S}.$$

Rearranging the terms above proves the statement. □

A.2. Spectral Clustering

Another component used in our analysis is spectral clustering. To analyse the theoretical performance of spectral clustering, one can examine the scenario in which there is a large gap between λ_{k+1} and $\rho(k)$. By the higher-order Cheeger inequality (1), we know that a low value of $\rho(k)$ ensures that the vertex set V of G can be partitioned into k subsets (clusters), each of which has conductance upper bounded by $\rho(k)$; on the other hand, a large value of λ_{k+1} implies that any $(k+1)$-th partition of V would introduce some $A \subset V$ with conductance $\Phi_G(A) \geqslant \rho(k+1) \geqslant \lambda_{k+1}/2$. Based on this, Peng et al. (2017) introduced the parameter

$$\Upsilon(k) \triangleq \frac{\lambda_{k+1}}{\rho(k)} \tag{9}$$

and showed that a large value of $\Upsilon(k)$ is sufficient to guarantee a good performance of spectral clustering. The result of Peng et al. (2017) has been improved by a sequence of works, and the following result, which can be shown easily by combining the proof technique of Peng et al. (2017) and Macgregor & Sun (2022), will be used in our analysis.

Lemma A.9. *There is an absolute constant $C_{A.9} \in \mathbb{R}_{>0}$, such that the following holds: Let G be a graph with k optimal clusters $\{S_i\}_{i=1}^k$, and $\Upsilon(k) \geqslant C_{A.9} \cdot k$. Let $\{P_i\}_{i=1}^k$ be the output of spectral clustering and, without loss of generality, the optimal correspondence of P_i is S_i for any $1 \leqslant i \leqslant k$. Then, it holds for any $1 \leqslant i \leqslant k$ that*

$$\mathrm{vol}(P_i \triangle S_i) \leqslant \frac{k \cdot C_{A.9}}{3\Upsilon(k)} \cdot \mathrm{vol}(S_i),$$

where $A \triangle B$ for any sets A and B is defined by $A \triangle B \triangleq (A \setminus B) \cup (B \setminus A)$. Moreover, these $P_1, \ldots, P_k$ can be computed in nearly-linear time.

A.3. Strong Decomposition Lemma

The objective of this subsection is to prove the following decomposition lemma. The lemma shows that, under a certain eigen-gap condition, an input graph G can be partitioned into clusters with bounded inner and outer conductance, and certain constraints on cut values. We remark that, while obtaining the partition promised by the lemma below requires high time complexity, we only need the existence of such partition in our analysis.

Lemma A.10 (Improved Strong Decomposition Lemma)**.** *Let $G = (V, E, w)$ be a graph such that $\lambda_{k+1} > 0$ and $\lambda_k < \left(\frac{1}{270 \cdot c_0 \cdot (k+1)^6}\right)^2$, where c_0 is the constant in Lemma A.12. Then, there is a polynomial-time algorithm that finds an ℓ-partition of V into sets $\{C_i\}_{i=1}^{\ell}$, for some $\ell \leqslant k$, such that for every $1 \leqslant i \leqslant \ell$ and every vertex $u \in C_i$ the following properties hold:*

(A1) $\Phi(C_i) = O(k^6\sqrt{\lambda_k})$;

(A2) $\Phi_{G[C_i]} = \Omega(\lambda_{k+1}^2/k^4)$;

(A3) $w(u, V \setminus C_i) \leqslant 6(k+1) \cdot \mathrm{vol}_{G[C_i]}(u)$.

We remark that the first two properties of the partitioning promised by Lemma A.10 are the same as the ones from Manghiuc & Sun (2021). However, the third property of our lemma is stronger than theirs, as Property (A3) now holds for *all* vertices in $u \in C_i$, instead of only the critical nodes $N \in \mathcal{S}_i$. We emphasise that this improved decomposition is crucial for our final analysis. In particular, we only use Lemma A.10 to show the *existence* of a strong decomposition with properties (A1), (A2), and (A3), and we use the strengthened property (A3) in the analysis of the approximation factor of our algorithm.

Before starting the analysis, by convention we set $\Phi_G(V) \triangleq 0$, and $\Phi_G(\emptyset) \triangleq 1$. We also set $w(\emptyset, S) \triangleq 0$ for any non-empty subset $S \subset V$. The following two results will be used in our analysis.

Lemma A.11 (Cheeger Inequality, (Alon, 1986))**.** *It holds for any graph G that $\frac{\lambda_2}{2} \leqslant \Phi_G \leqslant \sqrt{2\lambda_2}$. Furthermore, there is a nearly-linear time algorithm (i.e., the `Spectral Partitioning` algorithm) that finds a set S such that $\mathrm{vol}(S) \leqslant \mathrm{vol}(V)/2$, and $\Phi_G(S) \leqslant \sqrt{2\lambda_2}$.*

Lemma A.12 (Lemma 1.13, (Oveis Gharan & Trevisan, 2014))**.** *There is an absolute constant $c_0 > 1$ such that for any $k \geqslant 2$ and any r-way partition $C_1, \ldots, C_r$ of V, where $r \leqslant k-1$, we have that*

$$\min_{1 \leqslant i \leqslant r} \lambda_2\left(\mathcal{L}_{G[C_i]}\right) \leqslant 2c_0 \cdot k^6 \cdot \lambda_k.$$

Now we describe the underlying algorithm and show a sequence of claims, which are used to prove Lemma A.10. At the very high level, our algorithm for computing a stronger decomposition of a well-clustered graph can be viewed as an adjustment to Algorithm 3 in (Manghiuc & Sun, 2021), which itself is based on Algorithm 3 in (Oveis Gharan & Trevisan, 2014) The main idea can be summarised as follows: the algorithm starts with the trivial 1-partition of G, i.e., $C_1 = V$; in every iteration, the algorithm applies the `Spectral Partitioning` algorithm for every graph in $\{G[C_i]\}_{i=1}^{r}$, and tries to find a sparse cut $(S, C_i \setminus S)$ for some $S \subset C_i$.[4]

- If such a cut is found, the algorithm uses this cut to either introduce a new partition set C_{r+1} of low conductance, or refine the current partition $\{C_i\}_{i=1}^{r}$;
- If no such cut is found, the algorithm checks if it is possible to perform a local refinement of the partition sets $\{C_i\}_{i=1}^{r}$ in order to reduce the overall weight of the crossing edges, i.e. $\sum_{i \neq j} w(C_i, C_j)$. If such a refinement is not possible, the algorithm terminates and outputs the current partition; otherwise, the partition sets are locally refined and the process is repeated.

The output of the algorithm is guaranteed to satisfy Properties $(A1)$ and $(A2)$ of Lemma A.10.

Our improved analysis will show that Property $(A3)$ holds as well, and this will be proven with the two additional Properties $(A4)$ and $(A5)$ stated later. We begin our analysis by setting the notation, most of which follows from (Oveis Gharan & Trevisan, 2014). We write $\{C_i\}_{i=1}^{r}$ as a partition of V for some integer $r \geqslant 1$, and every partition set C_i contains some *core set* denoted by $\mathrm{core}(C_i) \subseteq C_i$. For an arbitrary subset $S \subset C_i$, we define $S^+ \triangleq S \cap \mathrm{core}(C_i)$, and $S^- \triangleq S \setminus S^+$. We further define $\overline{S^+} \triangleq \mathrm{core}(C_i) \setminus S$, and $\overline{S^-} \triangleq C_i \setminus (S \cup \mathrm{core}(C_i))$. Note that $\{S^+, \overline{S^+}\}$ forms a partition of $\mathrm{core}(C_i)$, and $\{S^-, \overline{S^-}\}$ forms a partition of $C_i \setminus \mathrm{core}(C_i)$. For the ease of presentation, we always write u^+ and u^- if the set S consists of a single vertex u. For any sets $S, T \subseteq V$ which are not necessarily disjoint, we write

$$w(S \to T) \triangleq w(S, T \setminus S),$$

For any subsets $S \subseteq C \subseteq V$, we follow (Oveis Gharan & Trevisan, 2014) and define the *relative conductance* as

$$\varphi(S, C) \triangleq \frac{w(S \to C)}{\frac{\mathrm{vol}(C \setminus S)}{\mathrm{vol}(C)} \cdot w(S \to V \setminus C)},$$

whenever the right-hand side is defined and otherwise we set $\varphi(S, C) = 1$. To explain the meaning of $\varphi(S, C)$, suppose that $C \subset V$ is the vertex set such that $\Phi_G(C)$ is low and $\Phi_{G[C]}$ is high, i.e., C is a cluster. Then, we know that most of the subsets $S \subset C$ with $\mathrm{vol}(S) \leqslant \mathrm{vol}(C)/2$ satisfy the following properties:

- Since $\Phi_{G[C]}(S)$ is high, a large fraction of the edges adjacent to vertices in S would leave S;
- Since $\Phi_G(C)$ is low, a small fraction of edges adjacent to S would leave C.

Combining the above observations, one could conclude that $w(S \to C) \gtrsim w(S \to V \setminus C)$ if C is a good cluster, which means that $\varphi(S, C)$ is lower bounded by a constant. Moreover, Oveis Gharan and Trevisan (Oveis Gharan & Trevisan, 2014) showed a converse of this fact: if $\varphi(S, C)$ is large for all $S \subset C$, then C has high inner conductance. These facts suggest that the relative conductance provides a good quantitative measure of the quality of a cluster.

Now we explain the high-level idea of the proposed algorithm, and refer the reader to Algorithm 3 for the formal description. Our algorithm starts with the partitioning algorithm (Algorithm 3 in (Oveis Gharan & Trevisan, 2014)), and obtains an intermediate partition $\{C_i\}_{i=1}^{r}$ (Lines 6–22). For every C_i $(1 \leqslant i \leqslant r)$, the algorithm further checks if the following conditions are satisfied:

$(A4)$ For every vertex $u \in C_i$ with $\mathrm{vol}(u^+) \leqslant \mathrm{vol}(\mathsf{CORE}(C_i))/2$, it holds that $\varphi(u^+, \mathsf{CORE}(C_i)) \geqslant \frac{1}{3(k+1)}$;

$(A5)$ For every vertex $u \in C_i$ with $\mathrm{vol}(u^-) \leqslant \mathrm{vol}(C_i)/2$, it holds that $w(u^- \to C_i) \geqslant w(u^- \to V \setminus C_i) \cdot \frac{1}{k+1}$.

[4] We denote by r the number of clusters in the current run of the algorithm, and denote by ℓ the final number of clusters output by the algorithm.

If (A4) is violated by some vertex $u \in C_i$ for some i, then the algorithm uses u to refine the core set $\mathrm{core}(C_i)$ (Line 27). If $(A5)$ is not satisfied, then the algorithm further refines the partition (Line 30). The algorithm repeats this local refinement process until no such update is found anymore. In the following analysis, we set

$$\rho^* \triangleq \min\left\{\frac{\lambda_{k+1}}{10}, 30c_0 \cdot (k+1)^5 \cdot \sqrt{\lambda_k}\right\},$$

where c_0 is the constant specified in Lemma A.12, and

$$\phi_{\mathrm{in}} \triangleq \frac{\lambda_{k+1}}{140(k+1)^2}, \qquad \phi_{\mathrm{out}} \triangleq 90c_0 \cdot (k+1)^6\sqrt{\lambda_k}. \tag{10}$$

Notice that, by assuming $\lambda_k < \left(\frac{1}{270 \cdot c_0 \cdot (k+1)^6}\right)^2$ in Lemma A.10, it holds that $\phi_{\mathrm{out}} < 1/3$. This fact will be used several times in our analysis.

Following the proof structure in (Oveis Gharan & Trevisan, 2014), we will prove Lemma A.10 via a sequence of claims. Notice that, during the entire execution of the algorithm, the sets $\{C_i\}_{i=1}^r$ always form a partition of V, and each $\mathsf{CORE}(C_i)$ is a subset of C_i. Firstly, we show that, at any point during the execution of the algorithm, the core sets $\mathsf{CORE}(C_i)$ $(1 \leqslant i \leqslant r)$ always have low conductance.

Claim A.1. *Throughout the algorithm, we always have that*

$$\max_{1 \leqslant i \leqslant r} \Phi_G(\mathsf{CORE}(C_i)) \leqslant \rho^* \cdot \left(1 + \frac{1}{k+1}\right)^r.$$

The following result will be used in our proof:

Lemma A.13 (Lemma 2.2, (Oveis Gharan & Trevisan, 2014))**.** *Let $G = (V, E, w)$ be a graph, and let S, W be two subsets such that $S \subset W \subseteq V$. Suppose that the following two conditions are satisfied for some $\varepsilon > 0$:*

1. $\varphi(S, W) \leqslant \varepsilon/3$ *and*
2. $\max\{\Phi_G(S), \Phi_G(W \setminus S)\} \geqslant (1+\varepsilon) \cdot \Phi_G(W)$.

Then it holds that

$$\min\{\Phi_G(S), \Phi_G(W \setminus S)\} \leqslant \Phi_G(W).$$

Proof of Claim A.1. Let r be the current number of clusters generated by the algorithm, and we prove by induction that the claim holds during the entire execution of the algorithm. First of all, for the base case of $r = 1$, we have that $\mathsf{CORE}(C_1) = C_1 = V$, which gives us that $\Phi_G(\mathsf{CORE}(C_1)) = 0$; hence, the statement holds trivially.

Secondly, for the inductive step, we assume that the statement holds for some fixed configuration of the core sets $\{\mathrm{core}(C_i)\}_{i=1}^r$ and we prove that the statement holds after the algorithm updates the current configuration. Notice that $\{\mathsf{CORE}(C_i)\}_{i=1}^r$ are updated through Lines 10, 13, 16, and 27 of the algorithm, so it suffices to show that the claim holds after executing these lines. We continue the proof with case distinction.

- When executing Lines 10, 16, the algorithm introduces some new set $\mathsf{CORE}(C_{r+1})$ such that

  $$\Phi_G(\mathsf{CORE}(C_{r+1})) \leqslant \rho^* \cdot \left(1 + \frac{1}{k+1}\right)^{r+1}.$$

  Combining this with the inductive hypothesis, which assumes the inequality holds for $\mathsf{CORE}(C_i)$ $(1 \leqslant i \leqslant r)$, we have that

  $$\max_{1 \leqslant i \leqslant r+1} \Phi_G(\mathsf{CORE}(C_i)) \leqslant \rho^* \cdot \left(1 + \frac{1}{k+1}\right)^{r+1}.$$

[5] If the set $S \subset C_i$ returned by `Spectral Partitioning` has $\mathrm{vol}(S^+) > \mathrm{vol}(G)/2$, swap S with $C_i \setminus S$.

Algorithm 3 Algorithm for partitioning G into $\ell \leqslant k$ clusters

Input: $G = (V, E, w)$, $k > 1$ such that $\lambda_{k+1} > 0$;
Output: A $\left(\phi_{\text{in}}^2/4, \phi_{\text{out}}\right)$ ℓ-partition $\{C_i\}_{i=1}^{\ell}$ of G satisfying $(A1) - (A3)$, for some $\ell \leqslant k$;
Let $r = 1$, $\mathsf{CORE}(C_1) = C_1 = V$;
Let $\phi_{\text{in}} = \frac{\lambda_{k+1}}{140(k+1)^2}$, and $\phi_{\text{out}} = 90c_0 \cdot (k+1)^6\sqrt{\lambda_k}$;
Let $\rho^* = \min\left\{\frac{\lambda_{k+1}}{10}, 30c_0 \cdot (k+1)^5 \cdot \sqrt{\lambda_k}\right\}$;
while At least one of the following conditions holds

1. $\exists 1 \leqslant i \leqslant r$ such that $w(C_i \setminus \mathsf{CORE}(C_i) \to C_i) < w(C_i \setminus \mathsf{CORE}(C_i) \to C_j)$ for some $j \neq i$;
2. `Spectral Partitioning` finds $S \subseteq C_i$ with [5] $\text{vol}(S^+) \leqslant \text{vol}(\text{core}(C_i))/2$, such that $\max\left\{\Phi_{G[C_i]}(S), \Phi_{G[C_i]}(C_i \setminus S)\right\} < \phi_{\text{in}}$;

do
  Order the sets $C_1, \ldots, C_r$ such that $\lambda_2(\mathcal{L}_{G[C_1]}) \leqslant \ldots \leqslant \lambda_2(\mathcal{L}_{G[C_r]})$;
  Let $1 \leqslant i \leqslant r$ the smallest index for which item 2 of the `while`-condition is satisfied, and let $S \subset C_i$ be the corresponding set;
  if $\max\left\{\Phi_G(S^+), \Phi_G\left(\overline{S^+}\right)\right\} \leqslant \left(1 + \frac{1}{k+1}\right)^{r+1} \cdot \rho^*$ **then**
    Let $C_i = C_i \setminus \overline{S^+}$, $\mathsf{CORE}(C_i) = S^+$, $C_{r+1} = \mathsf{CORE}(C_{r+1}) = \overline{S^+}$, $r = r + 1$ and **go to** Line 6;
  end if
  if $\min\left\{\varphi(S^+, \mathsf{CORE}(C_i)), \varphi\left(\overline{S^+}, \mathsf{CORE}(C_i)\right)\right\} \leqslant \frac{1}{3(k+1)}$ **then**
    Update $\mathsf{CORE}(C_i)$ to either S^+ or $\overline{S^+}$ with the lower conductance, and **go to** Line 6;
  end if
  if $\Phi_G(S^-) \leqslant \left(1 + \frac{1}{k+1}\right)^{r+1} \cdot \rho^*$ **then**
    Let $C_i = C_i \setminus S^-$, $C_{r+1} = \mathsf{CORE}(C_{r+1}) = S^-$, set $r = r + 1$ and **go to** Line 6;
  end if
  if $w(C_i \setminus \mathsf{CORE}(C_i) \to C_i) < w(C_i \setminus \mathsf{CORE}(C_i) \to C_j)$ for some $j \neq i$ **then**
    Let $C_i = \mathsf{CORE}(C_i)$, merge $(C_i \setminus \mathsf{CORE}(C_i))$ with $\text{argmax}_{C_j}\{w(C_i \setminus \mathsf{CORE}(C_i) \to C_j)\}$, and **go to** Line 6;
  end if
  if $w(S^- \to C_i) < w(S^- \to C_j)$ for some $j \neq i$ **then**
    Let $C_i = C_i \setminus S^-$, merge S^- with $\text{argmax}_{C_j}\{w(S^- \to C_j)\}$, and **go to** Line 6;
  end if
end while
For every partition set C_i
if $\exists u \in C_i$ such that $\text{vol}(u^+) \leqslant \text{vol}(\mathsf{CORE}(C_i))/2$ and $\varphi(u^+, \mathsf{CORE}(C_i)) \leqslant \frac{1}{3(k+1)}$ **then**
  Update $\mathsf{CORE}(C_i)$ to either u^+ or $\overline{u^+}$ with the lower conductance, and **go to** Line 6;
end if
if $\exists u \in C_i$ such that $\text{vol}(u^-) \leqslant \text{vol}(C_i)/2$ and $w(u^- \to C_i) < w(u^- \to C_j)$ for some $j \neq i$ **then**
  Let $C_i = C_i \setminus u^-$, merge u^- with $\text{argmax}_{C_j}\{w(u^- \to C_j)\}$, and **go to** Line 6;
end if
Return: the partition $\{C_i\}_{i=1}^{r}$.

- The case for executing Lines 13 and 27 is similar, and we prove this by applying Lemma A.13. We first focus on Line 13 here, dealing with Line 27 next. When executing Line 13, we know that the `if`-condition in Line 9 does not hold, so we have that

$$\max\left\{\Phi_G(S^+), \Phi_G\left(\overline{S^+}\right)\right\} > \left(1+\frac{1}{k+1}\right)^{r+1}\cdot\rho^* \geqslant \left(1+\frac{1}{k+1}\right)\cdot\Phi_G(\mathsf{CORE}(C_i)),$$

where the last inequality follows by the inductive hypothesis. Moreover, when executing Line 13, we also know that the `if`-condition in Line 12 holds, i.e.,

$$\min\left\{\varphi\left(S^+, \mathsf{CORE}(C_i)\right), \varphi\left(\overline{S^+}, \mathsf{CORE}(C_i)\right)\right\} \leqslant \frac{1}{3(k+1)}.$$

Therefore, by applying Lemma A.13 with $S^+ \subset \mathrm{core}(C_i)$ and $\varepsilon = 1/(k+1)$ and using the inductive hypothesis, we conclude that

$$\min\left\{\Phi_G(S^+), \Phi_G\left(\overline{S^+}\right)\right\} \leqslant \Phi_G(\mathsf{CORE}(C_i)) \leqslant \rho^*\cdot\left(1+\frac{1}{k+1}\right)^{r}.$$

- Now we look at Line 27. When executing Line 27, we know that $u^+ = u$; since otherwise $u^+ = \emptyset$ and $\varphi(\emptyset, \mathsf{CORE}(C_i)) = 1$. Therefore, it holds that

$$1 = \max\left\{\Phi_G(u^+), \Phi_G\left(\overline{u^+}\right)\right\} > \left(1+\frac{1}{k+1}\right)^{r+1}\cdot\rho^* \geqslant \left(1+\frac{1}{k+1}\right)\cdot\Phi_G(\mathsf{CORE}(C_i)),$$

where the first inequality holds because $\Phi_G(u^+) = 1$, and the last inequality follows by the inductive hypothesis. Moreover, when executing Line 27, we also know that the `if`-condition in Line 26 holds, i.e.,

$$\varphi\left(u^+, \mathsf{CORE}(C_i)\right) \leqslant \frac{1}{3(k+1)}.$$

Therefore, by applying Lemma A.13 with $u^+ \subset \mathrm{core}(C_i)$ and $\varepsilon = 1/(k+1)$ and using the inductive hypothesis, we conclude that

$$\min\left\{\Phi_G(u^+), \Phi_G\left(\overline{u^+}\right)\right\} = \Phi_G\left(\overline{u^+}\right) \leqslant \Phi_G(\mathsf{CORE}(C_i)) \leqslant \rho^*\cdot\left(1+\frac{1}{k+1}\right)^{r},$$

where the first equality holds because $\Phi_G(u^+) = 1$ holds for $u^+ = u$.

Combining the two cases above, we know that the claim always holds during the entire execution of the algorithm. This completes the proof. □

Next, we show that the number of partition sets cannot exceed k. This proof is identical to Claim 3.2 in (Oveis Gharan & Trevisan, 2014), and we include the proof here for completeness.

Claim A.2. *The total number of clusters returned by the algorithm satisfies that $\ell \leqslant k$.*

Proof. Suppose for contradiction that the number of clusters becomes $r = k+1$ at some point during the execution of the algorithm. Then, since $\mathsf{CORE}(C_1), \ldots, \mathsf{CORE}(C_{k+1})$ are disjoint, by the definition of $\rho(k+1)$ and Claim A.1 we have that

$$\rho(k+1) \leqslant \max_{1\leqslant i\leqslant k+1}\Phi_G(\mathrm{core}(C_i)) \leqslant \left(1+\frac{1}{k+1}\right)^{k+1}\cdot\rho^* \leqslant \mathrm{e}\cdot\rho^* \leqslant \mathrm{e}\cdot\frac{\lambda_{k+1}}{10} < \frac{\lambda_{k+1}}{2},$$

which contradicts the higher-order Cheeger inequality (1). Therefore, the total number of clusters at any time satisfies $r < k+1$, and the total number of returned clusters satisfies $\ell \leqslant k$. □

Now we are ready to show that the output $\{C_i\}_{i=1}^{\ell}$ of Algorithm 3 and its core sets $\{\mathsf{CORE}(C_i)\}_{i=1}^{\ell}$ satisfy Properties $(A4)$ and $(A5)$, which will be used in proving Lemma A.10.

Claim A.3. *Let $\{C_i\}_{i=1}^{\ell}$ be the output of Algorithm 3 with the corresponding core sets $\{\mathsf{CORE}(C_i)\}_{i=1}^{\ell}$. Then, the following hold for any $1 \leqslant i \leqslant \ell$:*

1. $\Phi_G(\mathsf{CORE}(C_i)) \leqslant \phi_{\mathrm{out}}/(k+1)$;
2. $\Phi_G(C_i) \leqslant \phi_{\mathrm{out}}$;
3. $\Phi_{G[C_i]} \geqslant \phi_{\mathrm{in}}^2/4$;
4. *For every $u \in C_i$ with $\mathrm{vol}(u^+) \leqslant \mathrm{vol}(\mathsf{CORE}(C_i))/2$, we have that*
$$\varphi(u^+, \mathsf{CORE}(C_i)) \geqslant \frac{1}{3(k+1)};$$
5. *For every $u \in C_i$ with $\mathrm{vol}(u^-) \leqslant \mathrm{vol}(C_i)/2$, we have that*
$$w(u^- \to C_i) \geqslant w(u^- \to V \setminus C_i) \cdot \frac{1}{k+1}.$$

Proof. First of all, by Claim A.1 we have for any $1 \leqslant i \leqslant \ell$ that

$$\Phi_G(\mathsf{CORE}(C_i)) \leqslant \rho^* \cdot \left(1 + \frac{1}{k+1}\right)^{\ell} \leqslant \mathrm{e} \cdot \rho^* \leqslant 30 \cdot \mathrm{e} \cdot c_0 \cdot (k+1)^5 \sqrt{\lambda_k} \leqslant \frac{\phi_{\mathrm{out}}}{k+1},$$

where the second inequality holds by the fact that $\ell \leqslant k+1$, the third one holds by the choice of ρ^*, and the last one holds by the choice of ϕ_{out}. This proves Item (1).

To prove Item (2), we notice that the first condition of the `while`-loop (Line 6) doesn't hold when the algorithm terminates, hence we have for any $1 \leqslant i \neq j \leqslant \ell$ that

$$w(C_i \setminus \mathsf{CORE}(C_i) \to C_i) \geqslant w(C_i \setminus \mathsf{CORE}(C_i) \to C_j).$$

By applying the averaging argument, we have that

$$\begin{aligned} w(C_i \setminus \mathsf{CORE}(C_i) \to \mathsf{CORE}(C_i)) &= w(C_i \setminus \mathsf{CORE}(C_i) \to C_i) \\ &\geqslant \frac{w(C_i \setminus \mathsf{CORE}(C_i) \to V)}{\ell} \geqslant \frac{w(C_i \setminus \mathsf{CORE}(C_i) \to V \setminus C_i)}{k}. \end{aligned} \tag{11}$$

We apply the same analysis used in (Oveis Gharan & Trevisan, 2014), and have that

$$\begin{aligned} \Phi_G(C_i) &= \frac{w(C_i \to V)}{\mathrm{vol}(C_i)} \\ &\leqslant \frac{w\left(\mathrm{core}(C_i) \to V\right) + w\left(C_i \setminus \mathrm{core}(C_i) \to V \setminus C_i\right) - w(C_i \setminus \mathrm{core}(C_i) \to \mathrm{core}(C_i))}{\mathrm{vol}(\mathrm{core}(C_i))} \\ &\leqslant \Phi_G(\mathrm{core}(C_i)) + \frac{(k-1) \cdot w(C_i \setminus \mathrm{core}(C_i) \to \mathrm{core}(C_i))}{\mathrm{vol}(\mathrm{core}(C_i))} \\ &\leqslant k \cdot \Phi_G(\mathrm{core}(C_i)) \\ &\leqslant \phi_{\mathrm{out}}, \end{aligned}$$

where the second inequality uses equation (11). This proves Item (2).

Next, we analyse Item (3). Again, we know that the second condition within the `while`-loop (Line 6) does not hold when the algorithm terminates. By the performance of the `Spectral Partitioning` algorithm (i.e., Lemma A.11), it holds for any $1 \leqslant i \leqslant \ell$ that $\Phi_{G[C_i]} \geqslant \phi_{\mathrm{in}}^2/4$. With this, we prove that Item (3) holds.

Similarly, when the algorithm terminates, we know that for any node $u \in C_i$ the `if`-condition in Line 26 does not hold. Hence, for any $1 \leqslant i \leqslant \ell$ and any $u \in C_i$ with $\mathrm{vol}(u^+) \leqslant \mathrm{vol}(\mathsf{CORE}(C_i))/2$, we have that

$$\varphi(u^+, \mathsf{CORE}(C_i)) \geqslant \frac{1}{3(k+1)}.$$

This shows that Item (4) holds as well.

Finally, since there is no $u \in C_i$ satisfying the `if`-condition in Line 29 of the algorithm, it holds for any $1 \leqslant i \neq j \leqslant \ell$ and every vertex $u \in C_i$ that $w(u^- \to C_i) \geqslant w(u^- \to C_j)$. Therefore, by the same averaging argument we have that

$$w(u^- \to C_i) \geqslant \frac{w(u^- \to V)}{\ell} \geqslant \frac{w(u^- \to V \setminus C_i)}{k+1},$$

which shows that Item (5) holds. □

It remains to prove that the algorithm does terminate. To prove this, we first show that, in each iteration of the `while`-loop (Lines 7–22), at least one of the `if`-conditions will be satisfied, and some sets are updated accordingly. This fact, stated as Claim A.4, is important, since otherwise the algorithm might end up in an infinite loop. The following result will be used in our proof.

Lemma A.14 (Lemma 2.6, (Oveis Gharan & Trevisan, 2014))**.** *Let* $\mathrm{core}(C_i) \subseteq C_i \subseteq V$*, and* $S \subset C_i$ *be such that* $\mathrm{vol}(S^+) \leqslant \mathrm{vol}(\mathrm{core}(C_i))/2$*. Suppose that the following hold for some parameters* ρ *and* $0 < \varepsilon < 1$*:*

1. $\rho \leqslant \Phi_G(S^-)$ *and* $\rho \leqslant \max\{\Phi_G(S^+), \Phi_G(\overline{S^+})\}$*;*
2. *If* $S^- \neq \emptyset$*, then* $w(S^- \to C_i) \geqslant w(S^- \to V)/k$*;*
3. *If* $S^+ \neq \emptyset$*, then* $\varphi(S^+, \mathrm{core}(C_i)) \geqslant \varepsilon/3$ *and* $\varphi\left(\overline{S^+}, \mathrm{core}(C_i)\right) \geqslant \varepsilon/3$*.*

Then, it holds that

$$\Phi_{G[C_i]}(S) \geqslant \varepsilon \cdot \frac{\rho}{14k}.$$

Claim A.4. *If at least one condition of the* `while`*-loop is satisfied, then at least one of the* `if`*-conditions (Lines 9,12,15,18 or 21) is satisfied.*

Proof. First of all, notice that if the first condition of the `while`-loop is satisfied, then the `if`-condition in Line 18 will be satisfied and the claim holds. Hence, we assume that only the second condition of the `while`-loop is satisfied, and we prove the claim by contradiction. That is, we show that, if none of the `if`-conditions holds, then the set S returned by the `Spectral Partitioning` algorithm would satisfy that $\Phi_{G[C_i]}(S) \geqslant \phi_{\mathrm{in}}$. The proof is structured in the following two steps:

1. We first prove that
$$\Phi_{G[C_i]}(S) \geqslant \frac{\max\{\rho^*, \rho(r+1)\}}{14(k+1)^2};$$
2. Using Item (1) we prove that $\Phi_{G[C_i]}(S) \geqslant \phi_{\mathrm{in}}$ and reach our desired contradiction.

Step 1: We prove this fact by applying Lemma A.14 with parameters

$$\rho \triangleq \max\{\rho^*, \rho(r+1)\} \quad \text{and} \quad \varepsilon \triangleq \frac{1}{k+1}.$$

Let us show that the conditions of Lemma A.14 are satisfied, beginning with the first one. If $S^- = \emptyset$, then we trivially have that $1 = \Phi_G(S^-) \geqslant \rho$; so we assume that $S^- \neq \emptyset$. As the `if`-condition in Line 15 is not satisfied, we have that

$$\Phi_G(S^-) \geqslant \left(1 + \frac{1}{k+1}\right)^{r+1} \cdot \rho^*,$$

and combining this with Claim A.1 gives us that

$$\Phi_G(S^-) = \max\left\{\Phi_G(\mathsf{CORE}(C_1)), \ldots, \Phi_G(\mathsf{CORE}(C_r)), \Phi_G(S^-)\right\} \geqslant \rho(r+1).$$

Therefore, we have that

$$\Phi_G(S^-) \geqslant \max\{\rho^*, \rho(r+1)\} = \rho. \tag{12}$$

Similarly, if $S^+ = \emptyset$, then we trivially have that $1 = \max\{\Phi_G(S^+), \Phi_G(\overline{S^+})\} \geqslant \rho$; so we assume that $S^+ \neq \emptyset$. Moreover, since we have chosen S^+ such that $\mathrm{vol}(S^+) \leqslant \mathrm{vol}(\mathrm{core}(C_i))/2$, we know that $\overline{S^+} = \mathrm{core}(C_i) \setminus S^+ \neq \emptyset$. As the `if`-condition in Line 9 is not satisfied, we have that

$$\max\left\{\Phi_G(S^+), \Phi_G\left(\overline{S^+}\right)\right\} \geqslant \left(1 + \frac{1}{k+1}\right)^{r+1} \cdot \rho^*.$$

Combining this with Claim A.1 gives us that

$$\begin{aligned}
&\max\left\{\Phi_G(S^+), \Phi_G\left(\overline{S^+}\right)\right\} \\
&= \max\left\{\Phi_G(\mathsf{CORE}(C_1)), \ldots, \Phi_G(\mathsf{CORE}(C_{i-1})), \Phi_G(S^+), \Phi_G\left(\overline{S^+}\right), \Phi_G(\mathsf{CORE}(C_{i+1})), \ldots, \Phi_G(\mathsf{CORE}(C_r))\right\} \\
&\geqslant \rho(r+1).
\end{aligned}$$

Therefore we have that

$$\max\left\{\Phi_G(S^+), \Phi_G\left(\overline{S^+}\right)\right\} \geqslant \max\{\rho^*, \rho(r+1)\} = \rho. \tag{13}$$

Combining (12) and (13), we see that the first condition of Lemma A.14 is satisfied. Since the `if`-condition in Line 21 is not satisfied, it follows by an averaging argument that

$$w(S^- \to C_i) \geqslant \frac{w(S^- \to V)}{k},$$

which shows that the second condition of Lemma A.14 is satisfied. Finally, since the `if`-condition in Line 12 is not satisfied, we know that

$$\min\left\{\varphi(S^+, \mathsf{CORE}(C_i)), \varphi\left(\overline{S^+}, \mathsf{CORE}(C_i)\right)\right\} \geqslant \frac{1}{3(k+1)},$$

which shows that the third condition of Lemma A.14 is satisfied as well. Hence, by Lemma A.14 we conclude that

$$\Phi_{G[C_i]}(S) \geqslant \frac{\varepsilon \cdot \rho}{14(k+1)} = \frac{\max\{\rho^*, \rho(r+1)\}}{14(k+1)^2}, \tag{14}$$

which completes the proof of the first step.

Step 2: We prove this step with a case distinction as follows:

Case 1: $r = k$. By (14) and (1), we have that

$$\Phi_{G[C_i]}(S) \geqslant \frac{\rho(r+1)}{14(k+1)^2} = \frac{\rho(k+1)}{14(k+1)^2} \geqslant \frac{\lambda_{k+1}}{28(k+1)^2} \geqslant \phi_{\mathrm{in}},$$

which leads to the desired contradiction.

Case 2: $r < k$. Recall that the partition sets $\{C_i\}_{i=1}^r$ are labelled such that $\lambda_2(\mathcal{L}_{G[C_1]}) \leqslant \ldots \leqslant \lambda_2(\mathcal{L}_{G[C_r]})$, and the algorithm has chosen the lowest index i for which the set $S \subset C_i$ returned by the `Spectral Partitioning` algorithm satisfies the second condition of the `while`-loop. Our proof is based on a further case distinction depending on the value of i.

Case 2a: $i = 1$ (i.e., the algorithm selects $S \subseteq C_1$). We combine the performance of the `Spectral Partitioning` algorithm (Lemma A.11) with Lemma A.12, and obtain that

$$\Phi_{G[C_1]}(S) \leqslant \sqrt{2\lambda_2(\mathcal{L}_{G[C_1]})} = \min_{1 \leqslant j \leqslant r} \sqrt{2\lambda_2(\mathcal{L}_{G[C_j]})} \leqslant \sqrt{4c_0 \cdot k^6 \cdot \lambda_k}. \tag{15}$$

Combining (14) and (15) we have that

$$\rho^* \leqslant 28c_0 \cdot (k+1)^5 \sqrt{\lambda_k}.$$

Thus, by the definition of ρ^* we have that

$$\rho^* = \frac{\lambda_{k+1}}{10}.$$

We combine this with (14), and have that

$$\Phi_{G[C_i]}(S) \geqslant \frac{\lambda_{k+1}}{140(k+1)^2} = \phi_{\text{in}},$$

which gives our desired contradiction.

Case 2b: $i > 1$ (i.e., the algorithm selects $S \subset C_i$ for some $i \geqslant 2$). Let $S_1 \subset C_1$ be the set obtained by applying the `Spectral Partitioning` algorithm to the graph $G[C_1]$. Since the algorithm did not select $S_1 \subset C_1$, we know that $\Phi_{G[C_1]}(S_1) \geqslant \phi_{\text{in}}$. Combining the performance of the `Spectral Partitioning` algorithm (Lemma A.11) with Lemma A.12, we have that

$$\phi_{\text{in}} \leqslant \Phi_{G[C_1]}(S_1) \leqslant \min_{1 \leqslant j \leqslant r} \sqrt{2\lambda_2(\mathcal{L}_{G[C_j]})} \leqslant 2c_0 \cdot k^3 \cdot \sqrt{\lambda_k}.$$

This gives us that

$$\frac{\lambda_{k+1}}{10} = 14(k+1)^2 \cdot \phi_{\text{in}} < 30c_0 \cdot (k+1)^5 \cdot \sqrt{\lambda_k},$$

and it holds by the definition of ρ^* that

$$\rho^* = \frac{\lambda_{k+1}}{10}.$$

Therefore, by (14) we have that

$$\Phi_{G[C_i]}(S) \geqslant \frac{\lambda_{k+1}}{140(k+1)^2} = \phi_{\text{in}}.$$

Combining the two cases above gives us the desired contradiction. With this, we complete the proof of the claim. □

Next, we will show that the total number of iterations that the algorithm runs, i.e., the number of times the instruction "`go to Line 6`" is executed, is finite.

Claim A.5. *For any graph $G = (V, E, w)$ with the minimum weight $w_{\min}$ as the input, Algorithm 3 terminates after executing the while-loop $O\left(k \cdot n \cdot \text{vol}(G)/w_{\min}\right)$ times.*

Proof. Notice that the algorithm goes back to check the loop conditions (Line 6) right after any of Lines 10, 13, 16, 19, 22, 27 and 30 is executed, and each of these commands changes the current structure of our partition $\{C_i\}_{i=1}^r$ with core sets $\mathsf{CORE}(C_i) \subseteq C_i$. We classify these updates into the following three types:

1. The updates that introduce a new partition set C_{r+1}. These correspond to Lines 10, and 16;
2. The updates that contract the core sets $\mathsf{CORE}(C_i)$ to a strictly smaller subset $T \subset \mathsf{CORE}(C_i)$. These correspond to Lines 13 and 27;
3. The updates that refine the partition sets $\{C_i\}_{i=1}^r$ by moving a subset $T \subseteq C_i \setminus \mathsf{CORE}(C_i)$ from the partition set C_i to a different partition set C_j, for some $C_i \neq C_j$. These correspond to Lines 19, 22, and 30.

We prove that these updates can occur only a finite number of times. The first type of updates can occur at most k times, since we know by Claim A.2 that the algorithm outputs $\ell \leqslant k$ clusters. Secondly, for a fixed value of ℓ, the second type of updates occurs at most n times, since each update decreases the size of some $\mathsf{CORE}(C_i)$ by at least one. Finally, for a fixed ℓ and a fixed configuration of core sets $\mathsf{CORE}(C_i) \subseteq C_i$, the third type of updates occurs at most $O(\text{vol}(G)/w_{\min})$ times. This is due to the fact that, whenever every such update is executed, the total weight between different partition sets, i.e., $\sum_{i \neq j} w(C_i, C_j)$, decreases by at least $w_{\min}$. Combining everything together proves the lemma. □

Finally, we combine everything together and prove Lemma A.10.

Proof of Lemma A.10. We first show that Properties $(A1)$, $(A2)$ and $(A3)$ hold, and in the end we analyse the runtime of the algorithm. Combining Items (2) and (3) of Claim A.3 with the choices of $\phi_{\text{in}}, \phi_{\text{out}}$ in (10), we obtain for all $1 \leqslant i \leqslant \ell$ that $\Phi_G(C_i) \leqslant \phi_{\text{out}} = O\left(k^6 \cdot \sqrt{\lambda_k}\right)$ and $\Phi_{G[C_i]} \geqslant \phi_{\text{in}}^2/4 = \Omega\left(\lambda_{k+1}^2/k^4\right)$. Hence, Properties $(A1)$ and $(A2)$ hold for every C_i.

To analyse Property (A3), we fix an arbitrary node $u \in C_i$ that belongs to the partition set C_i with core set $\mathsf{CORE}(C_i)$. By definition, we have that

$$w(u, V \setminus C_i) = w(u^+, V \setminus C_i) + w(u^-, V \setminus C_i).$$

We study $w(u^+, V \setminus C_i)$ and $w(u^-, V \setminus C_i)$ separately.

Bounding the value of $w(u^+, V \setminus C_i)$: We analyse $w(u^+, V \setminus C_i)$ by the following case distinction.

Case 1: $\text{vol}(u^+) \leqslant \text{vol}(\mathsf{CORE}(C_i))/2$. By Item (4) of Claim A.3 we know that

$$\varphi(u^+, \mathsf{CORE}(C_i)) \geqslant \frac{1}{3(k+1)},$$

which is equivalent to

$$3(k+1) \cdot \frac{\text{vol}(\mathsf{CORE}(C_i))}{\text{vol}(\overline{u^+})} \cdot w(u^+ \to \mathsf{CORE}(C_i)) \geqslant w(u^+ \to V \setminus \mathsf{CORE}(C_i)).$$

This implies that

$$6(k+1) \cdot w(u^+ \to \mathsf{CORE}(C_i)) \geqslant w(u^+ \to V \setminus \mathsf{CORE}(C_i)),$$

and we have that

$$w(u^+, V \setminus C_i) \leqslant w(u^+ \to V \setminus \mathsf{CORE}(C_i)) \leqslant 6(k+1) \cdot w(u^+ \to \mathsf{CORE}(C_i)) \leqslant 6(k+1) \cdot \text{vol}_{G[C_i]}(u^+).$$

Case 2: $\text{vol}(u^+) > \text{vol}(\mathsf{CORE}(C_i))/2$. We have that

$$\begin{aligned} w(u^+, V \setminus C_i) &\leqslant w(\mathsf{CORE}(C_i), V \setminus C_i) \leqslant w(\mathsf{CORE}(C_i), V \setminus \mathsf{CORE}(C_i)) \\ &= \text{vol}(\mathsf{CORE}(C_i)) \cdot \Phi_G(\mathsf{CORE}(C_i)) \leqslant \text{vol}(\mathsf{CORE}(C_i)) \cdot \frac{\phi_{\text{out}}}{k+1} \\ &< \frac{2\phi_{\text{out}}}{k+1} \cdot \text{vol}(u^+), \end{aligned} \tag{16}$$

where the third inequality follows by Item (1) of Claim A.3. Therefore, we have that

$$\text{vol}_{G[C_i]}(u^+) = \text{vol}(u^+) - w(u^+, V \setminus C_i) > \text{vol}(u^+)\left(1 - \frac{2\phi_{\text{out}}}{k+1}\right), \tag{17}$$

where the last inequality follows by (16). We further combine (16) with (17), and obtain that

$$w(u^+, V \setminus C_i) \leqslant \frac{2\phi_{\text{out}}}{k+1} \cdot \frac{1}{1 - \frac{2\phi_{\text{out}}}{k+1}} \cdot \text{vol}_{G[C_i]}(u^+) \leqslant \frac{2\phi_{\text{out}}}{k} \cdot \text{vol}_{G[C_i]}(u^+),$$

where the last inequality holds by our assumption that $\phi_{\text{out}} < 1/3$.

Therefore, combining the two cases above gives us that

$$w(u^+, V \setminus C_i) \leqslant 6(k+1) \cdot \text{vol}_{G[C_i]}\left(u^+\right). \tag{18}$$

Bounding the value of $w(u^-, V \setminus C_i)$: We analyse $w(u^-, V \setminus C_i)$ based on the following two cases.

Case 1: $\text{vol}(u^-) \leqslant \text{vol}(C_i)/2$. By Item (5) of Claim A.3, we know that

$$w(u^- \to C_i) \geqslant w(u^- \to V \setminus C_i) \cdot \frac{1}{(k+1)},$$

which gives us that

$$w(u^-, V \setminus C_i) \leqslant (k+1) \cdot w(u^- \to C_i) \leqslant (k+1) \cdot \mathrm{vol}_{G[C_i]}(u^-).$$

Case 2: $\mathrm{vol}(u^-) > \mathrm{vol}(C_i)/2$. In this case, we have that

$$w(u^-, V \setminus C_i) \leqslant w(C_i, V \setminus C_i) = \Phi_G(C_i) \cdot \mathrm{vol}(C_i) \leqslant \phi_{\mathrm{out}} \cdot \mathrm{vol}(C_i) \leqslant 2\phi_{\mathrm{out}} \cdot \mathrm{vol}(u^-), \tag{19}$$

where the second inequality follows by Item (2) of Claim A.3. This implies that

$$\mathrm{vol}_{G[C_i]}(u^-) = \mathrm{vol}(u^-) - w(u^-, V \setminus C_i) \geqslant (1 - 2\phi_{\mathrm{out}}) \cdot \mathrm{vol}(u^-), \tag{20}$$

where the last inequality follows by (19). Finally, combining (19) and (20) gives us that

$$w(u^-, V \setminus C_i) \leqslant \frac{2\phi_{\mathrm{out}}}{1 - 2\phi_{\mathrm{out}}} \cdot \mathrm{vol}_{G[C_i]}(u^-) \leqslant 2 \cdot \mathrm{vol}_{G[C_i]}(u^-),$$

where the last inequality follows by our assumption that $\phi_{\mathrm{out}} < 1/3$. Therefore, combining the two cases together gives us that

$$w(u^-, V \setminus C_i) \leqslant (k+1) \cdot \mathrm{vol}_{G[C_i]}(u^-). \tag{21}$$

Our claimed property $(A3)$ follows by summing the inequalities in (18) and (21) and the fact that

$$\mathrm{vol}_{G[C_i]}(u) = \mathrm{vol}_{G[C_i]}(u^+) + \mathrm{vol}_{G[C_i]}(u^-).$$

Finally, we analyse the runtime of the algorithm. By Claims A.4 and A.5, we know that the algorithm does terminate, and the total number of iterations of the main `while`-loop executed by the algorithm is upper bounded by $O(k \cdot n \cdot \mathrm{vol}(G)/w_{\min})$. Notice that this quantity is upper bounded by $O(\mathrm{poly}(n))$ given our assumption that $w_{\max}/w_{\min} = O(\mathrm{poly}(n))$. This completes the proof. □

B. Omitted Details of Proof of Theorem 5.1

This section presents omitted details of the proof of Theorem 5.1. Our key approach of breaking the running time barrier of Manghiuc & Sun (2021) is that, instead of approximating the optimal tree $\mathcal{T}^*$, we approximate the tree constructed by the algorithm of Manghiuc & Sun (2021).

At a very high level, we first slightly adjust the tree construction by Manghiuc & Sun (2021), and show that the cost of this adjusted variant is the same as their original tree construction. We call the adjusted tree $\mathcal{T}_{\mathsf{MS}}$. Then, we perform a sequence of changes to $\mathcal{T}_{\mathsf{MS}}$, such that the total induced cost of these changes is not too high. After this sequence of changes, we end up with a tree $\mathcal{T}''_{\mathsf{MS}}$, whose cost can be approximated in nearly-linear time with the output of Algorithm 1. In particular, we only require the *existence* of the tree $\mathcal{T}_{\mathsf{MS}}$, and not its explicit construction.

The section is organised as follows: In Section B.1 we introduce a variant of the algorithm and tree construction from Manghiuc & Sun (2021). In Section B.2 we prove Lemma 5.2 and Lemma 5.3. Finally, we prove Theorem 5.1 in Section B.3.

B.1. Caterpillar Tree on Extended Critical Nodes

Now we present an adjusted version of the result on well-clustered graphs from Manghiuc & Sun (2021). Their original algorithm first uses their variant of Lemma A.10 to explicitly construct a partition $\{C_1, \ldots, C_\ell\}$, and then decomposes every C_i into critical nodes $\mathcal{S}_i$. These critical nodes are used to construct their final tree.

In our adjusted form, instead of using the critical nodes $\mathcal{S}_i$ as the building blocks, we use the "extended" critical nodes as building blocks, which are described below.

Let $C_1, \ldots C_\ell$ be the partition returned by Lemma A.10. We assume for every C_i that $(A_0, \ldots, A_r)$ is the dense branch of $\mathbb{T}_i = \mathbb{T}_{\deg}(G[C_i])$ for some $r \in \mathbb{Z}_+$, and we extend the dense branch to $(A_0, \ldots, A_{i_{\max}-1})$ with the property that, for every $i \in [r, i_{\max} - 1]$, A_i is the child of A_{i-1} with the higher volume, and $A_{i_{\max}-1}$ having children $A_{i_{\max}}$ and $B_{i_{\max}}$ such that $|A_{i_{\max}}| \leqslant 2$ and let $\mathcal{S}_i \triangleq \{B_1, \ldots, B_{i_{\max}}, A_{i_{\max}}\}$ be the set of all *extended* critical nodes.

Based on this new notion of extended critical nodes, Algorithm 4 gives a formal description of our adjusted tree called $\mathcal{T}_{\mathsf{MS}}$, which consists of three phases: `Partition`, `Prune` and `Merge`. In the `Partition` phase (Lines 3–4), the algorithm

employs Lemma A.10 to partition $V(G)$ into sets $\{C_i\}_{i=1}^{\ell}$, and applies Theorem A.4 to obtain the corresponding trees $\{\mathbb{T}_i\}_{i=1}^{\ell}$. The `Prune` phase (Lines 7–9) consists of a repeated pruning process: for every such tree $\mathbb{T}_i$, the algorithm prunes the subtree $\mathbb{T}_i[N]$, where $N \in \mathcal{S}_i$ is the extended critical node closest to the root in $\mathbb{T}_i$, and adds $\mathbb{T}_i[N]$ to $\mathbb{T}$, which is the collection of all the pruned subtrees (Line 9). The process is repeated until $\mathbb{T}_i$ is completely pruned. Finally, in the `Merge` phase (Lines 13–16) the algorithm combines the trees in $\mathbb{T}$ in a "caterpillar style" according to an increasing order of their size, where we define the size of a tree $\mathcal{T}$ as $|\mathcal{T}| \triangleq |\mathsf{leaves}(\mathcal{T})|$. We call the final constructed tree $\mathcal{T}_{\mathsf{MS}}$, and the performance of this algorithm is summarised in Lemma B.1. We emphasise *again* that in our algorithm we do not explicitly run Algorithm 4 to construct our tree. Instead, we use it to show the *existence* of tree $\mathcal{T}_{\mathsf{MS}}$, and our final analysis is to approximate the cost of $\mathcal{T}_{\mathsf{MS}}$.

Algorithm 4 Merge Extended Critical Nodes

1: **Input:** A graph $G = (V, E, w)$, a parameter $k \in \mathbb{Z}_+$ such that $\lambda_{k+1} > 0$
2: **Output:** An HC tree $\mathcal{T}_{\mathsf{MS}}$ of G
3: Apply the partitioning algorithm (Lemma A.10) on input (G, k) to obtain $\{C_i\}_{i=1}^{\ell}$ for some $\ell \leqslant k$;
4: Let $\mathbb{T}_i = \texttt{HCwithDegrees}(G[C_i])$;
5: Initialise $\mathbb{T} = \emptyset$;
6: **for** All clusters C_i **do**
7: Let $\mathcal{S}_i$ be the set of *extended* critical nodes of $\mathbb{T}_i$;
8: **for** $N \in \mathcal{S}_i$ **do**
9: Update $\mathbb{T} \leftarrow \mathbb{T} \cup \mathbb{T}_i[N]$.
10: **end for**
11: **end for**
12: Let $t = |\mathbb{T}|$, and $\mathbb{T} = \{\widetilde{\mathbb{T}}_1, \ldots, \widetilde{\mathbb{T}}_t\}$ be such that $|\widetilde{\mathbb{T}}_i| \leqslant |\widetilde{\mathbb{T}}_{i+1}|$ for all $1 \leqslant i < t$;
13: Initialise $\mathcal{T}_{\mathsf{MS}} = \widetilde{\mathbb{T}}_1$;
14: **for** $i = 2, \ldots, t$ **do**
15: Let $\mathcal{T}_{\mathsf{MS}}$ be the tree with $\mathcal{T}_{\mathsf{MS}}$ and $\widetilde{\mathbb{T}}_i$ as its two children;
16: **end for**
17: **Return:** $\mathcal{T}_{\mathsf{MS}}$

Lemma B.1. *Let $G = (V, E, w)$ be a graph, and $k > 1$ such that $\lambda_{k+1} > 0$ and $\lambda_k < \left(\frac{1}{270 \cdot c_0 \cdot (k+1)^6}\right)^2$, where c_0 is the constant in Lemma A.12. Then, the tree $\mathcal{T}_{\mathsf{MS}}$ returned by Algorithm 4 satisfies $\mathsf{COST}_G(\mathcal{T}_{\mathsf{MS}}) = O\left(k^{22}/\lambda_{k+1}^{10}\right) \cdot \mathsf{OPT}_G$.*

The remaining part of this subsection is to analyse Algorithm 4, and prove Lemma B.1. First of all, notice that, since $\{C_i\}_{i=1}^{\ell}$ is the output of our partitioning algorithm, it holds for every $u \in C_i$ for any $1 \leqslant i \leqslant \ell$ that

$(A1)$ $\Phi(C_i) = O(k^6\sqrt{\lambda_k})$;

$(A2)$ $\Phi_{G[C_i]} = \Omega(\lambda_{k+1}^2/k^4)$;

$(A3)$ $w(u, V \setminus C_i) \leqslant 6(k+1) \cdot \mathrm{vol}_{G[C_i]}(u)$.

Generalising Property (A3), it is easy to see that it holds for every extended critical node $N \in \mathcal{S}_i$ $(1 \leqslant i \leqslant \ell)$ that

$(A3^*)$ $w(N, V \setminus C_i) \leqslant 6(k+1) \cdot \mathrm{vol}_{G[C_i]}(N)$.

Moreover, with the parameter $\phi_{\mathrm{in}} = \Theta\left(\lambda_{k+1}/k^2\right)$, we have that $\Phi_{G[C_i]} = \Omega(\phi_{\mathrm{in}}^2)$ holds for any $1 \leqslant i \leqslant \ell$. Let $\mathbb{T}_i = \mathbb{T}_{\deg}(G[C_i])$ with the corresponding set of extended critical nodes $\mathcal{S}_i$ for all $1 \leqslant i \leqslant \ell$, and $\mathcal{S} = \bigcup_{i=1}^{\ell} \mathcal{S}_i$ be the set of all extended critical nodes. Now we group the edges of G into two categories: let E_1 be the set of edges in the induced subgraphs $G[C_i]$ for all $1 \leqslant i \leqslant \ell$, i.e.,

$$E_1 \triangleq \bigcup_{i=1}^{\ell} E\left[G[C_i]\right],$$

and E_2 be the remaining crossing edges. Therefore, we write the cost of our tree $\mathcal{T}_{\mathsf{MS}}$ as

$$\mathsf{COST}_G(\mathcal{T}_{\mathsf{MS}}) = \sum_{e\in E_1} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e) + \sum_{e\in E_2} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e), \tag{22}$$

and analyse each sum individually in Lemmas B.3 and B.4.

Our first result bounds the size of the parent of an extended critical node $N \in \mathcal{S}_i$ in $\mathcal{T}_{\mathsf{MS}}$, with respect to the size of its parent in the tree $\mathbb{T}_i$. This result will be extensively used when analysing the cost of the edges adjacent to N.

Lemma B.2. *It holds for every $1 \leqslant i \leqslant \ell$ and every extended critical node $N \in \mathcal{S}_i$ that*

$$\left|\mathrm{parent}_{\mathcal{T}_{\mathsf{MS}}}(N)\right| \leqslant 6k \cdot \left|\mathrm{parent}_{\mathbb{T}_i}(N)\right|.$$

Proof. Suppose the extended dense branch of $\mathbb{T}_i$ is $(A_0, \ldots, A_{i_{\max}-1})$ for some $i_{\max} \in \mathbb{Z}_{\geqslant 0}$, with B_j being the sibling of A_j and $A_{i_{\max}-1}$ having children $A_{i_{\max}}, B_{i_{\max}}$. Recall that the set of extended critical nodes is $\mathcal{S}_i = \{B_1, \ldots, B_{i_{\max}}, A_{i_{\max}}\}$. By construction of the degree-based tree, we know from Lemma A.6 that it holds for all $1 \leqslant j \leqslant i_{\max} - 1$ that $|A_j| = 2\cdot|A_{j+1}|$, which implies that $|B_{j+1}| = |A_{j+1}|$ and $|B_j| = 2 \cdot |B_{j+1}|$ for all $j \geqslant 2$. Thus, we conclude that for every interval $\left(2^{s-1}, 2^s\right]$, for some $s \in \mathbb{Z}_{\geqslant 0}$, there are at most 3 critical nodes[6] $N \in \mathcal{S}_i$ of size $|N| \in \left(2^{s-1}, 2^s\right]$. Now let us fix $N \in \mathcal{S}_i$. By construction, we have that

$$\left|\mathrm{parent}_{\mathbb{T}_i}(N)\right| \geqslant 2 \cdot |N|. \tag{23}$$

On the other hand, by the construction of $\mathcal{T}_{\mathsf{MS}}$ we have that

$$\begin{aligned}\left|\mathrm{parent}_{\mathcal{T}_{\mathsf{MS}}}(N)\right| &= \sum_{j=1}^{\ell} \sum_{\substack{M\in\mathcal{S}_j \\ |M|\leqslant|N|}} |M| \leqslant \sum_{j=1}^{\ell} \sum_{s=0}^{\lceil \log |N| \rceil} \sum_{\substack{M\in\mathcal{S}_j \\ 2^{s-1}<|M|\leqslant 2^s}} |M| \\ &\leqslant \sum_{j=1}^{\ell} \sum_{s=0}^{\lceil \log |N| \rceil} 3 \cdot 2^s \leqslant \sum_{j=1}^{k} 3 \cdot 2^{\lceil \log |N| \rceil + 1} \leqslant 12k \cdot |N|. \end{aligned} \tag{24}$$

By combining (23) and (24), we have that

$$\left|\mathrm{parent}_{\mathcal{T}_{\mathsf{MS}}}(N)\right| \leqslant 12k \cdot |N| \leqslant 6k \cdot \left|\mathrm{parent}_{\mathbb{T}_i}(N)\right|,$$

which proves the statement. □

Now we prove the two main technical lemmas of this subsection.

Lemma B.3. *It holds that $\sum_{e\in E_1} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e) = O\left(k/\phi_{\mathrm{in}}^8\right) \cdot \mathsf{OPT}_G$.*

Proof. Notice that

$$\sum_{e\in E_1} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e) = \sum_{i=1}^{\ell} \sum_{e\in E[G[C_i]]} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e).$$

We prove that, for every $1 \leqslant i \leqslant \ell$ and $e \in E[G[C_i]]$, the cost of e in $\mathcal{T}_{\mathsf{MS}}$ and the one in $\mathbb{T}_i$ differ by at most a factor of $O(k)$. Combining this with Theorem A.4 will prove the lemma.

To prove this $O(k)$-factor bound, we fix any $1 \leqslant i \leqslant \ell$ and let $\mathcal{S}_i$ be the set of extended critical nodes of C_i. As the nodes of $\mathcal{S}_i$ form a partition of the vertices of $G[C_i]$, any edge $e \in E[G[C_i]]$ satisfies exactly one of the following conditions: (i) e is inside a critical node; (ii) e is adjacent to a critical node. Formally, it holds that

$$\sum_{e\in E[G[C_i]]} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e) = \sum_{\substack{N\in\mathcal{S}_i \\ e\in E(N,N)}} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e) + \sum_{\substack{N\in\mathcal{S}_i \\ M\in\mathcal{S}_i\setminus\{N\} \\ \left|\mathrm{parent}_{\mathbb{T}_i}(M)\right|\leqslant\left|\mathrm{parent}_{\mathbb{T}_i}(N)\right|}} \sum_{e\in E(N,M)} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e)$$

[6] We remark that, in the worst case, all three nodes $B_1, B_{i_{\max}}$ and $A_{i_{\max}}$ could have size in $(2^{s-1}, 2^s]$.

We first examine the case in which the cost of e in both trees are the same, i.e., Case (i). Since we do not change the structure of the tree inside any critical node, it holds that

$$\sum_{\substack{N\in\mathcal{S}_i\\ e\in E(N,N)}} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e) = \sum_{\substack{N\in\mathcal{S}_i\\ e\in E(N,N)}} \mathsf{cost}_{\mathbb{T}_i}(e).$$

For Case (ii), the cost of any such edge increases by at most a factor of $O(k)$ due to Lemma B.2 and the construction of $\mathcal{T}_{\mathsf{MS}}$. Formally, let $N \in \mathcal{S}_i$ be an arbitrary extended critical node, and $M \in \mathcal{S}_i \setminus \{N\}$ be an extended critical node such that $\left|\mathrm{parent}_{\mathbb{T}_i}(M)\right| \leqslant \left|\mathrm{parent}_{\mathbb{T}_i}(N)\right|$. Firstly, notice that if $\mathrm{parent}_{\mathbb{T}_i}(N)$ is the root node of $\mathbb{T}_i$, then for any edge $e \in E(N,M)$ it holds that $\mathsf{cost}_{\mathbb{T}_i}(e) = w_e \cdot |\mathbb{T}_i|$. On the other hand, by the construction of $\mathcal{T}_{\mathsf{MS}}$, we know that $\mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e) \leqslant 6k \cdot w_e \cdot |\mathbb{T}_i|$, so we conclude that $\mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e) \leqslant 6k \cdot \mathsf{cost}_{\mathbb{T}_i}(e)$. Secondly, if $\mathrm{parent}_{\mathbb{T}_i}(N)$ is not the root node of $\mathbb{T}_i$ and since $\left|\mathrm{parent}_{\mathbb{T}_i}(M)\right| \leqslant \left|\mathrm{parent}_{\mathbb{T}_i}(N)\right|$, we know that $|M| \leqslant |N|$. Therefore it holds for any edge $e \in E(N,M)$ that

$$\mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e) = w_e \cdot \left|\mathrm{parent}_{\mathcal{T}_{\mathsf{MS}}}(N)\right| \leqslant 6k \cdot w_e \cdot \left|\mathrm{parent}_{\mathbb{T}_i}(N)\right| = 6k \cdot \mathsf{cost}_{\mathbb{T}_i}(e),$$

where the inequality follows by Lemma B.2. Combining the above facts, we have that

$$\sum_{e\in E_1} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e) \leqslant \sum_{i=1}^{\ell} 6k \cdot \sum_{e\in E[G[C_i]]} \mathsf{cost}_{\mathbb{T}_i}(e) \leqslant \sum_{i=1}^{\ell} 6k \cdot \mathsf{COST}_{G[C_i]}(\mathbb{T}_i). \tag{25}$$

On the other side, let $\mathcal{T}^*$ be any optimal HC tree of G with cost OPT_G, and it holds that

$$\mathsf{OPT}_G \geqslant \sum_{i=1}^{\ell} \mathsf{OPT}_{G[C_i]} = \sum_{i=1}^{\ell} \Omega\left(\Phi^4_{G[C_i]}\right) \cdot \mathsf{COST}(\mathbb{T}_i) = \sum_{i=1}^{\ell} \Omega\left(\phi^8_{\mathrm{in}}\right) \cdot \mathsf{COST}(\mathbb{T}_i), \tag{26}$$

where the last equality follows by Property (A2) of Lemma A.10 and Theorem A.4 applied to every $G[C_i]$. Finally, by combining (25) and (26) we have that

$$\sum_{e\in E_1} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e) = O(k/\phi^8_{\mathrm{in}}) \cdot \mathsf{OPT}_G,$$

which proves the lemma. □

Lemma B.4. *It holds that*

$$\sum_{e\in E_2} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e) = O\left(k^2/\phi^{10}_{\mathrm{in}}\right) \cdot \mathsf{OPT}_G.$$

Proof. For the edges $e \in E_2$, we bound the cost with the help of Lemma B.2 similar as before. Specifically, we have that

$$\begin{aligned}
\sum_{e\in E_2} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e) &\leqslant \sum_{i=1}^{\ell} \sum_{\substack{N\in\mathcal{S}_i\\ M\in\mathcal{S}\setminus\mathcal{S}_i\\ |M|\leqslant|N|}} \sum_{e\in E(N,M)} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e)\\
&\leqslant \sum_{i=1}^{\ell} \sum_{N\in\mathcal{S}_i} \left|\mathrm{parent}_{\mathcal{T}_{\mathsf{MS}}}(N)\right| \cdot w(N, V\setminus C_i)\\
&\leqslant \sum_{i=1}^{\ell} \sum_{N\in\mathcal{S}_i} 36k(k+1) \cdot \left|\mathrm{parent}_{\mathbb{T}_i}(N)\right| \cdot \mathrm{vol}_{G[C_i]}(N) \qquad (27)\\
&\leqslant 36k(k+1) \cdot \sum_{i=1}^{\ell} \frac{4}{\Phi_{G[C_i]}} \cdot \mathsf{COST}_{G[C_i]}(\mathbb{T}_i) \qquad (28)\\
&= O(k^2) \cdot \sum_{i=1}^{\ell} \frac{1}{\Phi^5_{G[C_i]}} \cdot \mathsf{OPT}_{G[C_i]} \qquad (29)\\
&= O(k^2/\phi^{10}_{\mathrm{in}}) \cdot \mathsf{OPT}_G,
\end{aligned}$$

where (27) follows from Property $(A3^*)$ of Lemma A.10 and Lemma B.2, (28) follows by Lemma A.3, and (29) follows by Theorem A.4 applied to every induced subgraph $G[C_i]$. □

Finally, we are ready to prove Lemma B.1.

Proof of Lemma B.1. Let $C_1, \dots C_\ell$ be the partitioned returned by the algorithm from Lemma A.10. Let $\mathcal{S} \triangleq \bigcup_{i=1}^{k} \mathcal{S}_i$ be the set of all critical nodes, and $\mathcal{T}_{\mathrm{CC}}$ be the "caterpillar" style tree on the critical nodes according to an increasing order of their sizes. We have that

$$\begin{aligned}\mathsf{COST}_G(\mathcal{T}_{\mathsf{MS}}) &= \sum_{e\in E_1} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e) + \sum_{e\in E_2} \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e)\\ &= O(k/\phi_{\mathrm{in}}^8)\cdot \mathsf{OPT}_G + O(k^2/\phi_{\mathrm{in}}^{10})\cdot \mathsf{OPT}_G = O(k^2/\phi_{\mathrm{in}}^{10})\cdot \mathsf{OPT}_G = O(k^{22}/\lambda_{k+1}^{10})\cdot \mathsf{OPT}_G,\end{aligned}$$

where the second equality follows by Lemmas B.3 and B.4, and the last equality follows by the definition of ϕ_{in}. □

B.2. Proof of Lemmas 5.2 and 5.3

Now we prove Lemma 5.2 and Lemma 5.3, which allow us to approximate the cost of $\mathcal{T}_{\mathsf{MS}}$. To sketch the main proof idea for Lemma 5.3, we construct a tree $\mathcal{T}''_{\mathsf{MS}}$ and upper bound its weighted Dasgupta cost on the bucketing B, i.e., $\mathsf{WCOST}_{G/\mathcal{B}}(\mathcal{T}''_{\mathsf{MS}})$, with respect to OPT_G. To construct $\mathcal{T}''_{\mathsf{MS}}$, we start with the tree $\mathcal{T}_{\mathsf{MS}}$ on G, whose upper bound with respect to OPT_G is known because of Lemma B.1. We then transform $\mathcal{T}_{\mathsf{MS}}$ into $\mathcal{T}''_{\mathsf{MS}}$ in two steps, such that there is only a small induced cost for both transformations. This allows us to upper bound $\mathsf{COST}_G(\mathcal{T}''_{\mathsf{MS}})$ with respect to $\mathsf{COST}_G(\mathcal{T}_{\mathsf{MS}})$. Crucially, we can approximate the tree $\mathcal{T}_{\mathsf{MS}}$ with Algorithm 1 *without* explicitly constructing tree $\mathcal{T}_{\mathsf{MS}}$. The proof for Lemma 5.2 follows from the proof for Lemma 5.3.

In order to analyse the constructed trees $\mathcal{T}'_{\mathsf{MS}}$ and $\mathcal{T}''_{\mathsf{MS}}$, we analyse the properties of the buckets in the context of the entire graph G, and not just on every set P_i returned by spectral clustering. We denote by $\mathcal{B}$ the set containing all the buckets obtained throughout all sets P_i, i.e.,

$$\mathcal{B} \triangleq \bigcup_{i=1}^{k} \mathcal{B}_i;$$

notice that $\mathcal{B}$ is a partition of V. We use

$$\kappa \triangleq |\mathcal{B}|$$

to denote the total number of buckets. For convenience, we label the buckets $\mathcal{B} = \{B_1, \dots, B_\kappa\}$ arbitrarily, and in general we always refer to a bucket $B_j \in \mathcal{B}$ with the subscript j.

Next, we also analyse the properties of the extended critical nodes in the context of the entire graph G, not just from every set $\mathcal{S}_i$. Therefore, let

$$\mathcal{S} = \bigcup_{i=1}^{\ell} \mathcal{S}_i$$

be the set of all extended critical nodes, and we label them as $\mathcal{S} = \{N_1, \dots N_{k_1}\}$ in the order they appear in if one would travel upwards starting from the bottom of the caterpillar tree $\mathcal{T}_{\mathsf{MS}}$. In general, we always refer to an extended critical node $N_i \in \mathcal{S}$ with the subscript i. We use $C(N_i)$ to denote the original cluster that N_i belongs to, i.e, the cluster $C(N_i)$ is the C_s such that $N_i \in \mathcal{S}_s$. We also use $\mathbb{T}_{\deg}(N_i)$ to denote the tree $\mathbb{T}_{\deg}$ that N_i originally corresponds to. To avoid confusion, we remind the reader that $\mathbb{T}_i$ is the degree-based tree constructed on the induced graph $G[C_i]$, i.e., $\mathbb{T}_i = \mathbb{T}_{\deg}(G[C_i])$.

Next, we define

$$X_j \triangleq \{N_i \in \mathcal{S} | N_i \cap B_j \neq \emptyset\}$$

as the set of critical nodes that have non-empty intersection with bucket $B_j \in \mathcal{B}$. Note that the subscript of X_j matches the one of the corresponding bucket B_j. We then define the *anchor* (critical) node $N_{i'} \in \mathcal{S}$ of the set X_j as the largest critical node inside X_j. i.e., we have that

$$N_{i'} \triangleq \mathrm{argmax}_{N_{i''}\in X_j} |N_{i''}|,$$

where ties are broken by choosing $N_{i'}$ to be the critical node highest up in $\mathcal{T}_{\mathsf{MS}}$. These anchor nodes corresponding to the sets X_j – and therefore the buckets $B_j \in \mathcal{B}$ – are the key in our proof of the approximation factor of our algorithm; they

determine the locations in the tree $\mathcal{T}_{\mathsf{MS}}$ where we group together the buckets $B_j \in \mathcal{B}$. It is also important to note that for every X_j its anchor node is determined *uniquely*. On the other hand, any $N_i \in \mathcal{S}$ might be the anchor node for multiple X_j's.

Now we're ready to describe the two transformations.

Step 1: Splitting the Critical Nodes. In this first step, we transform the tree $\mathcal{T}_{\mathsf{MS}}$ such that all the sets $N_i \cap B_j$ for $1 \leqslant i \leqslant k_1$ and $1 \leqslant j \leqslant \kappa$ are separated and restructured to form a caterpillar tree. This is done by splitting every N_i into $N_i \cap B_j$ for all $1 \leqslant j \leqslant \kappa$ and appending them next to each other arbitrarily in the caterpillar tree in the original location of N_i in $\mathcal{T}_{\mathsf{MS}}$.

More formally, we consider the partition of every $N_i \in \mathcal{S}$ into $\{N_i \cap B_j\}_{j=1}^{\kappa}$. Then, for every internal node $\mathrm{parent}(N_i)$ in the tree $\mathcal{T}_{\mathsf{MS}}$, we create new internal nodes $\mathrm{parent}(N_i \cap B_{j'})$ for every $N_i \cap B_{j'} \in \{N_i \cap B_j\}_{j=1}^{\kappa}$, and let one of the children of each of these be a new internal node corresponding to $N_i \cap B_{j'}$, and let the other child be another node $\mathrm{parent}(N_i \cap B_{j''})$ for $j' \neq j''$. Moreover, we ensure that the ordering (starting from the bottom) of the new nodes $\mathrm{parent}(N_i \cap B_j)$ along the back of the caterpillar tree will be such that the first all the ones corresponding to N_1 will appear, then N_2 etc., up until N_{k_1}.

We call the resulting tree $\mathcal{T}'_{\mathsf{MS}}$, whose cost is bounded in the lemma below.

Lemma B.5. *It holds that*

$$\mathsf{COST}_G\left(\mathcal{T}'_{\mathsf{MS}}\right) \leqslant \mathsf{COST}_G\left(\mathcal{T}_{\mathsf{MS}}\right) + O(k/\phi_{\mathrm{in}}^{10}) \cdot \mathsf{OPT}_G.$$

Proof. First of all, notice that for any edge crossing critical nodes, i.e., $e \in E(N_i, N_{i'})$ for $i \neq i'$, we have that

$$\mathsf{cost}_{\mathcal{T}'_{\mathsf{MS}}}(e) \leqslant \mathsf{cost}_{\mathcal{T}_{\mathsf{MS}}}(e).$$

This is because splitting up the critical nodes cannot increase the size of the lowest common ancestor of any crossing edge between critical nodes, due to the caterpillar tree construction. So we don't need to consider the increase in the cost of crossing edges between critical nodes.

Next, we study the edges inside critical nodes $e \in E(N_i, N_i)$ for any i. These edges' cost could be increased compared with the one in $\mathcal{T}_{\mathsf{MS}}$, since they can span across nodes $N_i \cap B_j$. However, the increase in size of their lowest common ancestor is by construction *at most* $|\mathrm{parent}_{\mathcal{T}_{\mathsf{MS}}}(N_i)|$, and therefore the total cost of this transformation can be upper bounded as

$$\begin{aligned}
\sum_{i=1}^{k_1} \sum_{e \in E(N_i, N_i)} \mathsf{cost}_{\mathcal{T}'_{\mathsf{MS}}}(e) \leqslant & \sum_{i=1}^{k_1} \sum_{e \in E(N_i, N_i)} w(e) \cdot \left|\mathrm{parent}_{\mathcal{T}_{\mathsf{MS}}}(N_i)\right| \\
\leqslant & \sum_{i=1}^{k_1} 6k \cdot \left|\mathrm{parent}_{\mathbb{T}_{\deg}(N_i)}(N_i)\right| \cdot \mathrm{vol}_{G[C(N_i)]}(N_i) && (30) \\
\leqslant & 6k \cdot \sum_{s=1}^{\ell} \frac{4}{\Phi_{G[C_s]}} \cdot \mathsf{COST}_{G[C_s]}(\mathbb{T}_s) && (31) \\
= & O(k) \cdot \sum_{s=1}^{\ell} \frac{1}{\Phi^5_{G[C_s]}} \cdot \mathsf{OPT}_{G[C_s]} && (32) \\
= & O(k/\phi_{\mathrm{in}}^{10}) \cdot \mathsf{OPT}_G,
\end{aligned}$$

where (30) follows from Lemma B.2, (31) follows from Lemma A.3, and (32) follows from Theorem A.4 applied to every induced subgraph $G[C_i]$. This proves the statement. □

Step 2: Grouping Degree Buckets. In this second step, we move around the sets $N_i \cap B_j$, such that the buckets $B_1, \ldots, B_\kappa$ are separated after this step.

We do this by traversing the tree $\mathcal{T}'_{\mathsf{MS}}$ from the bottom to the top, and visiting every node $\mathrm{parent}(N_i \cap B_j)$ up along the "back" of the caterpillar construction. This means we first visit all the nodes $\mathrm{parent}(N_i \cap B_j)$ corresponding to N_1, then N_2 etc. up to N_{k_1}. When visiting $\mathrm{parent}(N_i \cap B_j)$ we check whether N_i is an anchor node of X_j. If not, we continue travelling up. *If it is*, we transform the tree $\mathcal{T}'_{\mathsf{MS}}$ by moving up all the internal nodes $N_{i'} \cap B_j$ for $N_{i'} \in X_j \setminus N_i$ and placing them in the same subtree as the internal node corresponding to $N_i \cap B_j$. We do so by creating a new root node R_{B_j}, and we loop through each $N_{i'} \cap B_j$ for $N_{i'} \in X_j$ in an arbitrary order. In the first iteration, we set the children of R_{B_j} to be

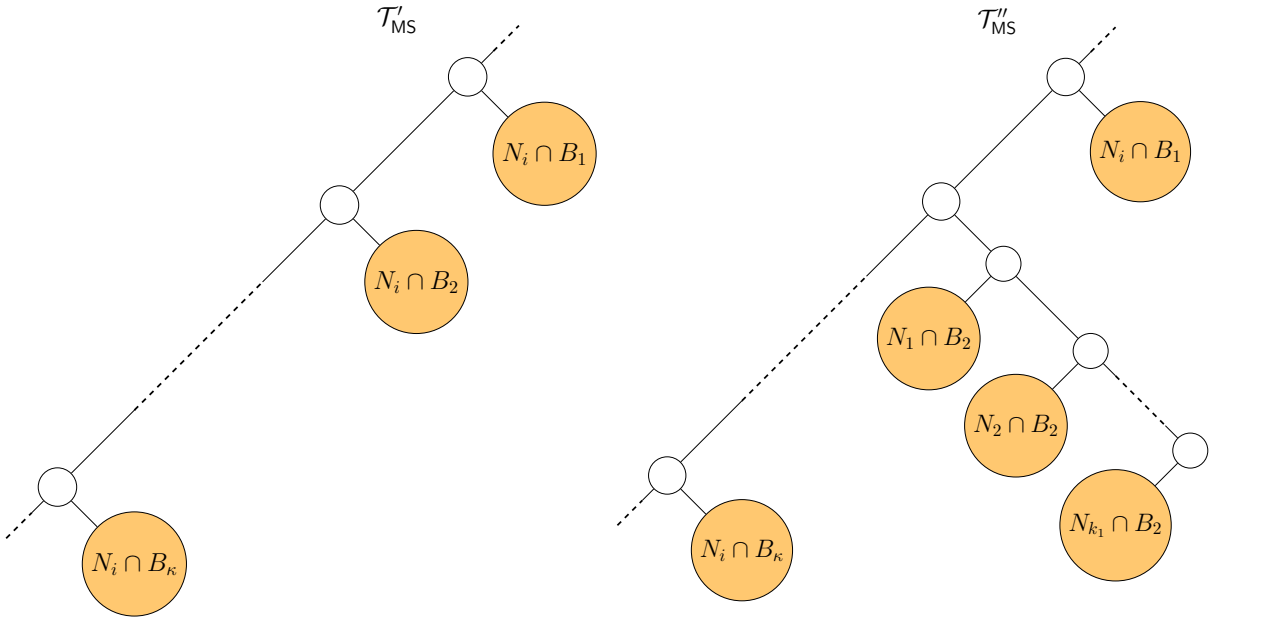

Figure 4. Visualisation of a part of the transformation described in step 2, where the buckets are grouped. On the left we visualise a section of the tree $\mathcal{T}'_{\mathsf{MS}}$, corresponding to a single critical node N_i. For illustration, we assume here that N_i is the anchor node for X_2. This means that all $N_{i'} \cap B_2$ for $N_{i'} \in X_2 \setminus N_j$ are moved up into the same subtree, as illustrated on the right in the new tree $\mathcal{T}''_{\mathsf{MS}}$.

$N_{i'} \cap B_j$, and a newly created node denoted by z. We iteratively do this, setting the two children of z in the same way as for the root node, such that we construct a caterpillar tree on the nodes $N_{i'} \cap B_j$ for $N_{i'} \in X_j$. Finally, after this construction, we replace the child node $N_i \cap B_j$ of $\mathrm{parent}(N_i \cap B_j)$ with the root node R_{B_j} of our newly constructed caterpillar tree.

Note that every $\mathrm{parent}(N_{i'} \cap B_j)$ for $N_{i'} \in X_j \setminus N_i$ no longer has a corresponding $N_{i'} \cap B_j$ as a child node, since it has been placed in the new caterpillar tree. Therefore, as a final step, we remove $\mathrm{parent}(N_{i'} \cap B_j)$ from the tree $\mathcal{T}'_{\mathsf{MS}}$, and appropriately update the child and parent node directly below and above $\mathrm{parent}(N_{i'} \cap B_j)$ to ensure that the tree stays connected.

After travelling up the entire tree, this transformation ensures that all the vertices in B_j are placed in the same subtree, since $\bigcup_{N_{i'} \in X_j} N_{i'} \cap B_j = B_j$. Crucially, because N_i is an anchor node, we know by the construction of the tree $\mathcal{T}_{\mathsf{MS}}$ that every $N_{i'} \in X_j \setminus N_i$ is placed lower in the tree $\mathcal{T}'_{\mathsf{MS}}$, and hence we do not have to consider the induced cost of moving any critical node higher up in the tree down to N_i. To illustrate, We visualise (part of) the transformation in Figure 4. We call the resulting tree $\mathcal{T}''_{\mathsf{MS}}$, and we bound its cost with the following lemma.

Lemma B.6. *It holds that*

$$\mathsf{COST}_G\left(\mathcal{T}''_{\mathsf{MS}}\right) \leqslant \mathsf{COST}_G\left(\mathcal{T}'_{\mathsf{MS}}\right) + O(\beta \cdot k^3/\phi_{\mathrm{in}}^{10}) \cdot \mathsf{OPT}_G.$$

Proof. We bound the cost of the transformation by looking at the total induced cost incurred at every $N_i \in \mathcal{S}$ that is an anchor node. First of all, notice that for all nodes $N_i \cap B_j$ in $\mathcal{T}'_{\mathsf{MS}}$ we have that

$$|\mathrm{parent}_{\mathcal{T}'_{\mathsf{MS}}}(N_i \cap B_j)| \leqslant |\mathrm{parent}_{\mathcal{T}_{\mathsf{MS}}}(N_i)| \leqslant 6k \cdot |\mathrm{parent}_{\mathbb{T}_{\deg}(N_i)}(N_i)|, \tag{33}$$

where the first inequality holds because of the construction of $\mathcal{T}'_{\mathsf{MS}}$ and the second inequality follows by Lemma B.2. Recall that $\mathbb{T}_{\deg}(N_i)$ is the tree $\mathbb{T}_{\deg}$ that N_i originally corresponds to.

Now let us fix some N_i that is also an anchor node, and let $\mathcal{X}_i$ be the set such that $X_j \in \mathcal{X}_i$ shares N_i as their anchor node, i.e., $N_i = \mathrm{argmax}_{N_{i'} \in X_j} |N_{i'}|$ for $X_j \in \mathcal{X}_i$. The only edges whose cost can increase when moving up $N_{i'} \cap B_j$ for some $N_{i'} \in X_j$ will be the edges with one endpoint in $N_{i'} \cap B_j$, since moving them up potentially increases their lowest common ancestor to be $|\mathrm{parent}_{\mathcal{T}'_{\mathsf{MS}}}(N_i \cap B_j)|$. Note that because we carefully choose to perform the transformations at the *anchor* node, we don't introduce any new vertices from higher up in the tree which could potentially change the size of the lowest common ancestor.

Therefore, we can bound the cost of moving up all $N_{i'} \cap B_j$ for $N_{i'} \in X_j$ and $X_j \in \mathcal{X}_i$ as

$$\begin{aligned}
&\sum_{X_j \in \mathcal{X}_i} \left(|\mathrm{parent}_{\mathcal{T}'_{\mathsf{MS}}}(N_i \cap B_j)| \sum_{N_{i'} \in X_j} \mathrm{vol}(N_{i'} \cap B_j) \right) \\
\leqslant & 6k \cdot |\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(N_i)| \cdot \sum_{X_j \in \mathcal{X}_i} \sum_{N_{i'} \in X_j} \mathrm{vol}(N_{i'} \cap B_j) && (34) \\
\leqslant & 6k \cdot |\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(N_i)| \cdot \sum_{X_j \in \mathcal{X}_i} \sum_{N_{i'} \in X_j} |N_{i'} \cap B_j| \cdot \Delta_G(N_{i'} \cap B_j) \\
\leqslant & 6k \cdot |\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(N_i)| \sum_{X_j \in \mathcal{X}_i} \sum_{N_{i'} \in X_j} |N_{i'} \cap B_j| \cdot \beta \cdot \Delta_G(N_i \cap B_j) && (35) \\
\leqslant & 6k \cdot |\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(N_i)| \cdot \sum_{X_j \in \mathcal{X}_i} \sum_{N_{i'} \in X_j} |N_{i'} \cap B_j| \cdot \beta \cdot \Delta_G(N_i) \\
\leqslant & 6k \cdot \beta \cdot |\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(N_i)| \cdot 14k \cdot \Delta_{G[C(N_i)]}(N_i) \cdot \sum_{X_j \in \mathcal{X}_i} \sum_{N_{i'} \in X_j} |N_{i'}| && (36) \\
\leqslant & O(k^3 \cdot \beta) \cdot |\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(N_i)| \cdot \Delta_{G[C(N_i)]}(N_i) \cdot |\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(N_i)| && (37) \\
\leqslant & O(k^3 \cdot \beta) \cdot |\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(N_i)| \cdot \delta_{G[C(N_i)]}(\mathrm{sib}_{\mathbb{T}_{\deg(N_i)}}(N_i)) \cdot |\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(N_i)| && (38) \\
\leqslant & O(k^3 \cdot \beta) \cdot |\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(\mathrm{sib}_{\mathbb{T}_{\deg(N_i)}}(N_i))| \cdot \mathrm{vol}_{G[C(N_i)]}\left(\mathrm{sib}_{\mathbb{T}_{\deg(N_i)}}(N_i)\right). && (39)
\end{aligned}$$

Here, (34) follows by (33), (35) follows by the fact that all degrees d_u of $u \in N_{i'} \cap B_j$ for $N_{i'} \in X_j$ lie within a factor of β of each other. Since we denote by $C(N_i)$ the cluster to which N_i originally corresponds, (36) holds because of Property (A3) of Lemma A.10, (37) holds because we only move the critical nodes $N_{i'}$ that are below N_i in the caterpillar construction, so the sum of their sizes is at most $|\mathrm{parent}_{\mathcal{T}_{\mathsf{MS}}}(N_i)|$, which we upper bound with (33). Moreover, (38) follows by the fact that the minimum degree in $G[C(N_i)]$ of a sibling node of N_i (Definition A.7) is at least the maximum degree of N_i in $G[C(N_i)]$, (39) holds because

$$|\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(N_i)| \leqslant 2 \cdot |\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(\mathrm{sib}_{\mathbb{T}_{\deg(N_i)}}(N_i))|$$

and

$$\delta_{G[C(N_i)]}(\mathrm{sib}_{\mathbb{T}_{\deg(N_i)}}(N_i)) \cdot |\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(N_i)| \leqslant 4 \cdot \mathrm{vol}_{G[C(N_i)]}\left(\mathrm{sib}_{\mathbb{T}_{\deg(N_i)}}(N_i)\right).$$

The bound above holds as long as N_i has a sibling node, and it remains to examine the case in which N_i doesn't have a sibling node. Recall that the set of extended critical nodes of a cluster C_s returned by Lemma B.1 is $\mathcal{S}_s \triangleq \{B_1, \ldots, B_{s_{\max}}, A_{s_{\max}}\}$. The only critical node *without* a sibling node is $A_{s_{\max}}$, and we know by construction that $|A_{s_{\max}}| \leqslant 2$. Because $\mathcal{T}_{\mathsf{MS}}$ is a caterpillar tree based on the sizes of the nodes, $A_{s_{\max}}$ is placed as one of the critical nodes furthest down the tree. Given that there are at most three critical nodes of size at most two in every C_s, $A_{s_{\max}}$ must be one of the last 3ℓ extended critical nodes in the caterpillar tree.

Suppose one of these $A_{s_{\max}}$ from some $\mathcal{S}_s$ is an anchor node. The only nodes in the tree $\mathcal{T}'_{\mathsf{MS}}$ that can be moved up to this node are other nodes of size at most two, of which there are at most 3ℓ (three for every partition C_s). Let $N_q \triangleq \operatorname{argmax}_{\substack{N_{i'} \in \mathcal{S} \\ |N_{i'}| \leqslant 2}} \mathrm{vol}(N_{i'})$ be the critical node of size at most two with the largest volume. Then, all the transformations (at most $3k$) occurring in the lower part of the tree $\mathcal{T}_{\mathsf{MS}}$ can be upper bounded by

$$\begin{aligned}
&3k \left(3k \cdot 2 \cdot \sum_{\substack{i' \\ |N_{i'}| \leqslant 2}} \mathrm{vol}_G(N_{i'}) \right) \\
=&O(k^3) \cdot \mathrm{vol}_G(N_q) \\
=&O(k^4) \cdot \mathrm{vol}_{G[C(N_q)]}(N_q) \\
=&O(\beta \cdot k^3) \cdot |\mathrm{parent}_{\mathbb{T}_{\deg(N_q)}}(N_q)| \cdot \mathrm{vol}_{G[C(N_q)]}(N_q), && (40)
\end{aligned}$$

where the first equality holds as N_q is the largest volume extended critical node of size at most two, the second one holds by Property (A3) of Lemma A.10, and the last one holds because $|\mathrm{parent}_{\mathbb{T}_{\deg(N_q)}}(N_q)| \leqslant 4$ and the fact that $k \leqslant 2^{k(\gamma+1)} = \beta$ for $k \geqslant 1$ and $\gamma > 0$.

Finally, to upper bound the total cost of the transformation, we assume for the upper bound that every extended critical node is an anchor node, and we split the sum over the induced cost of all the anchor nodes that have critical siblings (39), and the transformation in the bottom of the tree corresponding to critical nodes of size at most 2, which don't have critical siblings (40). Hence, the total cost of the transformation is at most

$$
\begin{aligned}
&O(\beta \cdot k^3) \cdot \sum_{\substack{i=1 \\ |N_i|>2}}^{k_1} |\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(\mathrm{sib}_{\mathbb{T}_{\deg(N_i)}}(N_i))| \cdot \mathrm{vol}_{G[C(N_i)]}\left(\mathrm{sib}_{\mathbb{T}_{\deg(N_i)}}(N_i)\right) + \\
&\qquad O(\beta \cdot k^3) \cdot |\mathrm{parent}_{\mathbb{T}_{\deg(N_q)}}(N_q)| \cdot \mathrm{vol}_{G[C(N_q)]}(N_q) \\
\leqslant &O(\beta \cdot k^3) \cdot \sum_{i=1}^{k_1} \left|\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(N_i)\right| \cdot \mathrm{vol}_{G[C(N_i)]}(N_i) && (41) \\
\leqslant &O(\beta \cdot k^3) \cdot \sum_{s=1}^{\ell} \frac{4}{\Phi_{G[C_s]}} \cdot \mathsf{COST}_{G[C_s]}(\mathbb{T}_s) && (42) \\
= &O(\beta \cdot k^3) \cdot \sum_{s=1}^{\ell} \frac{1}{\Phi^5_{G[C_s]}} \cdot \mathsf{OPT}_{G[C_s]} && (43) \\
= &O(\beta \cdot k^3/\phi_{\mathrm{in}}^{10}) \cdot \mathsf{OPT}_G.
\end{aligned}
$$

Here, (41) holds by the fact that the first sum of (41) is over $|\mathrm{parent}_{\mathbb{T}_{\deg(N_i)}}(N_i)| \cdot \mathrm{vol}_{G[C(N_i)]}(N_i)$ for all $1 \leqslant i \leqslant k_1$ *except* for the extended critical nodes that are not the sibling node of any critical node. Hence, we upper bound the sum by including these terms, and relabeling the indices. Moreover, (42) follows by Lemma A.3, and (43) follows by Theorem A.4 applied to every induced subgraph $G[C_s]$. This proves the statement. □

Proof of Lemma 5.3. Since the tree $\mathcal{T}''_{\mathsf{MS}}$ separates all the individual buckets $B \in \mathcal{B}$ by construction, we can upper bound the cost of $\mathsf{WOPT}_{G/\mathcal{B}}$ with respect to the cost of $\mathcal{T}''_{\mathsf{MS}}$ and have that

$$
\mathsf{WOPT}_{G/B} \leqslant \mathsf{COST}_G\left(\mathcal{T}_{\mathsf{MS}}\right) + O(k/\phi_{\mathrm{in}}^{10}) \cdot \mathsf{OPT}_G + O(\beta \cdot k^3/\phi_{\mathrm{in}}^{10}) \cdot \mathsf{OPT}_G = O\left(\beta \cdot k^{23}/\lambda_{k+1}^{10}\right) \cdot \mathsf{OPT}_G,
$$

where the first inequality holds by Lemmas B.5 and B.6, and the second one holds by Lemma B.1 and the fact $\phi_{\mathrm{in}} = \Theta(\lambda_{k+1}/k^2)$. □

Proof of Lemma 5.2. We bound the cost of the internal edges in $G[B_j]$ for $1 \leqslant j \leqslant \kappa$ using the trivial upper bound for every B_j, and have that

$$
\begin{aligned}
\sum_{i=1}^{k} \sum_{B \in \mathcal{B}_i} \mathsf{COST}_{G[B]}\left(\mathcal{T}_B\right) &\leqslant \sum_{j=1}^{\kappa} |B_j| \cdot \mathrm{vol}(B_j) = \sum_{i'=1}^{k_1} \sum_{X_j \in \mathcal{X}_{i'}} |B_j| \cdot \sum_{N_{i''} \in X_j} \mathrm{vol}(N_{i''} \cap B_j) \\
&\leqslant \sum_{i'=1}^{k_1} \sum_{X_j \in \mathcal{X}_{i'}} |\mathrm{parent}_{\mathcal{T}'_{\mathsf{MS}}}(N_{i'} \cap B_j)| \cdot \sum_{N_{i''} \in X_j} \mathrm{vol}(N_{i''} \cap B_j)
\end{aligned}
$$

where the last inequality holds by construction of the tree $\mathcal{T}'_{\mathsf{MS}}$ and the property of anchor nodes. By repeating the same calculation as the one in proving Lemma 5.3, we have that

$$
\begin{aligned}
\sum_{i=1}^{k} \sum_{B \in \mathcal{B}_i} \mathsf{COST}_{G[B]}\left(\mathcal{T}_B\right) &\leqslant \sum_{i'=1}^{k_1} \sum_{X_j \in \mathcal{X}_{i'}} |\mathrm{parent}_{\mathcal{T}'_{\mathsf{MS}}}(N_{i'} \cap B_j)| \cdot \sum_{N_{i''} \in X_j} \mathrm{vol}(N_{i''} \cap B_j) \\
&= O\left(\beta \cdot k^{23}/\lambda_{k+1}^{10}\right) \cdot \mathsf{OPT}_G,
\end{aligned}
$$

which proves the statement. □

B.3. Proof of Theorem 5.1

Now we prove the approximation and running time guarantee of Algorithm 1. In the following Lemma we analyse the approximation and running time guarantee of WRSC.

Lemma B.7. *Let $w_{\max}/w_{\min} = O(n^{\gamma})$ for a constant $\gamma > 1$, and $\mathcal{B}$ the set of all buckets constructed inside every $P_1, \dots P_k$. Then, given a contracted graph $H = G/\mathcal{B}$ as input the WRSC algorithm runs in $\widetilde{O}(n)$ time and returns an HC tree $\mathcal{T}$, such that*

$$\mathsf{WCOST}(\mathcal{T}) = O(1) \cdot \mathsf{WOPT}_{G/\mathcal{B}}.$$

Proof. We denote by $\mathcal{B}$ the set containing all the buckets obtained throughout all sets P_i, i.e.,

$$\mathcal{B} \triangleq \bigcup_{i=1}^{k} \mathcal{B}_i,$$

and it holds that $|\mathcal{B}| = \kappa \leqslant k \cdot \log_{\beta}(\Delta_G/\delta_G)$. Recall that our choice of γ and β satisfies

$$\frac{w_{\max}}{w_{\min}} = O(n^{\gamma})$$

and $\beta = 2^{k(\gamma+1)}$. Hence, it holds that

$$\frac{\Delta_G}{\delta_G} = O\left(n^{\gamma+1}\right),$$

and the total number of buckets in $\mathcal{B}$ is upper bounded by

$$k \cdot \max\{1, \log_{\beta} n^{\gamma+1}\} \leqslant k + \frac{k \cdot (\gamma+1)}{\log \beta} \cdot \log n = k + \log n.$$

Hence, there are at most $k + \log n$ vertices in $G/\mathcal{B}$. To compute the optimal weighted sparsest cut in $H = G/\mathcal{B}$, we compute $\mathsf{sparsity}_H(S)$ for every subset $S \subset V(H)$, the total number of which is upper bounded by $2^{k+\log n} = O\left(2^k \cdot n\right)$. Since both $w(S, V(H) \setminus S)$ and $\mathrm{vol}_H(S)$ can be computed in $\widetilde{O}(1)$ time, the sparsest cut of $G/\mathcal{B}$ can be computed in $\widetilde{O}(n)$ time given that k is a constant. Combining this with the master theorem proves the time complexity of the WRSC algorithm.

The approximation guarantee for the algorithm follows from Lemma 3.2 and the fact that we compute the optimal weighted sparsest cut; as such we have $\alpha = 1$ □

Next, we study the time complexity of Algorithm 1. We first show that the cost $\mathsf{COST}_G(\mathcal{T})$ can be computed in nearly-linear time, assuming that the internal nodes of our constructed tree $\mathcal{T}$ have several useful attributes.

Lemma B.8. *Let $\mathcal{T}_G$ be an HC tree of a graph $G = (V, E, w)$, with depth D. Assume that every internal node N of $\mathcal{T}_G$ has (i) a pointer* parent *that points to its parent node, (ii) an attribute* size*, which specifies the number of leaves in the subtree $\mathcal{T}_G(G[N])$, and (iii) an attribute* depth *that specifies the depth of $\mathcal{T}_G(G[N])$. Then, $\mathsf{COST}_G(\mathcal{T}_G)$ can be computed in $O(m \cdot D)$ time.*

Proof. Since G has m edges, it suffices to show that we can compute the value $\mathsf{cost}_G(e)$ for every edge $e = \{u, v\} \in E$ in $O(D)$ time. Let us fix an arbitrary edge $e = \{u, v\}$, and let N_u and N_v be the leaf nodes of $\mathcal{T}_G$ corresponding to u and v respectively. Without loss of generality, we assume that $N_u(\mathsf{depth}) \geqslant N_v(\mathsf{depth})$. We successively travel from N_u up the tree via the parent pointers until we reach a node N'_u, such that $N'_u(\mathsf{depth}) = N_v(\mathsf{depth})$. After that, we find the lowest common ancestor of u and v by simultaneously travelling up one node at a time, starting from both N'_u and N_v, until we reach the same internal node N_r. Once this condition is satisfied, we can readily compute $\mathsf{cost}_G(\{u, v\}) = w_e \cdot N_r(\mathsf{size})$. The overall running time for computing $\mathsf{cost}_G(\{u, v\})$ is $O(D)$, and therefore the statement holds. □

By Lemma B.8 we know that the time complexity of constructing $\mathcal{T}$ is linear in the depth of $\mathcal{T}$. The following lemma shows that the depth of $\mathcal{T}$ is $O(\log n)$, and the entire $\mathcal{T}$ can be constructed in nearly-linear time.

Lemma B.9. *Algorithm 1 runs in $\widetilde{O}(m)$ time, and constructs the tree $\mathcal{T}$ of depth $O(\log n)$. Moreover, every internal node $N \in \mathcal{T}$ has a pointer* parent*, and every node $T \in \mathcal{T}$ has attributes* size *and* depth*.*

Proof. We follow the execution of Algorithm 1 to analyse its time complexity, and the depth of its constructed $\mathcal{T}$. Line 2 of Algorithm 1 applies spectral clustering to obtain the clusters $\{P_i\}_{i=1}^k$, and the running time of this step is $\widetilde{O}(m)$ (Peng et al., 2017). Since ordering the vertices of every P_i with respect to their degrees takes $O(|P_i| \log |P_i|)$ time, the total time complexity of sorting the vertices for $\{P_i\}$, finding $\{u^{(i)}\}$, as well as constructing $\{\mathcal{B}_i\}$ (Lines 6–8) is $O\left(\sum_{i=1}^k |P_i| \log |P_i|\right) = O(n \log n)$.

Now let $B_j \in \mathcal{B}$ be an arbitrary bucket, and we study the construction of $\mathcal{T}_{B_j}$. We assume that B_j has vertices $u_1, \ldots, u_{n_j}$ sorted in increasing order of their degrees, where $n_j = |B_j|$. We construct the balanced tree $\mathcal{T}_{B_j}$ recursively as follows:

1. Initialise a root node N_0, such that $N_0(\mathsf{size}) = n_j$, and $N_0(\mathsf{parent}) = \emptyset$;
2. Create two nodes N_1 and N_2 corresponding to the vertices $u_1, \ldots, u_{\lceil n_j/2 \rceil}$ an $u_{\lceil n_j/2 \rceil + 1}, \ldots u_{n_j}$ respectively. Set $N_1(\mathsf{parent}) = N_0$ and $N_2(\mathsf{parent}) = N_0$;
3. Recursively construct the subtrees rooted at N_1 and N_2 on the corresponding sets of vertices, until we reach the leaves.

By construction, $\mathcal{T}_{B_j}$ is a balanced tree and has $O(n_j)$ internal nodes. Hence, the depth of $\mathcal{T}_{B_j}$ is $O(\log n_j) = O(\log n)$. Moreover, it's straightforward to see that this step (Lines 9–12) takes $O(n \log n)$ time for constructing each $\mathcal{T}_{B_j}$.

For constructing the contracted graph H from $\mathcal{B}$, notice that every edge of G only contributes to exactly one of the edge weights in H; as such constructing H (Line 15) takes $O(m)$ time. Finally, to complete the construction of $\mathcal{T}$, we run the WRSC algorithm on H (Line 16), and assume that we know how the vertices are split at each iteration of the recursive weighted sparsest cut algorithm. We construct the tree $\mathcal{T}$ recursively as follows:

1. Initialise a root node R_0, such that $R_0(\mathsf{size}) = n$, and $R_0(\mathsf{parent}) = \emptyset$;
2. Create two nodes R_1 and R_2 corresponding to the sets A and B, where (A, B) is the first split of the WRSC algorithm. Set $R_1(\mathsf{parent}) = R_0$ and $R_2(\mathsf{parent}) = R_0$;
3. Recursively construct the subtrees rooted at R_1 and R_2 on the corresponding sets of vertices and split (A, B) returned by the WRSC algorithm, until we reach the leaves corresponding to $B_1, \ldots B_\kappa$.
4. For each $\mathcal{T}_{B_j} \in \mathbb{T}$, we update the parent of the root node $N_0(\mathsf{parent})$ with $N_0(\mathsf{parent}) = R_{B_j}(\mathsf{parent})$, where R_{B_j} is the leaf node corresponding to B_j in the tree construction returned above by the WRSC algorithm. We also delete the leaf node R_{B_j}, thereby effectively replacing R_{B_j} by $\mathcal{T}_{B_j}$.

By Lemma B.7 we know that the WRSC algorithm (Line 16) runs in nearly-linear $\widetilde{O}(m)$ time. Taking everything together, this proves the time complexity of Algorithm 1. The $O(\log n)$ depth of $\mathcal{T}$ follows by the fact that (i) the depth of every tree $\mathcal{T}_{B_j}$ is $O(\log n)$, and (ii) the distance between the root of $\mathcal{T}$ and the root of any such tree $\mathbb{T}_{B_j}$ in $\mathcal{T}$ is also $O(\log n)$.

Finally, to set the depth attribute of every internal node, we perform a top-down traversal from the root of $\mathcal{T}$ to the leaves, and update the depth for every internal node as one more than the depth of their parent. □

We are now ready to prove the main result, which we will state here in full.

Theorem B.10 (Formal statement of Theorem 5.1)**.** *Let $G = (V, E, w)$ be a graph with $w_{\max}/w_{\min} = O(n^\gamma)$ for a constant $\gamma > 1$, and $k > 1$ such that $\lambda_{k+1} > 0$ and $\lambda_k < \left(\frac{1}{270 \cdot c_0 \cdot (k+1)^6}\right)^2$, where c_0 is the constant in Lemma A.12. Algorithm 1 runs in time $\widetilde{O}(m)$ and both constructs an HC tree $\mathcal{T}$ of G and computes $\mathsf{COST}_G(\mathcal{T})$ satisfying $\mathsf{COST}_G(\mathcal{T}) = O\left(2^{k(\gamma+1)} \cdot k^{23}/\lambda_{k+1}^{10}\right) \cdot \mathsf{OPT}_G$. In particular, when $\lambda_{k+1} = \Omega(1)$ and $k = O(1)$, the algorithm's constructed tree $\mathcal{T}$ satisfies that $\mathsf{COST}_G(\mathcal{T}) = O(1) \cdot \mathsf{OPT}_G$.*

Proof of Theorem B.10. Let $\mathcal{T}$ be the tree returned by Algorithm 1. By combining (4), (7), Lemma 5.2, and the approximation guarantee in Lemma B.7 we have that

$$\mathsf{COST}_G(\mathcal{T}) = O\left(2^{k(\gamma+1)} \cdot k^{23}/\lambda_{k+1}^{10}\right) \cdot \mathsf{OPT}_G.$$

The time complexity of Algorithm 1 follows by Lemma B.9. Moreover, as the depth of $\mathcal{T}$ is $O(\log n)$, and every internal node $N \in \mathcal{T}$ has a pointer parent and attributes size and depth, by Lemma B.8 we can compute $\mathsf{COST}_G(\mathcal{T})$ in nearly-linear time $\widetilde{O}(m)$.

We complete the proof by dealing with our assumption that the algorithm knows the number of clusters k. If the number of clusters k is unknown, we perform a standard technique from the literature (Cohen-Addad et al., 2017; Manghiuc & Sun, 2021) and run independent copies of Algorithm 1 with all possible values k' ranging from 1 to k. By adding an extra $O(k) = O(1)$ factor in the overall time complexity, we ensure that one of the runs has the correct number of clusters $k' = k$. □

C. Omitted Details of Proof of Theorem 5.4

This section presents omitted details of the proof of Theorem 5.4, and is structured as follows: we present a technical overview of our result in Section C.1; we describe our designed algorithm in Section C.2, and prove Theorem 5.4 in Section C.3. We introduce the parameter η_S such that

$$\eta_S \geqslant \max_{1 \leqslant i \leqslant k} \frac{\Delta(S_i)}{\delta(S_i)};$$

this will be used in our later discussion.

C.1. Overview of Main Techniques

We give an overview of our main techniques and discuss why spectral clustering suffices to construct HC trees with good approximation guarantees for well-clustered graphs. One general challenge for analysing the approximation ratio of any HC tree $\mathcal{T}$ of G is due to the limited understanding of the "structure" of an optimal HC tree. There are two main approaches to address this: (1) the first approach is to ensure that most "cuts" induced at the top of the tree $\mathcal{T}$ are sparse; as such, a recursive application of the sparsest cut algorithm is employed; (2) the second approach is to reason about the structure of an optimal HC tree assuming that the underlying graphs have a highly symmetric hierarchical structure of clusters (Cohen-Addad et al., 2017), or a clear cluster structure (Manghiuc & Sun, 2021). In particular, the algorithm in (Manghiuc & Sun, 2021) requires that every returned vertex set has high inner conductance. Therefore, taking these into account, the starting point of our approach is to consider the role of high inner-conductance in bounding the cost of an HC tree.

Suppose that the optimal clusters $S_1, \ldots, S_k$ corresponding to $\rho(k)$ are given, and it holds $\Phi_{G[S_i]} \geqslant \Phi_{\text{in}}$ for $1 \leqslant i \leqslant k$. These k clusters could have different sizes, volumes, and degree distributions. We show that one can construct a tree $\mathcal{T}_S$, by first constructing trees $\mathcal{T}_{G[S_i]}$ on every induced subgraph $G[S_i]$, and simply concatenating these $\mathcal{T}_{G[S_i]}$ in a "caterpillar" style according to the sizes of $|S_i|$ $(1 \leqslant i \leqslant k)$. Moreover, the cost of our constructed tree $\mathcal{T}_S$ can be upper bounded with respect to OPT_G. See Figure 5 for an illustration of a caterpillar tree, and Lemma C.1 for a formal statement; the proof of Lemma C.1 can be found at the end of the subsection.

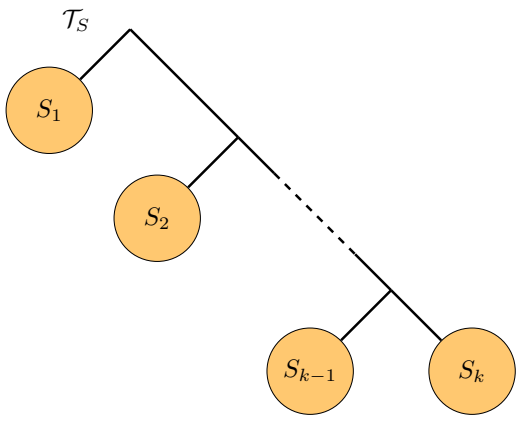

Figure 5. An illustration of our constructed "caterpillar" tree from optimal clusters $S_1, \ldots, S_k$ that satisfy $|S_1| \geqslant \ldots \geqslant |S_k|$.

Lemma C.1. *Let $G = (V, E, w)$ be an input graph of n vertices, and G has optimal clusters $S_1, \ldots, S_k$ corresponding to $\rho(k) \leqslant 1/k$ and it holds that $\Phi_{G[S_i]} \geqslant \Phi_{\text{in}}$ for $1 \leqslant i \leqslant k$. Given $S_1, \ldots, S_k$ as part of the input, there is a nearly-linear*

time algorithm that constructs an HC *tree* $\mathcal{T}_S$ *such that*

$$\mathsf{COST}_G(\mathcal{T}_S) \leqslant \frac{36 \cdot \eta_S}{\Phi_{\text{in}}} \cdot \mathsf{OPT}_G.$$

Since $\Phi_{\text{in}} = \Omega(1)$ for many well-clustered graphs, Lemma C.1 states that a simple caterpillar construction based on the k optimal clusters achieves an approximation of $O(\eta_S)$, which is $O(1)$ if the degrees of *every* cluster are almost-balanced. Moreover, thanks to this caterpillar structure over the S_i's and an arbitrary tree construction for every $G[S_i]$, we can construct $\mathcal{T}_S$ and compute its cost in nearly-linear time.

However, the above discussion assumes that the optimal clusters are known, which are co-NP hard to find (Blum et al., 1981). As the starting point of our algorithm, we therefore employ the spectral clustering algorithm, expecting that the output clusters $P_1, \ldots, P_k$ could directly replace $S_1, \ldots, S_k$. Unfortunately, it is unclear whether this approach would work for the following two reasons:

1. while the output $P_1, \ldots, P_k$ of spectral clustering can be applied to approximate the optimal $S_1, \ldots, S_k$, it is unknown whether every P_i approximates their corresponding optimal S_i with respect to $\Phi_{G[S_i]}$; moreover, it remains open whether the inner conductance $\Phi_{G[S_i]}$ is approximated by $\Phi_{G[P_i]}$. Without the inner conductance guarantee of $\Phi_{G[P_i]}$ for all $1 \leqslant i \leqslant k$, we cannot apply Lemma C.1.
2. while Lemma A.9 shows that the symmetric difference between every P_i and their optimal corresponding S_i can be bounded, this approximation guarantee can still increase the ratio between the maximum and minimum degrees in P_i. That is, with spectral clustering alone, we cannot upper bound η_P (which is an upper bound for $\max_i(\Delta(P_i)/\delta(P_i))$) from η_S. Moreover, Lemma A.9 does not provide any guarantees on the difference in size between S_i and P_i, which is crucial if we want to construct arbitrary $\mathcal{T}_{P_i}$'s.

We overcome these two bottlenecks through the following high-level ideas:

1. we show that bounded inner conductance of every output P_i isn't needed, and that the approximate P_i's returned from spectral clustering suffice to construct an HC tree with guaranteed approximation. This is a fundamental difference compared with (Manghiuc & Sun, 2021), which requires the inner conductance being constant for every partition set.
2. we introduce a novel bucketing procedure that decomposes P_i's into further subsets, by grouping together vertices of similar degree. These subsets will form the basis of our tree construction.

Thanks to the bucketing step, we are able to construct an HC tree in nearly-linear line.

Proof of Lemma C.1. Given an input graph G and optimal clusters $S_1, \ldots, S_k$ such that $|S_1| \geqslant \ldots \geqslant |S_k|$, we construct the tree $\mathcal{T}_S$ through the following two steps: first of all, we construct an arbitrary HC tree $\mathcal{T}_{S_i}$ for every $G[S_i]$, $1 \leqslant i \leqslant k$; secondly, we construct $\mathcal{T}_S$ by concatenating the trees $\mathcal{T}_{S_1}, \ldots, \mathcal{T}_{S_k}$ in a caterpillar style, i.e., $\mathcal{T}_{S_1}$ is at the top of the tree, and $\mathcal{T}_{S_2}, \ldots, \mathcal{T}_{S_k}$ are appended inductively. Since we can construct every $\mathcal{T}_{G[S_i]}$ in an arbitrary manner and the tree $\mathcal{T}_S$ can be constructed from $\{\mathcal{T}_{G[S_i]}\}_{1 \leqslant i \leqslant k}$ through sorting the clusters by their sizes, the overall algorithm runs in nearly-linear time.

Next, we analyse $\mathsf{COST}_G(\mathcal{T}_S)$. By definition, it holds that

$$\mathsf{COST}_G(\mathcal{T}_S) = \sum_{i=1}^{k} \sum_{e \in E[G[S_i]]} \mathsf{cost}_G(e) + \sum_{e \in \widetilde{E}} \mathsf{cost}_G(e), \tag{44}$$

where $\widetilde{E}$ is the set of edges crossing different clusters. For the first summation of (44), we have that

$$\sum_{i=1}^{k} \mathsf{COST}_{G[S_i]}(\mathcal{T}_{S_i}) \leqslant \frac{18 \cdot \eta_S}{\Phi_{\text{in}}} \cdot \sum_{i=1}^{k} \mathsf{OPT}_{G[S_i]} \leqslant \frac{18 \cdot \eta_S}{\Phi_{\text{in}}} \cdot \mathsf{OPT}_G. \tag{45}$$

Here, the first inequality follows the fact that the cost of any HC tree of $G = (V, E, w)$ is at most $|V| \cdot \mathrm{vol}(G)$ (Fact A.1) and subsequently applying Lemma A.8; the second inequality holds because $\mathsf{OPT}_{G[S_i]}$ is at most OPT_G for all $1 \leqslant i \leqslant k$.

For the second summation of (44), it holds that

$$\sum_{e\in\widetilde{E}}\mathsf{cost}_G(e)=\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\sum_{\substack{u\in S_i,v\in S_j\\e=\{u,v\}}}\mathsf{cost}_G(e).$$

We look at all the edges adjacent to vertices in S_1. By construction, it holds that $|S_1|\geqslant n/k$, and $|\mathsf{leaves}\,(\mathcal{T}_S[u\vee v])|=n$ for any edge $\{u,v\}$ satisfying $u\in S_1, v\in V\setminus S_1$. Since $w(S_1,V\setminus S_1)\leqslant \mathrm{vol}(S_1)\cdot\rho(k)$, it holds that

$$\sum_{\substack{u\in S_1,v\notin S_1\\e=\{u,v\}}}\mathsf{cost}_G(e)=n\cdot w(S_1,V\setminus S_1)\leqslant n\cdot\mathrm{vol}(S_1)\cdot\rho(k)\leqslant k\cdot|S_1|\cdot\rho(k)\cdot\mathrm{vol}(S_1).$$

We continue with looking at the edges between S_2 and $V\setminus(S_1\cup S_2)$. As S_2 is the largest cluster among $S_2,\ldots,S_k$, it holds that $|S_2|\geqslant\frac{n-|S_1|}{k-1}$, which implies that $n-|S_1|<k\cdot|S_2|$. Notice that any edge $e=\{u,v\}$ with $u\in S_2$ and $v\in V\setminus(S_1\cup S_2)$ satisfies that $|\mathsf{leaves}\,(\mathcal{T}_S[u\vee v])|=n-|S_1|$, and as such we have that

$$\sum_{\substack{u\in S_2,v\notin S_1\cup S_2\\e=\{u,v\}}}\mathsf{cost}_G(e)\leqslant(n-|S_1|)\cdot w(S_2,V\setminus(S_1\cup S_2))\leqslant k\cdot|S_2|\cdot\rho(k)\cdot\mathrm{vol}(S_2).$$

Generalising the analysis above, we have that

$$\begin{aligned}\sum_{e\in\widetilde{E}}\mathsf{cost}_G(e)&=\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\sum_{\substack{u\in S_i,v\in S_j\\e=\{u,v\}}}\mathsf{cost}_G(e)\\&\leqslant k\cdot\rho(k)\cdot\sum_{i=1}^{k}|S_i|\cdot\mathrm{vol}(S_i)\\&\leqslant\frac{18\cdot\eta_S}{\Phi_{\mathrm{in}}}\cdot\sum_{i=1}^{k}\mathsf{OPT}_{G[S_i]},\end{aligned}\tag{46}$$

where the last inequality follows from Lemma A.8. Combining (44), (45) with (46), we have that

$$\mathsf{COST}_G(\mathcal{T}_S)\leqslant\frac{36\cdot\eta_S}{\Phi_{\mathrm{in}}}\cdot\mathsf{OPT}_G,$$

which proves the statement. □

C.2. Algorithm Description

Now informally describe our algorithm and refer the reader to Algorithm 2 for the formal presentation. The algorithm takes as input a graph $G=(V,E,w)$ with k optimal clusters $\{S_i\}_{i=1}^k$. For simplicity, we assume that the algorithm knows the target number of clusters k and an upper bound η_S such that

$$\eta_S\geqslant\max_{1\leqslant i\leqslant k}\frac{\Delta(S_i)}{\delta(S_i)},$$

and we carefully deal with this assumption at the end of the section. As the first step, the algorithm runs the SpectralClustering subroutine and obtains the approximate clusters $\{P_i\}_{i=1}^k$. Following our previous discussion, it is unclear whether the algorithm can or cannot use the clusters P_i directly in order to construct an $O(1)$-approximate tree. To overcome this difficulty, we further partition every P_i into degree-based buckets, but rather than setting $\beta=2^{k(\gamma+1)}$ as we did for Algorithm 1, we set $\beta=\eta_S$. Recall then for any set P_i and every vertex $u\in P_i$, the *bucket* of P_i starting at u is defined

$$B\,(u)=\{v\in P_i:d_u\leqslant d_v<\eta_S\cdot d_u\}\,.$$

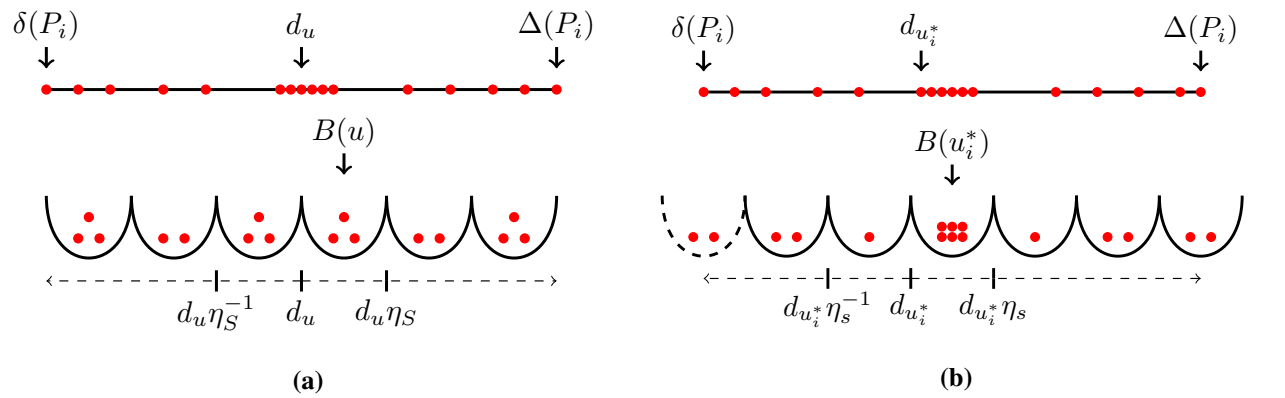

Figure 6. This figure illustrates two different bucketings of the vertices inside some P_i. Figure **(a)** is a bucketing induced by some vertex u. Figure **(b)** is the bucketing induced by u_i^* with $\mathrm{vol}(B(u^*))$ maximised.

Again, it is important to note that the bucket $B(u)$ contains *every* vertex $v \in P_i$ whose degree satisfies $d_u \leqslant d_v < \eta_S \cdot d_u$. We refer the reader to Figure 6(a) for an illustration of the bucketing procedure.

Similar to before, vertices $u, v \in P_i$ having different degrees $d_u \neq d_v$ generally induce different bucketings. In order to determine the appropriate bucketing, Algorithm 2 chooses as representative a vertex $u_i^* \in P_i$, whose induced bucket $B(u_i^*)$ has the highest volume (Line 8 of Algorithm 2). To motivate this choice, notice that every cluster P_i has a large overlap with its corresponding optimal cluster S_i and the degrees of the vertices inside S_i are within a factor of η_S of each other. If an arbitrary vertex $u \in P_i$ is chosen as representative, the bucketing $B(u)$ of P_i might equally divide the vertices in $P_i \cap S_i$ into two consecutive buckets, which is undesirable. However, we will prove that our specific choice of u_i^* guarantees that the bucketing $\mathcal{B}_{u_i^*}$ contains one bucket $B_j \in \mathcal{B}_{u_i^*}$ largely overlapping with S_i (see Figure 6(b) for an illustration).

Once the bucketing procedure of P_i is complete the algorithm proceeds and constructs an arbitrary *balanced* binary tree $\mathcal{T}_{B_j}$, for every bucket B_j, and adds the tree $\mathcal{T}_{B_j}$ to a global set of trees $\mathbb{T}$. This process is repeated for every cluster P_i. As a final step, the algorithm merges the trees inside $\mathbb{T}$ in a caterpillar fashion in decreasing order of the sizes, placing the larger sized trees closer to the root (Lines 16-18 of Algorithm 2). The resulting tree $\mathcal{T}_{\mathrm{SCB}}$ is the output of the algorithm.

C.3. Algorithm Analysis

This subsection analyses the performance of Algorithm 2, and proves Theorem 5.4. It is organised as follows: in the first part we prove several properties of our carefully constructed bucketing procedure. In the second part, we prove that the cost of our constructed tree $\mathcal{T}_{\mathrm{SCB}}$ achieves the claimed approximation guarantee. Thirdly, we give a detailed description of how we can efficiently construct the tree $\mathcal{T}_{\mathrm{SCB}}$ and compute its cost in nearly-linear time. The proof of Theorem 5.4 follows by combining the three main results.

We first prove that our bucketing procedure induces at most $2k$ buckets inside every cluster P_i, and the total number of buckets throughout all clusters P_i is at most $2k^2$.

Lemma C.2. *Let P_i be an arbitrary cluster, and $u_i^* = \mathrm{argmax}_{u \in P_i} \mathrm{vol}(B(u))$ be a vertex inducing the bucketing $\mathcal{B}_{u_i^*}$. The following statements hold:*

1. *$B(u_i^*) \in \mathcal{B}_{u_i^*}$;*
2. *For every optimal cluster S_j, the vertices in $S_j \cap P_i$ belong to at most two buckets, i.e.,*
$$\left|\left\{B_t \in \mathcal{B}_{u_i^*} : S_j \cap B_t \neq \emptyset\right\}\right| \leqslant 2;$$
3. *The bucketing induced by u_i^* contains at most $2k$ buckets $\left|\mathcal{B}_{u_i^*}\right| \leqslant 2k$.*

Proof. The first property follows trivially from the definition. The second property follows from our assumption that $\Delta(S_i) \leqslant \eta_S \cdot \delta(S_i)$ for every optimal cluster S_i. Therefore, the vertices inside S_i belong to at most 2 buckets. The third property follows from the second one and summing over all clusters. □

Our second result presents a very important property of our bucketing procedure. We show that, in every cluster P_i, exactly one bucket $B_j \in \mathcal{B}_{u_i^*}$ has large overlap with the optimal cluster S_i, and all the other buckets have low volume. We emphasise that this result is only possible due to our calibration technique of choosing the bucketing induced by a vertex $u_i^* \in P_i$, whose bucket $B(u_i^*)$ has the highest volume.

Lemma C.3. *Let P_i be an arbitrary cluster, and $u_i^* = \operatorname{argmax}_{u \in P_i} \operatorname{vol}(B(u))$ be a vertex inducing the bucketing $\mathcal{B}_{u_i^*}$. The following two statements hold:*

1. *There exists a unique* heavy *bucket $B_j \in \mathcal{B}_{u_i^*}$ such that $\operatorname{vol}(B_j \cap S_i) \geqslant \left(1 - \frac{2k \cdot C_{A.9}}{3\Upsilon(k)}\right) \cdot \operatorname{vol}(S_i)$;*
2. *For every other bucket $B_t \in \mathcal{B}_{u_i^*} \setminus \{B_j\}$ it holds that $\operatorname{vol}(B_t) \leqslant \frac{k \cdot C_{A.9}}{\Upsilon(k)} \cdot \operatorname{vol}(S_i)$.*

Proof. For the first statement, we show that the bucket $B(u_i^*)$ satisfies the claim. Let $u_{\min}$ be a vertex in $P_i \cap S_i$ of the lowest degree. Since the degrees of all vertices in S_i (and hence in $P_i \cap S_i$) are within a factor of η_S from each other, we know that all vertices in $P_i \cap S_i$ must be in the bucket $B(u_{\min})$, i.e., $P_i \cap S_i \subseteq B(u_{\min})$. Moreover, by the choice of u_i^* we have that

$$\operatorname{vol}(B(u_i^*)) \geqslant \operatorname{vol}(B(u_{\min})) \geqslant \operatorname{vol}(P_i \cap S_i) \geqslant \operatorname{vol}(S_i) - \operatorname{vol}(P_i \triangle S_i) \geqslant \left(1 - \frac{k \cdot C_{A.9}}{3\Upsilon(k)}\right) \cdot \operatorname{vol}(S_i), \tag{47}$$

where the last inequality follows by Lemma A.9. We now write

$$\begin{aligned}
\operatorname{vol}(B(u_i^*) \cap S_i) &= \operatorname{vol}(B(u_i^*)) - \operatorname{vol}(B(u_i^*) \setminus S_i) \\
&\geqslant \left(1 - \frac{k \cdot C_{A.9}}{3\Upsilon(k)}\right) \cdot \operatorname{vol}(S_i) - \operatorname{vol}(P_i \setminus S_i) \\
&\geqslant \left(1 - \frac{2k \cdot C_{A.9}}{3\Upsilon(k)}\right) \cdot \operatorname{vol}(S_i),
\end{aligned}$$

where the first inequality uses (47). The uniqueness of the heavy bucket follows by noticing that $\operatorname{vol}(B(u_i^*) \cap S_i) > \operatorname{vol}(S_i)/2$.

The second statement essentially follows from the fact that S_i has a large volume overlap with $B(u_i^*)$. Concretely, let $B_t \in \mathcal{B}_{u_i^*} \setminus \{B(u_i^*)\}$ be an arbitrary bucket. We have that

$$\begin{aligned}
\operatorname{vol}(B_t) &= \operatorname{vol}(B_t \cap S_i) + \operatorname{vol}(B_t \setminus S_i) \\
&\leqslant \operatorname{vol}(S_i) - \operatorname{vol}(S_i \setminus B_t) + \operatorname{vol}(P_i \setminus S_i) \\
&\leqslant \operatorname{vol}(S_i) - \operatorname{vol}(S_i \cap B(u_i^*)) + \operatorname{vol}(P_i \triangle S_i) \\
&\leqslant \frac{2k \cdot C_{A.9}}{3\Upsilon(k)} \cdot \operatorname{vol}(S_i) + \frac{k \cdot C_{A.9}}{3\Upsilon(k)} \cdot \operatorname{vol}(S_i) \\
&= \frac{k \cdot C_{A.9}}{\Upsilon(k)} \cdot \operatorname{vol}(S_i).
\end{aligned}$$

This completes the proof. □

In order to analyse the final constructed tree $\mathcal{T}_{\mathrm{SCB}}$, we analyse the properties of the buckets in the context of the entire graph G, and not just on every set P_i. We denote by $\mathcal{B}$ the set containing all the buckets obtained throughout all sets P_i, i.e.,

$$\mathcal{B} \triangleq \bigcup_{i=1}^{k} \mathcal{B}_{u_i^*},$$

where $u_i^* = \operatorname{argmax}_{u \in P_i} \operatorname{vol}(B(u))$ is a vertex inducing the corresponding bucketing in set P_i. We remark that $\mathcal{B}$ is a partition of V. We use

$$\ell \triangleq |\mathcal{B}|$$

to denote the total number of buckets, and we know by Lemma C.2 that $\ell \leqslant 2k^2$. For convenience, we label the buckets $\mathcal{B} = \{B_1, \ldots, B_\ell\}$ in decreasing order of the sizes breaking ties arbitrarily, i.e., $|B_i| \geqslant |B_{i+1}|$, for all $i < \ell$.

Based on Lemma C.3, exactly k buckets in $\mathcal{B}$ are heavy, meaning that they have large overlap with the optimal clusters S_i. We denote by $\mathcal{B}_{\mathrm{H}}$ the set of those k buckets; we emphasise once more that each heavy bucket corresponds to a unique cluster S_i and vice-versa. Additionally, we denote by $\mathcal{B}_{\mathrm{L}} \triangleq \mathcal{B} \setminus \mathcal{B}_{\mathrm{H}}$ the set of the remaining *light* buckets.

The next two results summarise the main properties of our constructed buckets. We first present three properties of the heavy buckets which bound their size, volume and weight of the crossing edges.

Lemma C.4. *For every heavy bucket $B_j \in \mathcal{B}_{\mathrm{H}}$ corresponding to S_i, the following statements hold:*

(B1) $\mathrm{vol}(B_j) \leqslant \left(1 + \frac{k \cdot C_{A.9}}{3\Upsilon(k)}\right) \cdot \mathrm{vol}(S_i)$;

(B2) $|B_j| \leqslant (1+\eta_S) \cdot |S_i|$.

(B3) $w(B_j, \overline{B_j}) \leqslant \frac{k \cdot C_{A.9} + \lambda_{k+1}}{\Upsilon(k)} \cdot \mathrm{vol}(S_i)$.

Proof. Let $B_j \in \mathcal{B}_{\mathrm{H}}$ be a heavy bucket corresponding to S_i. For (B1), we have that

$$\mathrm{vol}(B_j) = \mathrm{vol}(B_j \cap S_i) + \mathrm{vol}(B_j \setminus S_i) \leqslant \mathrm{vol}(S_i) + \mathrm{vol}(P_i \triangle S_i) \leqslant \left(1 + \frac{k \cdot C_{A.9}}{3\Upsilon(k)}\right) \cdot \mathrm{vol}(S_i)\,.$$

To prove (B2), suppose for contradiction that $|B_j| > (1+\eta_S) \cdot |S_i|$, which implies that $|B_j \setminus S_i| = |B_j| - |B_j \cap S_i| > \eta_S \cdot |B_j \cap S_i|$. Based on this, we have that

$$\begin{aligned}\mathrm{vol}(B_j \setminus S_i) &\geqslant |B_j \setminus S_i| \cdot \delta(B_j \setminus S_i) > \eta_S \cdot |B_j \cap S_i| \cdot \frac{\Delta(B_j \cap S_i)}{\eta_S} \\ &\geqslant \mathrm{vol}(B_j \cap S_i) \geqslant \left(1 - \frac{2k \cdot C_{A.9}}{3\Upsilon(k)}\right) \cdot \mathrm{vol}(S_i)\,,\end{aligned}$$

where the second inequality follows from the fact that, inside bucket B_j all the degrees are within a factor of η_S of each other and the last inequality follows by Lemma C.3. This implies that

$$\mathrm{vol}(S_i) < \frac{1}{1 - \frac{2k \cdot C_{A.9}}{3\Upsilon(k)}} \cdot \mathrm{vol}(P_i \triangle S_i) \leqslant \frac{\mathrm{vol}(S_i)}{\frac{3\Upsilon(k)}{k \cdot C_{A.9}} - 2} \leqslant \mathrm{vol}(S_i)\,,$$

where the second inequality follows by Lemma A.9, and last inequality by our assumption on $\Upsilon(k)$.

Finally, we prove (B3). We have that

$$\begin{aligned}w(B_j, \overline{B_j}) &\leqslant w(B_j \cap S_i, \overline{B_j}) + w(B_j \setminus S_i, \overline{B_j}) \\ &\leqslant w(B_j \cap S_i, \overline{B_j} \cap S_i) + w(B_j \cap S_i, \overline{B_j} \setminus S_i) + \mathrm{vol}(B_j \setminus S_i) \\ &\leqslant \mathrm{vol}\big(\overline{B_j} \cap S_i\big) + w(S_i, \overline{S_i}) + \mathrm{vol}(P_i \triangle S_i) \\ &\leqslant \frac{2k \cdot C_{A.9}}{3\Upsilon(k)} \cdot \mathrm{vol}(S_i) + \rho(k) \cdot \mathrm{vol}(S_i) + \frac{k \cdot C_{A.9}}{3\Upsilon(k)} \cdot \mathrm{vol}(S_i) \qquad (48) \\ &\leqslant \frac{k \cdot C_{A.9} + \lambda_{k+1}}{\Upsilon(k)} \cdot \mathrm{vol}(S_i)\,, \qquad (49)\end{aligned}$$

where (48) follows from Lemmas C.3 and A.9, and (49) uses the fact that $\rho(k) = \frac{\lambda_{k+1}}{\Upsilon(k)}$. □

The final technical result is similar with Lemma C.4, and presents several useful upper bounds of the volume and size of the light buckets.

Lemma C.5. *For every* light *bucket $B_t \in \mathcal{B}_{\mathrm{L}}$, the following statements hold:*

(B4) $\mathrm{vol}(B_t \cap S_j) \leqslant \eta_S \cdot \mathrm{vol}(B_t \cap S_i)$, *for all clusters S_i, S_j satisfying $|B_t \cap S_j| \leqslant |B_t \cap S_i|$;*

(B5) $\mathrm{vol}(B_t \cap S_i) \leqslant \frac{2k \cdot C_{A.9}}{3\Upsilon(k)} \cdot \mathrm{vol}(S_i)$, *for all clusters S_i;*

(B6) $\sum_{B_t\in\mathcal{B}_{\mathrm{L}}}|B_t|\cdot\mathrm{vol}(B_t)\leqslant\frac{4k^2\cdot C_{A.9}\cdot\eta_S}{3\Upsilon(k)}\cdot\sum_{i=1}^{k}|S_i|\cdot\mathrm{vol}(S_i)$.

Proof. Let $B_t\in\mathcal{B}_{\mathrm{L}}$ be a light bucket, and let S_i,S_j be optimal clusters so that $|B_t\cap S_j|\leqslant|B_t\cap S_i|$. It holds that

$$\mathrm{vol}(B_t\cap S_j)\leqslant|B_t\cap S_j|\cdot\Delta(B_t\cap S_j)\leqslant|B_t\cap S_j|\cdot\eta_S\cdot\delta(B_t\cap S_i)\leqslant\eta_S\cdot\mathrm{vol}(B_j\cap S_i)\,,$$

where the second inequality uses the fact that inside bucket B_t all the degrees are within a factor of η_S. This proves (B4).

Next, we prove (B5). Suppose that B_t is a bucket from some cluster P_r, and let S_i be an arbitrary cluster. The proof continues by a case distinction:

Case 1**:** $r=i$**.** In this case, since B_t is a light bucket in P_r, by Lemma C.3 we have that

$$\mathrm{vol}(B_t\cap S_i)=\mathrm{vol}(B_t\cap S_r)\leqslant\frac{2k\cdot C_{A.9}}{3\Upsilon(k)}\cdot\mathrm{vol}(S_r)\,.$$

Case 2**:** $r\neq i$**.** In this case we simply have that

$$\mathrm{vol}(B_t\cap S_i)\leqslant\mathrm{vol}(P_r\cap S_i)\leqslant\mathrm{vol}(P_i\triangle S_i)\leqslant\frac{k\cdot C_{A.9}}{3\Upsilon(k)}\cdot\mathrm{vol}(S_i)\,.$$

Combining the two cases proves (B5).

We finally prove (B6). We have that

$$\begin{aligned}
&\sum_{B_t\in\mathcal{B}_{\mathrm{L}}}|B_t|\mathrm{vol}(B_t)\\
&=\sum_{B_t\in\mathcal{B}_{\mathrm{L}}}\left(\sum_{i=1}^{k}|B_t\cap S_i|\right)\left(\sum_{j=1}^{k}\mathrm{vol}(B_t\cap S_j)\right)\\
&=\sum_{B_t\in\mathcal{B}_{\mathrm{L}}}\sum_{i,j=1}^{k}|B_t\cap S_i|\mathrm{vol}(B_t\cap S_j)\\
&=\sum_{B_t\in\mathcal{B}_{\mathrm{L}}}\left(\sum_{\substack{i,j=1:\\|B_t\cap S_i|\leqslant|B_t\cap S_j|}}^{k}|B_t\cap S_i|\mathrm{vol}(B_t\cap S_j)+\sum_{\substack{i,j=1:\\|B_t\cap S_i|>|B_t\cap S_j|}}^{k}|B_t\cap S_i|\mathrm{vol}(B_t\cap S_j)\right) &(50)\\
&\leqslant\sum_{B_t\in\mathcal{B}_{\mathrm{L}}}\left(\sum_{\substack{i,j=1:\\|B_t\cap S_i|\leqslant|B_t\cap S_j|}}^{k}|B_t\cap S_j|\mathrm{vol}(B_t\cap S_j)+\sum_{\substack{i,j=1:\\|B_t\cap S_i|>|B_t\cap S_j|}}^{k}|B_t\cap S_i|\cdot\eta_S\cdot\mathrm{vol}(B_t\cap S_i)\right) &(51)\\
&\leqslant\sum_{B_t\in\mathcal{B}_{\mathrm{L}}}\left(k\sum_{j=1}^{k}|B_t\cap S_j|\mathrm{vol}(B_t\cap S_j)+k\sum_{i=1}^{k}|B_t\cap S_i|\cdot\eta_S\cdot\mathrm{vol}(B_t\cap S_i)\right)\\
&\leqslant 2k\cdot\eta_S\sum_{i=1}^{k}\sum_{B_t\in\mathcal{B}_{\mathrm{L}}}|B_t\cap S_i|\mathrm{vol}(B_t\cap S_i)\\
&\leqslant 2k\cdot\eta_S\cdot\frac{2k\cdot C_{A.9}}{3\Upsilon(k)}\cdot\sum_{i=1}^{k}\sum_{B_t\in\mathcal{B}_{\mathrm{L}}}|B_t\cap S_i|\mathrm{vol}(S_i) &(52)\\
&\leqslant\frac{4k^2\cdot C_{A.9}\cdot\eta_S}{3\Upsilon(k)}\sum_{i=1}^{k}|S_i|\mathrm{vol}(S_i)\,,
\end{aligned}$$

where (50) follows by grouping the terms $|B_t\cap S_i|\mathrm{vol}(B_t\cap S_j)$ into two categories, (51) follows by (B4), and (52) follows by (B5). □

Now we combine the properties shown earlier, and upper bound the cost of our constructed tree $\mathcal{T}_{\mathrm{SCB}}$ through the following lemma.

Lemma C.6. *It holds that*

$$\mathsf{COST}_G\left(\mathcal{T}_{\mathrm{SCB}}\right) \leqslant \left(1+\frac{5k^4\cdot C_{A.9}}{\Upsilon(k)}\right)\cdot\frac{36\cdot\eta_S^2}{\Phi_{\mathrm{in}}}\cdot\mathsf{OPT}_G.$$

Proof. By construction, we know that $\mathcal{T}_{\mathrm{SCB}}$ is the caterpillar tree formed by merging the trees $\mathcal{T}_{B_j}$, for all $B_j\in\mathcal{B}$. Therefore, in order to bound the overall cost it suffices to bound the cost within each induced subtree $\mathcal{T}_{B_j}$ as well as the cost of the edges crossing different buckets. Formally, we have that

$$\begin{aligned}\mathsf{COST}_G\left(\mathcal{T}_{\mathrm{SCB}}\right)&=\sum_{j=1}^{\ell}\mathsf{COST}_{G[B_j]}\left(\mathcal{T}_{B_j}\right)+\sum_{j=1}^{\ell-1}\sum_{t=j+1}^{\ell}\sum_{e\in E(B_j,B_t)}\mathsf{cost}_G(e)\\&\leqslant\sum_{j=1}^{\ell}|B_j|\mathrm{vol}(B_j)+\sum_{j=1}^{\ell-1}\sum_{t=j+1}^{\ell}\sum_{e\in E(B_j,B_t)}\mathsf{cost}_G(e).\end{aligned}\tag{53}$$

W study the first term of (53), and have that

$$\begin{aligned}&\sum_{j=1}^{\ell}|B_j|\mathrm{vol}(B_j)\\&\leqslant\sum_{B_j\in\mathcal{B}_{\mathrm{H}}}|B_j|\mathrm{vol}(B_j)+\sum_{B_t\in\mathcal{B}_{\mathrm{L}}}|B_t|\mathrm{vol}(B_t)\\&\leqslant(1+\eta_S)\left(1+\frac{k\cdot C_{A.9}}{3\Upsilon(k)}\right)\left(\sum_{i=1}^{k}|S_i|\mathrm{vol}(S_i)\right)+\frac{4k^2\cdot C_{A.9}\cdot\eta_S}{3\Upsilon(k)}\cdot\left(\sum_{i=1}^{k}|S_i|\mathrm{vol}(S_i)\right)&(54)\\&\leqslant\left(2\eta_S\cdot\left(1+\frac{k\cdot C_{A.9}}{3\Upsilon(k)}\right)+\frac{4k^2\cdot C_{A.9}\cdot\eta_S}{3\Upsilon(k)}\right)\sum_{i=1}^{k}|S_i|\mathrm{vol}(S_i)\\&\leqslant\left(2\eta_S+\frac{2k^2\cdot C_{A.9}\cdot\eta_S}{\Upsilon(k)}\right)\cdot\frac{18\cdot\eta_S}{\Phi_{\mathrm{in}}}\cdot\mathsf{OPT}_G&(55)\\&=\left(1+\frac{k^2\cdot C_{A.9}}{\Upsilon(k)}\right)\cdot\frac{36\cdot\eta_S^2}{\Phi_{\mathrm{in}}}\cdot\mathsf{OPT}_G\end{aligned}$$

where (54) follows from (B1), (B2), (B6), and (55) follows by Lemma A.8.

Next we look at the second term of (53). Let $B_j\in\mathcal{B}$ be an arbitrary bucket, and we study the cost of the edges between B_j and all the other buckets B_t positioned lower in the tree, i.e., $|B_t|\leqslant|B_j|$. By construction, for every such edge e we have that

$$\mathsf{cost}_G(e)=w_e\sum_{t=j+1}^{\ell}|B_t|\leqslant w_e\cdot\ell\cdot|B_j|.$$

Therefore, we can upper bound the total cost of the crossing edges adjacent to B_j by

$$\sum_{t=j+1}^{\ell}\sum_{e\in E(B_j,B_t)}\mathsf{cost}_G(e)\leqslant\ell\cdot|B_j|w(B_j,\overline{B_j}).$$

Hence, by summing over all buckets B_j we have that

$$\begin{aligned}
&\sum_{j=1}^{\ell-1}\sum_{t=j+1}^{\ell}\sum_{e\in E(B_j,B_t)}\mathsf{cost}_G(e)\\
&\leqslant\sum_{j=1}^{\ell-1}\ell\cdot|B_j|w(B_j,\overline{B_j})\\
&\leqslant\ell\left(\sum_{B_j\in\mathcal{B}_{\mathrm{H}}}|B_j|w(B_j,\overline{B_j})+\sum_{B_t\in\mathcal{B}_{\mathrm{L}}}|B_t|\mathrm{vol}(B_t)\right)\\
&\leqslant\ell\left((1+\eta_S)\cdot\frac{k\cdot C_{A.9}+\lambda_{k+1}}{\Upsilon(k)}\cdot\left(\sum_{i=1}^{k}|S_i|\mathrm{vol}(S_i)\right)+\frac{4k^2\cdot C_{A.9}\cdot\eta_S}{3\Upsilon(k)}\cdot\left(\sum_{i=1}^{k}|S_i|\mathrm{vol}(S_i)\right)\right) \qquad (56)\\
&\leqslant\ell\left(\frac{2\eta_S\cdot 4k\cdot C_{A.9}}{3\Upsilon(k)}+\frac{4k^2\cdot C_{A.9}\cdot\eta_S}{3\Upsilon(k)}\right)\sum_{i=1}^{k}|S_i|\mathrm{vol}(S_i)\\
&\leqslant\ell\left(\frac{4k^2\cdot C_{A.9}\cdot\eta_S}{\Upsilon(k)}\right)\cdot\frac{18\cdot\eta_S}{\Phi_{\mathrm{in}}}\cdot\mathsf{OPT}_G \qquad (57)\\
&\leqslant\frac{4k^4\cdot C_{A.9}}{\Upsilon(k)}\cdot\frac{36\cdot\eta_S^2}{\Phi_{\mathrm{in}}}\cdot\mathsf{OPT}_G
\end{aligned}$$

where (56) follows by (B2) and (B3) for the heavy buckets and (B6) for the light ones, and (57) follows by Lemma A.8. Finally, combining (53), (55) and (57) we conclude that

$$\begin{aligned}
\mathsf{COST}_G\left(\mathcal{T}_{\mathrm{SCB}}\right)&\leqslant\left(1+\frac{k^2\cdot C_{A.9}}{\Upsilon(k)}\right)\cdot\frac{36\cdot\eta_S^2}{\Phi_{\mathrm{in}}}\cdot\mathsf{OPT}_G+\frac{4k^4\cdot C_{A.9}}{\Upsilon(k)}\cdot\frac{36\cdot\eta_S^2}{\Phi_{\mathrm{in}}}\cdot\mathsf{OPT}_G\\
&\leqslant\left(1+\frac{5k^4\cdot C_{A.9}}{\Upsilon(k)}\right)\cdot\frac{36\cdot\eta_S^2}{\Phi_{\mathrm{in}}}\cdot\mathsf{OPT}_G,
\end{aligned}$$

which completes the proof. □

Finally, the following theorem summarises the performance of Algorithm 2.

Theorem C.7 (Formal Statement of Theorem 5.4)**.** *Let $G=(V,E,w)$ graph of n vertices, m edges, and optimal clusters $\{S_i\}_{i=1}^k$ corresponding to $\rho(k)$ satisfying $k=O(\mathrm{poly}\log(n))$, $\Phi_{G[S_i]}\geqslant\Phi_{\mathrm{in}}$ for $1\leqslant i\leqslant k$, and $\Upsilon(k)\geqslant C_{A.9}\cdot k$. Then, there is an $\widetilde{O}(m)$ time algorithm that returns both an HC tree $\mathcal{T}_{\mathrm{SCB}}$ of G, and $\mathsf{COST}\left(\mathcal{T}_{\mathrm{SCB}}\right)$. Moreover, it holds that*

$$\mathsf{COST}_G\left(\mathcal{T}_{\mathrm{SCB}}\right)\leqslant\left(1+\frac{5k^4\cdot C_{A.9}}{\Upsilon(k)}\right)\cdot\frac{36\cdot\eta_S^2}{\Phi_{\mathrm{in}}}\cdot\mathsf{OPT}_G$$

where η_S is an upper bound of $\max_i\left(\Delta(S_i)/\delta(S_i)\right)$.

Proof. The approximation guarantee of the returned tree $\mathcal{T}_{\mathrm{SCB}}$ from Algorithm 2 follows from Lemma C.6. By a similar proof as the proof of Lemma B.9, it holds that the time complexity of Algorithm 2 is $\widetilde{O}(m)$ and the depth of $\mathcal{T}_{\mathrm{SCB}}$ is $O(\mathrm{polylog}\,n)$. Therefore, by Lemma B.8, we can compute $\mathsf{COST}_G(\mathcal{T}_{\mathrm{SCB}})$ in nearly-linear time $\widetilde{O}(m)$.

It remains to deal with our assumption that the algorithm knows the number of clusters k and an upper bound

$$\eta_S\geqslant\max_{1\leqslant i\leqslant k}\frac{\Delta(S_i)}{\delta(S_i)}.$$

If the number of clusters k is unknown, we perform the technique described before and run independent copies of Algorithm 2 with all possible values k' ranging from 1 to $O(\mathrm{polylog}(n))$. By adding an extra $O(\mathrm{polylog}\,n)=\widetilde{O}(1)$ factor in the overall time complexity, we ensure that one of the runs has the correct number of clusters $k'=k$.

Finally, in order to obtain an approximation of η_S, we additionally run $O(\log(n))$ independent copies of Algorithm 2 with different values $\widetilde{\eta_S} = 2^i$, for all $i \in [\log \delta_G, \log \Delta_G]$. This process ensures that, at least one such estimate satisfies that $\widetilde{\eta_S}/2 \leqslant \eta_S \leqslant \widetilde{\eta_S}$. By introducing a factor of $O(\log n)$ to the overall running time, this ensures that at least one of the constructed trees of Algorithm 2 satisfies the promised approximation ratio, up to an additional factor of $(\eta_S/\widetilde{\eta_S})^2$. As a result, we return the tree with the minimum cost among all different runs of our algorithm. □