Hierarchical Clustering: O(1)-Approximation for Well-Clustered Graphs

Citation for published version:
Hierarchical Clustering: 
\(O(1)\)-Approximation for Well-Clustered Graphs

Bogdan-Adrian Manghiuc† He Sun‡

Abstract

Hierarchical clustering studies a recursive partition of a data set into clusters of successively smaller size, and is a fundamental problem in data analysis. In this work we study the cost function for hierarchical clustering introduced by Dasgupta [Das16], and present two polynomial-time approximation algorithms: Our first result is an \(O(1)\)-approximation algorithm for graphs of high conductance. Our simple construction bypasses complicated recursive routines of finding sparse cuts known in the literature (e.g., [CAKMT19, CC17]). Our second and main result is an \(O(1)\)-approximation algorithm for a wide family of graphs that exhibit a well-defined structure of clusters. This result generalises the previous state-of-the-art [CAKMT17], which holds only for graphs generated from stochastic models. The significance of our work is demonstrated by the empirical analysis on both synthetic and real-world data sets, on which our presented algorithm outperforms the previously proposed algorithm for graphs with a well-defined cluster structure [CAKMT17].

† School of Informatics, University of Edinburgh, UK. b.a.manghiuc@sms.ed.ac.uk. This work is supported by an EPSRC Doctoral Training Studentship (EP/R513209/1).
‡ School of Informatics, University of Edinburgh, UK. h.sun@ed.ac.uk. This work is supported by an EPSRC Early Career Fellowship (EP/T00729X/1).
1 Introduction

Hierarchical clustering (HC) studies a recursive partition of a data set into clusters of successively smaller size, via an effective binary tree representation. As a basic technique, hierarchical clustering has been employed as a standard package in data analysis, and has comprehensive applications in practice. While traditionally HC trees are constructed through bottom-up (agglomerative) heuristics, which lacked a clearly-defined objective function, Dasgupta [Das16] has recently introduced a simple objective function to measure the quality of a particular hierarchical clustering and his work has inspired a number of research on this topic [AAV20, CAKMT17, CAKMTM19, CC17, CCN19, CCNY19, MW17, RP17]. Consequently, there has been a significant interest in studying efficient HC algorithms that not only work in practice, but also have proven theoretical guarantees with respect to Dasgupta’s cost function.

1.1 Our contribution

We present two new approximation algorithms for constructing HC trees that can be rigorously analysed with respect to Dasgupta’s cost function. For our first result, we construct an HC tree of an input graph $G$ entirely based on the degree sequence of $V(G)$, and show that the approximation guarantee of our constructed tree is with respect to the conductance of $G$, which will be defined formally in Section 2. The striking fact of this result is that, for any $n$-vertex graph $G$ with $m$ edges and conductance $\Omega(1)$ (a.k.a. expander graph), an $O(1)$-approximate HC tree of $G$ can be very easily constructed in $O(m + n \log n)$ time, although obtaining such result for general graphs in polynomial time is impossible under the Small Set Expansion Hypothesis (SSEH) [CC17]. Our theorem is in line with a sequence of results for problems that are naturally linked to the Unique Games and Small Set Expansion problems: it has been shown that such problems are much easier to solve once the input instance exhibits the high conductance property [ABS15, AKK08, Kol10, LSZ19]. However, to the best of our knowledge, our result is the first of this type for hierarchical clustering, and can be informally described as follows:

**Theorem 1** (informal statement of Theorem 3). Given any graph $G = (V, E, w)$ with constant conductance as input, there is an algorithm that runs in $O(|E| + |V| \log(|V|))$ time and returns an $O(1)$-approximate HC tree of $G$.

While our first result presents an interesting theoretical fact, we further study whether we can extend this $O(1)$-approximate construction to a much wider family of graphs occurring in practice. Specifically, we look at well-clustered graphs, i.e., the graphs in which vertices within each cluster are better connected than vertices between different clusters and the total number of clusters is constant. This includes a wide range of graphs occurring in practice with a clear cluster-structure, and have been extensively studied over the past two decades (e.g., [GT14, KVV04, PSZ17, SZ19, vL07]). As our second and main result, we present an approximation algorithm for well-clustered graphs, and our result is informally described as follows:

**Theorem 2** (informal statement of Theorem 4). Let $G = (V, E, w)$ be a well-clustered graph with $O(1)$ clusters. Then, there is a polynomial-time algorithm that constructs an $O(1)$-approximate HC tree of $G$.

Given that the class of well-clustered graphs includes graphs with clusters of different sizes and asymmetrical internal structure, our result significantly improves the previous state-of-the-art [CAKMT17], which only holds for graphs generated from stochastic models. At the technical level, the design of our algorithm is based on the graph decomposition algorithm presented in [GT14], which is designed to find a good partition of a well-clustered graph. However, our analysis suggests that, in order to obtain an $O(1)$-approximation algorithm, directly applying
their decomposition is not sufficient for our purpose. To overcome this bottleneck, we refine their output decomposition via a pruning technique, and carefully merge the refined parts to construct our final HC tree. In our point of view, our presented stronger graph decomposition procedure might have applications in other settings as well.

To demonstrate the significance of our work, we compare our algorithm against the previous state-of-the-art with similar approximation guarantee [CAKMT17] and well-known linkage heuristics on both synthetic and real-world data sets. Although our algorithm’s performance is marginally better than [CAKMT17] for the graphs generated from the stochastic block models (SBM), the cost of our algorithm’s output is up to 50% lower than the one from [CAKMT17] when the clusters of the input graph have different sizes and some cliques are embedded into a cluster.

1.2 Related work

Our work fits in a line of research initiated by Dasgupta [Das16], who introduced a cost function to measure the quality of an HC tree. Dasgupta proved that a recursive application of the algorithm for the Sparsest Cut problem can be used to construct an $O(\log^{3/2} n)$-approximate HC tree. The approximation factor was first improved to $O(\log n)$ by Roy and Pokutta [RP17]. Charikar and Chatziafratis [CC17] improved Dasgupta’s analysis of the recursive sparsest cut algorithm by establishing the following black-box connection: an $\alpha$-approximate algorithm for the Sparsest Cut problem can be used to construct an $O(\alpha)$-approximate HC tree according to Dasgupta’s cost function. Hence, an $O(\sqrt{\log n})$-approximate HC tree can be computed in polynomial time by using the celebrated result of [ARV09]. For general input instances, it is known to be NP-hard to find an optimal HC tree [Das16], and SSEH-hard to achieve an $O(1)$-approximation with respect to Dasgupta’s cost function [RP17, CC17].

Cohen-Addad et al. [CAKMTM19] analysed the performance of several linkage heuristics (e.g., Average Linkage) for constructing HC trees, and showed that such algorithms could output an HC tree of high cost in the worst case. Moving beyond the worst-case scenario, Cohen-Addad et al. [CAKMT17] studied a hierarchical extension of the SBM and showed that, for graphs generated according to this model, a certain SVD projection algorithm [McS01] together with several linkage heuristics can be applied to construct a $(1+o(1))$-approximate HC tree with high probability. We emphasise that our notion of well-clustered graphs generalises the SBM variant studied in [CAKMT17], and does not assume the rigid hierarchical structure of the clusters.

For another line of related work, Moseley and Wang [MW17] studied the dual objective function and proved that Average Linkage achieves a $(1/3)$-approximation for the new objective. Notice that, although this has received significant attention very recently [AAV20, CCN19, CCNY19, CYL+20, VCC+21], achieving an $O(1)$-approximation is tractable under this alternative objective. This suggests the fundamental difference on the hardness of the problem under different objective functions, and is our reason to entirely focus on Dasgupta’s cost function in this work.

1.3 Organisation

The remaining part of the paper is organised as follows: we introduce the necessary notation about graphs and matrices, the basis of hierarchical clustering and graph partitioning in Section 2. In Section 3, we present and analyse the algorithm of constructing an HC tree for graphs of high conductance. We present and analyse our main algorithm, i.e., the algorithm for constructing $O(1)$-approximate HC trees for well-clustered graphs in Section 4. In Section 5 we present the experimental analysis where we compared our developed algorithm against other algorithms in the literature. We end the paper with several concluding remarks and directions of future work in Section 6.
2 Preliminaries

Throughout the paper, we always assume that \( G = (V, E, w) \) is an undirected graph with \( |V| = n \) vertices, \(|E| = m \) edges and weight function \( w : V \times V \rightarrow \mathbb{R}_{\geq 0} \). For any edge \( e = \{u, v\} \in E \), we write \( w_e \) or \( w_{uv} \) to indicate the similarity weight between \( u \) and \( v \). For a vertex \( u \in V \), we denote its degree by \( d_u \triangleq \sum_{v \in V} w_{uv} \) and we assume that \( w_{\text{max}}/w_{\text{min}} = O(\text{poly}(n)) \), where \( w_{\text{min}}(w_{\text{max}}) \) is the minimum (maximum) edge weight. We will use \( d_{\text{min}}, d_{\text{max}}, \) and \( d_{\text{avg}} \) for the minimum, maximum and average degrees in \( G \) respectively, where \( d_{\text{avg}} \triangleq \sum_{u \in V} d_u/n \). For a nonempty subset \( S \subseteq V \), we define \( G[S] \) to be the induced subgraph on \( S \) and we denote by \( G(S) \) the subgraph \( G[S] \), where self loops are added to vertices \( v \in S \) such that their degrees in \( G \) and \( G(S) \) are the same. For any two subsets \( S, T \subseteq V \), we define the cut value \( w(S, T) \triangleq \sum_{e \in E(S, T)} w_e \), 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 \( \text{vol}_G(S) \triangleq \sum_{u \in S} d_u \), and we write \( \text{vol}(G) \) when referring to \( \text{vol}_G(V) \). Sometimes we drop the subscript \( G \) when it is clear from the context. For any nonempty subset \( S \subseteq V \), we define the conductance of \( S \) by

\[
\Phi_G(S) \triangleq \frac{w(S, V \setminus S)}{\text{vol}(S)}.
\]

Notice that \( \Phi_G(V) = 0 \) and we conventionally choose \( \Phi_G(\emptyset) = 1 \), where \( \emptyset \) is the empty set. Furthermore, we define the conductance of the graph \( G \) by

\[
\Phi_G \triangleq \min_{S \subseteq V} \Phi_G(S),
\]

and we call \( G \) an expander graph if \( \Phi_G = \Omega(1) \).

For a graph \( G = (V, E, w) \), let \( D \in \mathbb{R}^{n \times n} \) be the diagonal matrix defined by \( D_{uu} = d_u \) for all \( u \in V \). We denote by \( A \in \mathbb{R}^{n \times n} \) the adjacency matrix of \( G \), where \( A_{uv} = w_{uv} \) for all \( u, v \in V \). The normalised Laplacian matrix of \( G \) is defined as \( L \triangleq I - D^{-1/2}AD^{-1/2} \), where \( I \) is the \( n \times n \) identity matrix. The normalised Laplacian \( L \) is symmetric and real-valued, therefore it has \( n \) real eigenvalues which we will write as \( \lambda_1 \leq \ldots \leq \lambda_n \). It is known that \( \lambda_1 = 0 \) and \( \lambda_n \leq 2 \) [Chu97].

2.1 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 some graph \( G = (V, E, w) \), and \( N \in \mathcal{T} \) be an arbitrary internal node\(^1\) of \( \mathcal{T} \). We write \( N \to (N_1, N_2) \) to indicate that \( N_1 \) and \( N_2 \) are the children of \( N \). We denote \( \mathcal{T}[N] \) to be the subtree of \( \mathcal{T} \) rooted at \( N \), \( \text{leaves}(\mathcal{T}[N]) \) to be the set of leaf nodes of \( \mathcal{T}[N] \) and \( \text{parent}(N) \) 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 \( \text{leaves}(\mathcal{T}[N]) \). For the ease of presentation, we will 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 [Das16] introduced the cost function defined by

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

where \( u \vee v \) is the lowest common ancestor of \( u \) and \( v \) in \( \mathcal{T} \). Trees that achieve a better hierarchical clustering have a lower cost, and the objective of HC is to construct trees with

\(^1\)We consider any non-leaf node of \( \mathcal{T} \) an internal node. We will always use the term \( \text{node(s)} \) for the nodes of \( \mathcal{T} \) and the term \( \text{vertices} \) for the elements of the vertex set \( V \).
the minimum 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 \( |\text{leaves}(T[u \lor v])| \). We denote by \( \text{OPT}_G \) the minimum cost of any HC tree of \( G \), i.e., \( \text{OPT}_G = \min_T \text{cost}_G(T) \), and use the notation \( T^* \) to refer to an optimal tree achieving the minimum cost. We say that an HC tree \( T \) is an \( \alpha \)-approximate tree if \( \text{cost}_G(T) \leq \alpha \cdot \text{OPT}_G \), for some \( \alpha \geq 1 \).

Sometimes, it is convenient to consider the cost of an edge \( e = \{u,v\} \in E \) in \( T \) as \( \text{cost}_T(e) \triangleq w_e \cdot |\text{leaves}(T[u \lor v])| \), so that we can write
\[
\text{cost}_G(T) = \sum_{e \in E} \text{cost}_T(e).
\]

Alternatively, as observed by Dasgupta [Das16], the cost function can be expressed with respect to all cut values induced at every internal node.

**Lemma 2.1** ([Das16]). The cost function of an HC tree \( T \) of \( G \) can be written as
\[
\text{cost}_G(T) = \sum_{N \in T} \left| \text{leaves}(T[N]) \right| \cdot w(\text{leaves}(T[N]), \text{leaves}(T[N_1]), \text{leaves}(T[N_2])),
\]
where the sum is taken over all internal nodes \( N \).

The following lemma presents a simple upper bound on the cost of any HC tree \( T \):

**Lemma 2.2.** It holds for any HC tree \( T \) of \( G \) that
\[
\text{cost}_G(T) \leq \frac{n \cdot \text{vol}(G)}{2}.
\]

**Proof.** For any HC tree \( T \) of \( G \), we have that
\[
\text{cost}_G(T) = \sum_{e \in E} \text{cost}_T(e) \leq \sum_{e \in E} n \cdot w_e \leq \frac{n \cdot \text{vol}(G)}{2},
\]
which proves the statement. \( \square \)

### 2.2 Graph partitioning

The following results on graph partitioning will be used in our analysis, and we list them here for completeness.

**Lemma 2.3** (Cheeger Inequality, [Alo86]). It holds for any graph \( G \) that
\[
\frac{\lambda_2}{2} \leq \Phi_G \leq \sqrt{2\lambda_2}.
\]
Furthermore, there is a nearly-linear time algorithm\(^2\) (i.e., the Spectral Partitioning algorithm) that finds a set \( S \) such that \( \text{vol}(S) \leq \frac{\text{vol}(V)}{2} \), and \( \Phi_G(S) \leq 2 \cdot \sqrt{\Phi_G} \).

One can generalise the notion of conductance, and for any \( k \geq 2 \) define the \( k \)-way expansion of \( G \) by
\[
\rho(k) \triangleq \min_{\text{disjoint } S_1, \ldots, S_k} \max_{1 \leq i \leq k} \Phi_G(S_i).
\]

\( ^2 \)We say that a graph algorithm runs in nearly-linear time if the algorithm runs in \( O(m \cdot \text{polylog } n) \) time, where \( m \) and \( n \) are the number of edges and vertices of the input graph.
Lemma 2.4 (Higher-Order Cheeger Inequality, [LGT14]). It holds for any graph $G$ and $k \geq 2$ that

$$\frac{\lambda_k}{2} \leq \rho(k) \leq O(k^2)\sqrt{\lambda_k}.$$ 

Lemma 2.5 (Lemma 1.13, [GT14]). There is a universal constant $c_0 > 1$ such that for any $k \geq 2$ and any partitioning of $V$ into $r$ sets $P_1, \ldots, P_r$ of $V$, where $r \leq k - 1$, we have that

$$\min_{1 \leq i \leq r} \lambda_2(G[P_i]) \leq 2c_0 \cdot k^6 \cdot \lambda_k.$$ 

3 Hierarchical clustering for graphs of high conductance

In this section we study hierarchical clustering for graphs with high conductance and prove that, for any input graph $G$ with $\Phi_G = \Omega(1)$, an $O(1)$-approximate HC tree of $G$ can be simply constructed based on the degree sequence of $G$. This section is organised as follows: in Section 3.1, we give an upper bound for the cost of any HC tree $T$ based on the degree distribution and conductance of $G$. In Section 3.2, we present an algorithm for constructing an HC tree for graphs of high conductance, whose analysis is presented in Section 3.3.

3.1 Upper bounding Cost$_G(T)$ with respect to the degrees of $G$

As a starting point, we show that Cost$_G(T)$ for any $T$ can be upper bounded with respect to $\Phi_G$ and the degree distribution of $V(G)$.

Lemma 3.1. It holds for any HC tree $T$ of graph $G$ that Cost$_G(T) \leq \frac{9}{4\Phi_G} \cdot \min \left\{ \frac{d_{\text{avg}}}{d_{\text{min}}}, \frac{d_{\text{max}}}{d_{\text{avg}}} \right\} \cdot \text{OPT}_G.$

The proof is based on a combination of Lemma 2.2 and the following technical result:

Lemma 3.2. It holds for any optimal HC tree $T^*$ of graph $G$ that

$$\text{Cost}_G(T^*) \geq \frac{2\Phi_G}{9} \cdot \max \left\{ \frac{\text{vol}(G)^2}{d_{\text{max}}}, d_{\text{min}} \cdot n^2 \right\} = \frac{2\Phi_G}{9} \cdot n \cdot \text{vol}(G) \cdot \max \left\{ \frac{d_{\text{avg}}}{d_{\text{max}}}, \frac{d_{\text{min}}}{d_{\text{avg}}} \right\}.$$

We remark that, as a corollary, the lower bound above holds for any HC tree $T$ of $G$.

Proof of Lemma 3.2. We will give two lower bounds for Cost$_G(T^*)$. Let $A_0$ be the root of $T^*$. For the first lower bound, we start with the root node $A_0$ and travel along $T^*$ recursively as follows: at every intermediate node $A_i$, we travel down to the node of higher volume among its two children. This process stops when we reach node $A_k$ such that vol$(A_k) \leq \frac{2\text{vol}(G)}{3}$. We denote $(A_0, \ldots, A_k)$ to be the path in $T^*$ from the root $A_0$ to $A_k$ and we define $A \triangleq A_k$ as well as $B \triangleq V \setminus A_k$. By construction, it holds that vol$(A) > \text{vol}(G)/3$ and vol$(B) \geq \text{vol}(G)/3$. We will show that the cut $(A, B)$ has significant contribution to cost$_G(T^*)$.

By the stopping criteria, we know that vol$(A_i) > \frac{2\text{vol}(G)}{3}$ for every $0 \leq i < k$. For any edge $e = \{u, v\}$ in the cut $(A, B)$, if $A_i = u \cup v$ for some $0 \leq i < k$, then we have that

$$|\text{leaves}(T^*[u \cup v])| = |A_i| = \frac{\text{vol}(A_i)}{d_{\text{avg}}(A_i)} > \frac{2 \cdot \text{vol}(G)}{3 \cdot d_{\text{max}}}.$$ 

Therefore, we have that

$$\text{Cost}_G(T^*) = \sum_{e = \{u, v\}} w_e \cdot |\text{leaves}(T^*[u \cup v])| \geq \sum_{e \in E(A, B)} w_e \cdot |\text{leaves}(T^*[u \cup v])| \geq \frac{2}{3} \cdot \text{vol}(G) \cdot w(A, B) \geq \frac{2}{3} \cdot \text{vol}(G) \cdot \Phi_G \cdot \min\{\text{vol}(A), \text{vol}(B)\}.$$
\[
\begin{align*}
\geq & \frac{2}{3} \cdot \frac{\text{vol}(G)}{d_{\text{max}}} \cdot \Phi_G \cdot \frac{\text{vol}(G)}{3} \\
= & \frac{2\Phi_G \cdot \text{vol}(G)^2}{9 \cdot d_{\text{max}}}.
\end{align*}
\]

The second lower bound is proven in a similar way: we start with the root node \(A_0\), and travel along \(T^*\) recursively as follows: at every intermediate node \(A'_i\) we travel down to the node of larger size; this process stops when we reach node \(A'_\ell\) such that \(|A'_\ell| \leq 2n/3\). We denote \((A_0 = A'_0, \ldots, A'_\ell)\) to be the path in \(T^*\) from the root to \(A'_\ell\), and we define \(A' = A'_\ell\) as well as \(B' \triangleq V \setminus A'_\ell\). By construction, it holds that \(|A'| > n/3\), and \(|B'| > n/3\). We will show that the cut \((A', B')\) has significant contribution to \(\text{cost}_G(T^*)\).

Similar to the analysis in the first case, by the stopping criteria we have that \(|A'_i| > \frac{2n}{3}\), for all \(0 \leq i < \ell\). Moreover, for any \(e = (u, v) \in E(A', B')\), we have that \(|\text{leaves}(T^*[u \lor v])| > \frac{2n}{3}\).

Hence, it holds that
\[
\begin{align*}
\text{cost}_G(T^*) &= \sum_{e=(u,v)} w_e \cdot |\text{leaves}(T^*[u \lor v])| \\
&\geq \frac{2n}{3} \cdot w(A, B) \\
&\geq \frac{2n}{3} \cdot \Phi_G \cdot \min\{\text{vol}(A), \text{vol}(B)\} \\
&= \frac{2\Phi_G \cdot n^2}{9} \cdot \frac{d_{\text{min}} \cdot n}{3} \\
&= \frac{2\Phi_G \cdot n^2}{9} \cdot \frac{d_{\text{min}}}{3}.
\end{align*}
\]

Combining the two cases above gives us that
\[
\text{cost}_G(T^*) \geq \frac{2\Phi_G}{9} \cdot \max \left\{ \frac{\text{vol}(G)^2}{d_{\text{max}}}, \frac{d_{\text{min}} \cdot n^2}{3} \right\},
\]
which proves the statement.

\[\square\]

**Proof of Lemma 3.1.** Let \(T\) be an arbitrary HC tree, and \(T^*\) be an optimal tree. Combining Lemmas 2.2 and 3.2, we have that
\[
\frac{\text{cost}_G(T)}{\text{cost}_G(T^*)} \leq \frac{n \cdot \text{vol}(G)}{2} \cdot \frac{9}{2\Phi_G \cdot n \cdot \text{vol}(G) \cdot \max \left\{ \frac{d_{\text{avg}}}{d_{\text{max}}}, \frac{d_{\text{min}}}{d_{\text{avg}}} \right\}}
\]
\[
= \frac{9}{4} \cdot \Phi_G \cdot \min \left\{ \frac{d_{\text{avg}}}{d_{\text{min}}}, \frac{d_{\text{max}}}{d_{\text{avg}}} \right\},
\]
which proves the statement.

\[\square\]

### 3.2 The algorithm for graphs of high conductance

While Lemma 3.1 holds for any graph \(G\), it implies some interesting facts for expander graphs: first of all, when \(G = (V, E, w)\) satisfies \(d_{\text{max}}/d_{\text{min}} = O(1)\) and \(\Phi_G = \Omega(1)\), Lemma 3.1 shows that any HC tree \(T\) is an \(O(1)\)-approximate tree. In addition, although \(\Phi_G\) plays a crucial role in analysing \(\text{cost}_G(T)\) as for many other graph problems, Lemma 3.1 indicates that the degree distribution of \(G\) might also have a significant impact. One could naturally ask the extend to which the degree distribution of \(V(G)\) would influence the construction of \(\text{OPT}_G\). To answer this question, we study the following example.

**Example 3.3.** We study the following graph \(G\), in which all the edges have unit weight:

1. Let \(G_1 = (V, E_1)\) be a constant-degree expander graph of \(n\) vertices with \(\Phi_{G_1} = \Omega(1)\), e.g., the ones presented in [HLW06];
2. We choose \( \lfloor n^{2/3} \rfloor \) vertices from \( V \) to form \( S \), and let \( K = (S, S \times S) \) be a complete graph defined on \( S \);

3. Partition the vertices of \( V \setminus S \) into \( \lfloor n^{2/3} \rfloor \) groups of roughly the same size, associate each group to a unique vertex in \( S \), and let \( E_2 \) be the set of edges formed by connecting every vertex in \( S \) with all the vertices in its associated group;

4. We define \( G = (V, E_1 \cup (S \times S) \cup E_2) \), see Figure 1(a) for illustration.

By construction, we know that \( \Phi_G = \Omega(1) \), and the degrees of \( G \) satisfy \( d_{\text{max}} = \Theta(n^{2/3}) \), \( d_{\text{min}} = \Theta(1) \), and \( d_{\text{avg}} = \Theta(n^{1/3}) \) as \( \text{vol}(G) = \Theta(n^{4/3}) \). Therefore, the ratio between \( \text{cost}_G(T) \) for any \( HC \) tree \( T \) and \( \text{OPT}_G \) could be as high as \( \Theta(n^{1/3}) \). On the other side, it is not difficult to show that the tree \( T^* \) illustrated in Figure 1(b), which first separates the set \( S \) of high-degree vertices from \( V \setminus S \) at the top of the tree, actually \( O(1) \)-approximates \( \text{OPT}_G \). To see this, notice that \( \text{cost}(T^*) \leq \text{cost}(T^*[S]) + n \cdot \text{vol}(G_1) + n^2 = \Theta(n^2) \), as the complete subgraph \( G[S] \) induces a cost of \( \Theta((n^{2/3})^3) \) [Das16]. The existence of the subgraph \( G[S] \) also implies that \( \text{OPT}_G = \Omega(n^2) \).

![Figure 1: (a) Our constructed graph \( G \); (b) the tree that separates the vertices of high degrees from the others achieves \( O(1) \)-approximation.](image)

This example suggests that grouping vertices of similar degrees first would potentially help reduce \( \text{cost}_G(T) \) for our constructed \( T \). This motivates us to design the following Algorithm 1 to construct an \( HC \) tree, and the algorithm’s performance is summarised in Theorem 3. We highlight that the output of Algorithm 1 is uniquely determined by the ordering of the vertices of \( G \) according to their degrees, which can be computed in \( O(n \cdot \log n) \) time.

**Algorithm 1:** \( HC\text{withDegrees}(G\{V\}) \)

**Input:** \( G = (V, E, w) \) with the ordered vertices such that \( d_{v_1} \geq \ldots \geq d_{v_{|V|}} \);

**Output:** An \( HC \) tree \( T_{\text{deg}}(G) \);

1. if \( |V| = 1 \) then
   2. return the single vertex in \( V \) as the tree;
2. else
   4. \( i_{\text{max}} := \lfloor \log_2(|V| - 1) \rfloor \); \( r := 2^{i_{\text{max}}} \); \( A := \{v_1, \ldots, v_r\} \); \( B := V \setminus A \);
   5. Let \( T_1 := HC\text{withDegrees}(G\{A\}); T_2 := HC\text{withDegrees}(G\{B\}); \)
   6. return \( T_{\text{deg}} \) with \( T_1 \) and \( T_2 \) as the two children.

**Theorem 3.** Given any graph \( G = (V, E, w) \) with conductance \( \Phi_G \) as input, Algorithm 1 runs in \( O(m + n \cdot \log n) \) time, and returns an \( HC \) tree \( T_{\text{deg}} \) of \( G \) that satisfies \( \text{cost}_G(T_{\text{deg}}) = O(1/\Phi_G^3) \cdot \text{OPT}_G \).

Theorem 3 shows that, when the input \( G \) satisfies \( \Phi_G = \Omega(1) \), the output \( T_{\text{deg}} \) of Algorithm 1 achieves an \( O(1) \)-approximation. It is important to notice that, while it is known [CC17] that there is no polynomial-time algorithm that \( O(1) \)-approximates \( \text{OPT}_G \) for a general graph \( G \) under the SSEH, our result shows that an \( O(1) \)-approximate \( HC \) tree can be constructed in polynomial-time for expander graphs. Our result is in line with a sequence of research showing that this type of problems become easier when the input graphs exhibit a good expansion.
property (e.g., [ABS15, AKK+08, Kol10, LSZ19]). To the best of our knowledge, Theorem 3 is the first such result for hierarchical clustering on graphs of high expansion conductance. Moreover, as the high-conductance property can be determined in nearly-linear time by computing $\lambda_2(L_G)$ and applying the Cheeger inequality, Algorithm 1 presents a very simple construction of an $O(1)$-approximate HC tree once the input $G$ is known to have high conductance.

3.3 Analysis of the algorithm

In this subsection we analyse Algorithm 1, and prove Theorem 3.

3.3.1 The dense branch and its properties

Our analysis is crucially based on the notion of dense branch, which can be informally described as follows: for any given $T$, we perform a traversal in $T$ starting at its root node $A_0$ and sequentially travel to the child of higher volume. The process stops whenever we reach a node $A_k$, for some $k \in \mathbb{Z}_{\geq 0}$, such that $\text{vol}(A_k) > \text{vol}(G)/2$ and both of its children have volume at most $\text{vol}(G)/2$. The sequence of visited nodes in this process is the dense branch of $T$. Formally, we define the dense branch as follows:

**Definition 3.4** (Dense branch). Given a graph $G$ with an HC tree $T$, the dense branch is the path $(A_0, A_1, \ldots, A_k)$ in $T$, for some $k \in \mathbb{Z}_{\geq 0}$, such that the following hold:

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

It is important to note that the dense branch of $T$ is unique, and consists of all the nodes $A_i$ with $\text{vol}(A_i) > \text{vol}(G)/2$. 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. Now we will present some properties of the dense branch, which will be used extensively in our analysis.

**Lemma 3.5** (Lower bound of $\text{cost}_G(T)$ based on the dense branch). Let $G$ be a graph of conductance $\Phi_G$, and let $T$ be an arbitrary HC tree of $G$. Suppose $(A_0, \ldots, A_k)$ is the dense branch of $T$, for some $k \in \mathbb{Z}_{\geq 0}$, and suppose each node $A_i$ has sibling $B_i$, for all $1 \leq i \leq k$. Then, the following lower bounds of $\text{cost}_G(T)$ hold:

1. $\text{cost}_G(T) \geq \frac{\Phi_G}{2} \sum_{i=1}^{k} |A_{i-1}| \cdot \text{vol}(B_i)$;
2. $\text{cost}_G(T) \geq \frac{\Phi_G}{2} \cdot |A_k| \cdot \text{vol}(A_k)$.

**Proof.** Let $S = \{A_k, B_1, \ldots, B_k\}$. We focus on the edges $e$ crossing different pairs $(X,Y)$ of $X, Y \in S$, and have that

$$\text{cost}_G(T) \geq \sum_{(X,Y) \in S^2} \sum_{X \neq Y} \text{cost}_T(e) = \frac{1}{2} \sum_{X \in S} \sum_{e \in E(X,V \setminus X)} \text{cost}_T(e)$$

$$\geq \frac{1}{2} \sum_{i=1}^{k} \sum_{e \in E(B_i, V \setminus B_i)} \text{cost}_T(e) \geq \frac{1}{2} \sum_{i=1}^{k} \sum_{e \in E(B_i, V \setminus B_i)} w_e \cdot |A_{i-1}|$$

$$= \frac{1}{2} \sum_{i=1}^{k} |A_{i-1}| \cdot w(B_i, V \setminus B_i) \geq \frac{\Phi_G}{2} \cdot \sum_{i=1}^{k} |A_{i-1}| \cdot \text{vol}(B_i),$$

where the third inequality holds by the fact that $A_{i-1}$ is the parent of $B_i$ and any edge $e$ with exactly one endpoint in $B_i$ satisfies $\text{cost}_T(e) \geq w_e \cdot |A_{i-1}|$, and the last inequality holds by the fact that $G$ has conductance $\Phi_G$ and $\text{vol}(B_i) \leq \text{vol}(G)/2$ by the definition of the dense branch.
For the second bound, let \( A_{k+1}, B_{k+1} \) be the children of \( A_k \) such that \( \text{vol}(A_{k+1}) \geq \text{vol}(B_{k+1}) \). Therefore, we have that
\[
\text{cost}_G(T) \geq \sum_{e \in E(A_k, V \setminus A_k)} \text{cost}_T(e) \geq \sum_{e \in E(A_k, V \setminus A_k)} w_e \cdot |A_k| \\
\geq |A_k| \cdot w(A_k, V \setminus A_k) \geq |A_k| \cdot \Phi_G \cdot \text{vol}(A_k) \\
\geq \frac{\Phi_G}{2} \cdot |A_k| \cdot \text{vol}(A_k).
\]

Next, we prove that, if the dense branch of an optimal tree \( T^* \) consists of a single node, i.e., the root of \( T^* \), then any tree \( T \) achieves a \((1/\Phi_G)\)-approximation. If we assume this corner case occurs, we are able to prove a stronger result over Theorem 3, and this result is presented in the following lemma.

**Lemma 3.6.** Let \( G \) be a graph of conductance \( \Phi_G \), and let \( T^* \) be any optimal HC tree of \( G \) such that its dense branch only consists of the root of \( T^* \). Then, it holds for any HC tree \( T \) of \( G \) that
\[
\text{cost}_G(T) \leq \frac{\text{cost}_G(T^*)}{\Phi_G}.
\]
In particular, it implies that \( \text{cost}_G(T_{\text{deg}}) \leq 1/\Phi_G \cdot \text{OPT}_G \).

**Proof.** Suppose the dense branch of \( T^* \) only consists of the root node \( A_0 \), and let \( A_1, B_1 \) be the two children of \( A_0 \) such that \( \text{vol}(A_1) \geq \text{vol}(B_1) \). Since \( A_1 \) does not belong to the dense branch, we know that \( \text{vol}(A_1) \leq \text{vol}(G)/2 \) and hence \( \text{vol}(A_1) = \text{vol}(B_1) = \text{vol}(G)/2 \). Therefore, we have that
\[
\text{cost}_G(T^*) \geq \sum_{e \in E(A_1, B_1)} \text{cost}_T(e) = n \cdot w(A_1, B_1) \geq n \cdot \Phi_G \cdot \frac{\text{vol}(G)}{2},
\]
where the last inequality uses that \( G \) has conductance \( \Phi_G \) and \( \text{vol}(A_1) = \text{vol}(B_1) = \text{vol}(G)/2 \).

On the other hand, by Lemma 2.2 it holds for any tree \( T \) that
\[
\text{cost}_G(T) \leq n \cdot \frac{\text{vol}(G)}{2}.
\]
Combining the two inequalities above proves the claimed statement. \( \square \)

Finally, we prove that if the size of the last node on the dense branch of an optimal tree \( T^* \) is significantly large, then any tree \( T \) achieves an \( O(1/\Phi_G) \)-approximation. Again, if we assume this corner case occurs, we prove a stronger result over Theorem 3 and we present the result in the following lemma.

**Lemma 3.7.** Let \( G \) be a graph of conductance \( \Phi_G \), and let \( T^* \) be any optimal HC tree of \( G \) whose dense branch is \( (A_0, \ldots, A_k) \), for some \( k \in \mathbb{Z}_{\geq 0} \). If \( |A_k| \geq (n - 1)/2 \), then it holds for any HC tree \( T \) of \( G \) that
\[
\text{cost}_G(T) \leq \frac{8 \cdot \text{cost}_G(T^*)}{\Phi_G}.
\]
In particular, it implies that \( \text{cost}_G(T_{\text{deg}}) \leq 8/\Phi_G \cdot \text{OPT}_G \).

**Proof.** We apply the second property of Lemma 3.5 for the tree \( T^* \), and have that
\[
\text{cost}_G(T^*) \geq \frac{\Phi_G}{2} \cdot |A_k| \cdot \text{vol}(A_k) \geq \frac{\Phi_G}{2} \cdot \frac{n - 1}{2} \cdot \frac{\text{vol}(G)}{2} \geq \frac{\Phi_G}{8} \cdot \frac{n \cdot \text{vol}(G)}{2},
\]
where the second inequality uses that \( \text{vol}(A_k) > \text{vol}(G)/2 \) and our assumption on the size of \( A_k \). On the other hand, by Lemma 2.2 it holds for any tree \( T \) that
\[
\text{cost}_G(T) \leq n \cdot \frac{\text{vol}(G)}{2}.
\]
Combining the two inequalities above proves the claimed statement. \( \square \)
3.3.2 Proof of Theorem 3

Now we prove the main result of this section, i.e., Theorem 3. To sketch the main proof ideas, we start with an optimal HC tree $T_0$ of $G$, and construct trees $T_1, T_2, T_3$ and $T_4$ with the following properties:

1. $\text{cost}_G(T_i)$ can be upper bounded with respect to $\text{cost}_G(T_{i-1})$ for every $1 \leq i \leq 4$;
2. The final constructed tree $T_4$ is exactly the tree $T_{\text{deg}}(G)$, the output of Algorithm 1.

Combining these two facts allows us to upper bound $\text{cost}_G(T_{\text{deg}})$ with respect to $\text{OPT}_G$. In the remaining part of the section, we assume that the dense branch of the optimal tree $T_0$ contains at least two nodes, since otherwise one can apply Lemma 3.6 to obtain a stronger result than Theorem 3. Moreover, if $A_{k_0}$ is the last node on the dense of $T_0$ we assume that $|A_{k_0}| < (n-1)/2$, as otherwise by Lemma 3.7 we again obtain a stronger result than Theorem 3.

**Step 1: Regularisation.** Let $(A_0, \ldots, A_k)$, for some $k_0 \in Z^+$, be the dense branch of $T_0$, and let $B_i$ be the sibling of $A_i$, for all $1 \leq i \leq k_0$. Let $i_{\text{min}} = \lfloor \log_2 |A_{k_0}| \rfloor$, $i_{\text{max}} = \lfloor \log_2 (|V| - 1) \rfloor$, and notice that $i_{\text{min}} < i_{\text{max}}$. The goal of this step is to adjust the dense branch of $T_0$, such that the resulting tree $T_1$ satisfies the following conditions:

1. $|A_1| \geq 2^{i_{\text{max}}}$;
2. For all $i \in (i_{\text{min}}, i_{\text{max}}]$, there is a node of size exactly $2^i$ along the dense branch of $T_1$;
3. If $|A_{k_0}| > 2^{i_{\text{min}}}$, then $A_{k_0}$ has a child of size $2^{i_{\text{min}}}$.

We deal with each condition individually, and we start with the first one. If the first condition is not already satisfied, we perform an initial adjustment to the tree $T_0$ to ensure that, in the resulting tree $T'_0$ the second node on its dense branch has size $2^{i_{\text{max}}}$, as illustrated in Figure 2(a). Specifically, we consider an arbitrary partition of $B_1 = B_1^1 \cup B_1^2$ such that $|A_1| + |B_1^1| = 2^{i_{\text{max}}}$. We adjust $T_0$ as follows: we set the two children of $A_0$ in $T'_0$ as some newly created node $A'_1$ and $B_1^2$, and set the two children of $A'_1$ as $A_1$ and $B_1^1$; the remaining part of $T'_0$ is the same as $T_0$. As such, the dense branch of the new tree $T'_0$ becomes $(A_0, A'_1, A_1, \ldots, A_{k_0})$, and $|A'_1| = 2^{i_{\text{max}}}$ by construction.

To obtain the tree satisfying the second condition in the mean time, we apply a sequence of adjustments, each of which creates a new node of exact size $2^i$ for some suitable $i$. Specifically, let $i \in (i_{\text{min}}, i_{\text{max}}]$ be the largest integer such that there is no node of size $2^i$ on the dense branch of $T'_0$. Since $|A_{k_0}| \geq 2^{i_{\text{min}}}$, there is some node $A_j$ on the dense branch such that $|A_j| > 2^i$ and $|A_{j+1}| < 2^i$. We adjust the branch at $A_j$ as follows: (i) we consider a partition of $B_{j+1} = B_{j+1}^1 \cup B_{j+1}^2$ such that $|A_{j+1}| + |B_{j+1}^2| = 2^i$; (ii) we replace the node $A_j$ by some newly created node $A'_j$ that has children $B_{j+1}^1$ and a new node $A_{j+1}^2$; (iii) the two children of $A_{j}^2$ will be $A_{j+1}$ and $B_{j+1}^2$. This adjustment is illustrated in Figure 2(b), and we repeat this process until no such $i$ exists anymore.

To ensure that the third condition is satisfied (assuming $|A_{k_0}| > 2^{i_{\text{min}}}$), we adjust the dense branch at $A_{k_0}$ in a similar way as before. Specifically, let $A_{k_0+1}, B_{k_0+1}$ be the two children of $A_{k_0}$ such that $A_{k_0+1}$ is the child of smaller size $|A_{k_0+1}| < 2^{i_{\text{min}}}$. We consider an arbitrary partition of $B_{k_0+1} = B_{k_0+1}^1 \cup B_{k_0+1}^2$ such that $|A_{k_0+1}| + |B_{k_0+1}^2| = 2^{i_{\text{min}}}$. We replace the node $A_{k_0}$ with a new node $A'_{k_0}$ that has children $B_{k_0+1}^1$ and a new node $A_{k_0}^2$. The two children of $A_{k_0}^2$ are $A_{k_0+1}$ and $B_{k_0+1}^2$. After this transformation, one of the following two conditions happens: (i) the dense branch increases its size by one, having the final node $A_{k_0}^2$ of size $|A_{k_0}^2| = 2^{i_{\text{min}}}$, or (ii) $A_{k_0}^2$ is the final node on the dense branch having size $|A_{k_0}^1| = |A_{k_0}| > 2^{i_{\text{min}}}$ and $A_{k_0}^1$ has one child $A_{k_0}^2$ of size $|A_{k_0}^2| = 2^{i_{\text{min}}}$. In both cases, the third condition is satisfied.

We call the resulting tree $T_1$, and the following lemma gives an upper bound of $\text{cost}_G(T_1)$.

---

3This follows from the assumption that $|A_{k_0}| < (|A_0| - 1)/2$. 

10
Hence, the total additional cost of the adjustment for all adjustments, including the initial adjustment to transform \( T_0 \) into \( T_1 \), is upper bounded by

\[
\sum_{e \in E} \text{cost}_{T_1}(e) \leq \sum_{e \in E} \text{cost}_{T_0}(e) + \sum_{e \in E \setminus F} \text{cost}_{T_0}(e) \leq \left( \sum_{e \in F} \text{cost}_{T_1}(e) \right) + \text{cost}(T_0). \tag{3.1}
\]

Hence, it suffices to show that

\[
\sum_{e \in F} \text{cost}_{T_1}(e) \leq \frac{4}{\Phi_G} \cdot \text{cost}(T_0),
\]

and we sometimes refer to \( \sum_{e \in F} \text{cost}_{T_1}(e) \) as the additional cost of the transformation from \( T_0 \) to \( T_1 \). We look at the additional cost introduced by one of our adjustments, say at node \( A_j \), as illustrated in Figure 2(b). It is not difficult to show that the cost can increase only for edges \( e \in E(B_{j+1}, B_{j+1}^2) \), for each of which the additional cost is exactly \( \text{cost}_{T_1}(e) = w_e \cdot |A_j^1| = w_e \cdot |A_j| \).

Hence, the total additional cost of the adjustment for \( A_j \) is at most \( |A_j| \cdot w(B_{j+1}, B_{j+1}^2) \), and after all adjustments, including the initial adjustment to transform \( T_0 \) in \( T_0' \) and the final adjustment at node \( A_{k_0} \), the total additional cost is upper bounded by

\[
\sum_{e \in F} \text{cost}_{T_1}(e) \leq \sum_{i=1}^{k_0+1} \sum_{e \in E(B_i, B_i)} w_e \cdot |A_{i-1}| \leq \sum_{i=1}^{k_0+1} |A_{i-1}| \cdot \text{vol}(B_i)
\]

\[
\leq \left( \sum_{i=1}^{k_0} |A_{i-1}| \cdot \text{vol}(B_i) \right) + |A_{k_0}| \cdot \text{vol}(A_{k_0}) \leq \frac{4}{\Phi_G} \cdot \text{cost}(T_0), \tag{3.2}
\]

where the last inequality follows by Lemma 3.5. Combining (3.1) with (3.2) proves the statement.

\( \square \)

**Step 2: Compression.** With potential relabelling of the nodes, let \( (A_0, \ldots, A_{k_1}) \) be the dense branch of \( T_1 \), for some \( k_1 \in \mathbb{Z}_+ \) satisfying \( |A_{k_1}| \geq 2^{\min^+} \), and \( B_i \) be the sibling of \( A_i \). The objective of this step is to ensure that, by a sequence of adjustments, all the nodes along the dense branch are of size equal to some power of 2. In this step we perform two similar types of adjustments: the first type concerns all nodes \( A_j \) of size at least \( 2^{\min^+} \), and the second type
Figure 3: The nodes between $A_{j_1}$ and $A_{j_2}$ (left) have size in $(2^{i-1}, 2^i)$; These nodes are compressed (right) and only the nodes of size some power of 2 remain in the dense branch.

concerns the nodes $A_j$ of size $2^{i_{\text{min}}} \leq |A_j| < 2^{i_{\text{min}}+1}$. We begin by describing how an adjustment of the first type is performed, and refer the reader to Figure 3 for illustration.

Let $i \in (i_{\text{min}} + 1, i_{\text{max}}]$ be some index such that $|A_{j_1}| = 2^i$ and $|A_{j_2}| = 2^{i-1}$ for some $j_1 < j_2$. We compress the dense branch by removing all nodes between $A_{j_1}$ and $A_{j_2}$ as follows: the two children of $A_{j_1}$ will be $A_{j_2}$ and some new node $C_{j_1}$, which has children $C_{j_1+1}$ and $B_{j_1+1}$; the two children of $C_{j_1+1}$ will be some new node $C_{j_1+2}$ and $B_{j_1+2}$, etc. The last node $C_{j_2-2}$ has children $B_{j_2-1}$ and $B_{j_2}$. In addition, we perform one more such adjustment to remove all nodes $A_j$ of size $2^{i_{\text{max}}} < |A_j| < n$.

For the second type of adjustment, we have a case distinction based on the size of the final node $A_{k_1}$. Specifically, if $|A_{k_1}| = 2^{i_{\text{min}}}$, then we perform once more the previous adjustment for the value $i = i_{\text{min}} + 1$. On the other side, if $|A_{k_1}| > 2^{i_{\text{min}}}$, we perform the adjustment as illustrated in Figure 4: let $A_j$ be the node on the dense branch of size $2^{i_{\text{min}}+1}$, which exists since $i_{\text{min}} + 1 \leq i_{\text{max}}$. Also, let $A_{k_1+1}, B_{k_1+1}$ be the two children of $A_{k_1}$ such that $|A_{k_1+1}| = 2^{i_{\text{min}}}$. Similar to before, we compress the dense branch by removing all nodes between $A_j$ and $A_{k_1+1}$ as follows: the two children of $A_j$ will be $A_{k_1+1}$ and a new node $C_j$ which has children $B_{k_1+1}$ and a new node $C_{j+1}$; the two children of $C_{j+1}$ are $B_{j+1}$ and a new node $C_{j+2}$, etc. The last node $C_{k_1-1}$ has children $B_{k_1-1}$ and $B_{k_1}$. Notice that, after this adjustment, the final node on the dense branch is either $C_j$ if $\text{vol}(C_j) > \text{vol}(G)/2$, or $A_j$ otherwise. Since $|C_j| = 2^{i_{\text{min}}}$ and $|A_j| = 2^{i_{\text{min}}+1}$, this ensures that all nodes (potentially except for $A_0$) on the dense branch have size $2^i$ for some $i \in [i_{\text{min}}, i_{\text{max}}]$. We call the resulting tree $T_2$, and the following lemma gives an upper bound of $\text{cost}_G(T_2)$.

**Lemma 3.9.** Our constructed tree $T_2$ satisfies that $\text{cost}_G(T_2) \leq 2 \cdot \text{cost}_G(T_1)$.

**Proof.** We claim that the cost of every edge at most doubles in $T_2$, and this implies that $\text{cost}(T_2) \leq 2 \cdot \text{cost}(T_1)$. To see why our claim holds, let us consider the set

$$S = \{B_1, B_2, \ldots, B_{k_1}, B_{k_1+1}, A_{k_1+1}\},$$

which consists of the sibling of every node on the dense branch of $T_1$, along with the children of $A_{k_1}$. This forms a partition of the vertex set. Now consider any two different internal nodes $X, Y \in S$ such that $|\text{parent}_{T_1}(X)| \geq |\text{parent}_{T_1}(Y)|$, and an arbitrary edge $e \in E$. If
Lemma 3.10. Our constructed tree $T_3$ satisfies that $\text{cost}(T_3) \leq (1 + 4/\Phi_G) \cdot \text{cost}(T_2)$.

Proof. Let $S = \{B_1, \ldots, B_{k_2}, A_{k_2}\}$ be the set of internal nodes of $T_2$ that undergo the above transformation. By construction, for every distinct $X, Y \in S$ and any edge $e \in E(X, Y)$, it holds that $\text{cost}_{T_2}(e) = \text{cost}_{T_3}(e)$. Therefore, the additional cost will be introduced only by edges $e \in E(X, X)$, for some $X \in S$. However, by Lemma 2.2 we have that

$$\sum_{X \in S} \sum_{e \in E(X, X)} \text{cost}_{T_3}(e) \leq \sum_{X \in S} |X| \cdot \text{vol}(X) = \left( \sum_{i=1}^{k_2} |B_i| \cdot \text{vol}(B_i) \right) + |A_{k_2}| \cdot \text{vol}(A_{k_2}) \leq \frac{4}{\Phi_G} \cdot \text{cost}(T_2),$$

where the last inequality follows by Lemma 3.5. Combining the two cases together proves the statement.

Step 4: Sorting. We assume that $(A_0, \ldots, A_{k_3})$ is the dense branch of $T_3$ for some $k_3 \in \mathbb{Z}_+$, and we extend the dense branch to $(A_0, \ldots, A_{i_{\text{max}}})$ with the property that, for every $i \in [k_3, i_{\text{max}}]$, $A_i$ is the child of $A_{i-1}$ with the higher volume, and let $S \triangleq \{B_1, \ldots, B_{i_{\text{max}}}, A_{i_{\text{max}}} \}$. Recall that in $T_{\text{deg}}(G)$ the first $r = 2^{i_{\text{max}}} - 1$ vertices of the highest degrees, i.e., $\{v_1, \ldots, v_r\}$ belong to $A_1$, of which the first $2^{i_{\text{max}}-1}$ vertices belong to $A_2$, and so on; however, this might not be the case for $T_3$. Hence, in the final step, we prove that $T_3$ can be transformed into $T_{\text{deg}}(G)$ without a
Let \( e \) be the vertex of the highest degree that is currently misplaced. For the sake of argument, we assume that \( v_1 \not\in A_{\text{imax}} \) is misplaced, and we apply the following operation to move \( v_1 \) to \( A_{\text{imax}} \):

1. Let \( v_1 \in B_{i_0} \) for some \( i_0 \geq 1 \), and let \( y \) be the vertex of the lowest degree among the vertices in \( S \setminus \{B_1, \ldots, B_{i_0}\} \). Say \( y \in B_i \) for some \( i_1 > i_0 \), and swap \( v_1 \) with \( y \) while keeping the structure of the tree unchanged.

2. Repeat the swap operation above until \( v_1 \) reaches its correct place in \( A_{\text{imax}} \).

Once the above process is complete and \( v_1 \) reaches \( A_{\text{imax}} \), we apply a similar chain of swaps for \( v_2 \) to ensure \( v_2 \) also reaches \( A_{\text{imax}} \). Then, we sequentially apply the process for \( v_3 \) and \( v_4 \) to ensure they reach \( B_{i_{\text{max}}} \), and continue this process until there are no more misplaced vertices.

We call the resulting tree \( T' \), and notice that every node \( X \in S \) in \( T' \) contains the correct set of vertices. However, the positions of these vertices in \( T' \setminus \{X\} \) might be different from the ones in \( T_{\text{deg}}(G\{X\}) \). To overcome this issue, we repeat Step 3 again to the tree \( T' \), and this will introduce another factor of \((1 + 4/\Phi_G)\) to the total cost of the constructed tree. Importantly, the final constructed tree after this step is exactly the tree \( T_{\text{deg}}(G) \) and we bound its cost by the following lemma:

**Lemma 3.11.** It holds for \( T_{\text{deg}} \) that \( \text{cost}(T_{\text{deg}}) \leq (1 + 24/\Phi_G) (1 + 4/\Phi_G) \cdot \text{cost}(T_3) \).

**Proof.** We first prove that

\[
\text{cost}(T'_3) \leq \left(1 + \frac{24}{\Phi_G}\right) \cdot \text{cost}(T_3).
\]

For the sake of analysis, we assume that \( v_1 \in B_{i_0} \) is misplaced, and let \( (y_1, \ldots, y_t) \), for some \( t \in \mathbb{Z}^+ \), be the sequence of vertices with which \( v_1 \) performs the swap operations in order to reach \( A_{\text{imax}} \). We first upper bound the additional cost that the swap between \( y_1 \) and \( v_1 \) introduces. Without loss of generality, we assume that \( y_1 \in B_{i_1} \) such that \( |\text{parent}_{T_3}(B_{i_0})| > |\text{parent}_{T_3}(B_{i_1})| \).

Let \( e \in E(G) \) be any edge. Our analysis is based on the following case distinction:

- If \( e \cap \{y_1, v_1\} = \emptyset \), then the swap between \( y_1 \) and \( v_1 \) would not change the cost of \( e \);
- If \( e = \{y_1, v_1\} \), then the swap between \( y_1 \) and \( v_1 \) would not change the cost of \( e \) either;
- If \( e \) is adjacent to \( y_1 \) or \( v_1 \), the cost of \( e \) would increase by at most
  \[
  w_e \cdot |\text{parent}_{T_3}(B_{i_0})| = w_e \cdot |A_{i_0-1}|.
  \]

Hence, the total additional cost of the swap between \( y_1 \) and \( v_1 \) is at most

\[
|A_{i_0-1}| \cdot (d_{v_1} + d_{y_1}) \leq 2 \cdot |A_{i_0-1}| \cdot d_{v_1},
\]

as \( v_1 \) is the misplaced vertex of largest degree.

We apply the same analysis for the swap between \( v_1 \) and \( y_2 \in B_{i_2} \) for some \( i_2 \geq i_1 \), implying that the additional cost by the swap between \( y_2 \) and \( v_1 \) is at most \( 2 \cdot |A_{i_1-1}| \cdot d_{v_1} \), and so on. Therefore, the total additional cost introduced in order for \( v_1 \) to reach \( A_{\text{imax}} \) is at most

\[
2 \cdot d_{v_1} \cdot (|A_{i_0-1}| + \ldots + |A_{i_{t-1}-1}|) \leq 6 \cdot d_{v_1} \cdot |A_{i_0-1}|.
\]
Since we only need to consider all the misplaced vertices in some $B_j$, the total additional cost introduced over all sequences of swaps is at most

$$6 \cdot \sum_{i=1}^{i_{\max}} \text{vol}(B_i)|A_{i-1}| \leq \frac{24}{\Phi_G} \cdot \text{cost}(T_3),$$

where the last inequality follows by Lemma 3.5. In summary, we have that

$$\text{cost}\left(T'_3\right) \leq \left(1 + \frac{24}{\Phi_G}\right) \text{cost}(T_3).$$

Since we use Step 3 in the end, we apply Lemma 3.10 once more and obtain the claimed statement.

Proof of Theorem 3. By combining Lemmas 3.8, 3.9, 3.10 and 3.11, we prove the approximation guarantee of $T_{\text{deg}}$ in Theorem 3. The runtime of Algorithm 1 follows by a simple application of the master theorem.

4 Hierarchical clustering for well-clustered graphs

So far we have shown that an $O(1)$-approximate HC tree can be easily constructed for expander graphs. In this section we study a wider class of graphs that exhibit a clear structure of clusters, i.e., well-clustered graphs. Informally, a well-clustered graph is a collection of densely-connected components (clusters) of high conductance, which are weakly interconnected. As these graphs form some of the most meaningful objects for clustering in practice, one would naturally ask whether our $O(1)$-approximation result for expanders can be extended to well-clustered graphs. In this section, we will give an affirmative answer to this question.

To formalise the well-clustered property, we consider the notion of $(\Phi_{\text{in}}, \Phi_{\text{out}})$-decomposition introduced by Gharan and Trevisan [GT14]. Formally, for a graph $G = (V, E, w)$ and $k \in \mathbb{Z}_+$, we say that $G$ has $k$ well-defined clusters if $V(G)$ can be partitioned into disjoint subsets $\{P_i\}_{i=1}^k$, such that the following hold:

1. There is a sparse cut between $S_i$ and $V \setminus P_i$ for any $1 \leq i \leq k$, which is formulated as $\Phi_G(P_i) \leq \Phi_{\text{out}}$;
2. Each induced subgraph $G[P_i]$ has high conductance $\Phi_{G[P_i]} \geq \Phi_{\text{in}}$.

We underline that, through the celebrated higher-order Cheeger inequality [LGT14], this condition of $(\Phi_{\text{in}}, \Phi_{\text{out}})$-decomposition can be approximately reduced to other formulations of a well-clustered graph studied in the literature, e.g., [PSZ17, vL07, ZLM13].

The rest of the section is structured as follows: we present the key notion and the strong decomposition lemma (Lemma 4.4) used in our main algorithm in Section 4.1. In Section 4.2, we present the algorithm for well-clustered graphs, whose performance is summarised in Theorem 4. We analyse the algorithm’s performance, and prove Theorem 4 in Section 4.3. Lastly, we prove the strong decomposition lemma in Section 4.4, due to the complexity of the underlying algorithm.

4.1 Partitioning well-clustered graphs and critical nodes

The starting point of our second result is the following polynomial-time algorithm presented by Gharan and Trevisan [GT14], which produces a $(\Phi_{\text{in}}, \Phi_{\text{out}})$-decomposition of a graph $G$, for some parameters $\Phi_{\text{in}}$ and $\Phi_{\text{out}}$. Specifically, given a well-clustered graph as input, their algorithm returns disjoint sets of vertices $\{P_i\}_{i=1}^k$ with bounded $\Phi(P_i)$ and $\Phi_{G[P_i]}$ for each $P_i$, and the algorithm’s performance is as follows:
Lemma 4.1 (Theorem 1.6, [GT14]). Let $G = (V, E, w)$ be a graph such that $\lambda_{k+1} > 0$, for some $k \geq 1$. Then, there is a polynomial-time algorithm that finds an $\ell$-partition $\{P_i\}_{i=1}^{\ell}$ of $V$, for some $\ell \leq k$, such that the following hold for every $1 \leq i \leq \ell$:

\begin{align*}
(1) & \quad \Phi(P_i) = O(k^6 \sqrt{\lambda_i}) ; \\
(2) & \quad \Phi_{G[P_i]} = \Omega(\lambda_{k+1}^2 / k^4).
\end{align*}

Informally, this result states that, when the underlying input graph $G$ presents a clear structure of clusters, one can find in polynomial-time a partition $\{P_i\}_{i=1}^{\ell}$ such that both the outer and inner conductance of every $P_i$ can be bounded. One natural question raising from this partition $\{P_i\}_{i=1}^{\ell}$ is whether we can directly use $\{P_i\}_{i=1}^{\ell}$ to construct an HC tree. As an obvious approach, one could consider to (i) construct trees $T_i = T_{\text{deg}}(G[P_i])$ for every $1 \leq i \leq \ell$, and (ii) merge the trees $\{T_i\}$ in the best way to construct the final tree $T_G$. Unfortunately, as we will see in the following example, this approach fails to achieve an $O(1)$-approximation.

Example 4.2. We study the following graph $G$, in which all the edges have unit weight:

1. Let $P_1 = (V_1, E_1)$ and $P_2 = (V_2, E_2)$ be the two copies of the graph constructed in Example 3.3. Specifically, every $P_i$ is a constant-degree expander graph of $n$ vertices with an additional planted clique. We use $S_i$ to represent the clique embedded in $P_i$, and $|S_i| = |n^{2/3}|$ for $i \in \{1, 2\}$;

2. We define $G \triangleq (V_1 \cup V_2, E_1 \cup E_2 \cup E_3)$, where $E_3$ consists of $\Theta(n^{1.1})$ crossing edges between $S_1$ and $S_2$, see Figure 5(a) for illustration.

By construction, we know that both of $P_1$ and $P_2$ have low outer conductance $\Phi(P_i) = O(n^{-0.23})$ and high inner conductance $\Phi_{G[P_i]} = \Omega(1)$, for $i \in \{1, 2\}$. Moreover, it is easy to see that the tree $T_G$ constructed by merging the trees $T_1$ and $T_2$, as shown in Figure 5(b) satisfies that $\text{cost}(T_G) = \Theta(n^{2.1})$. On the other hand, as illustrated in Figure 5(c), we can place the two cliques together and further down the HC tree, and obtain a tree $T^*$ with cost $\Theta(n^2)$. Thus we conclude that $\text{cost}(T_G) = \Omega(n^{0.1}) \cdot \text{OPT}_G$.

To address this, we follow our intuition gained from Example 3.3, and further decompose every $P_i$ into smaller subsets. Similar with analysing dense branches, we introduce the critical nodes associated with each $T_i$.

Definition 4.3 (Critical nodes). Let $T_i = T_{\text{deg}}(G[P_i])$ be the tree computed by Algorithm 1 to the induced graph $G[P_i]$. Suppose $(A_0, \ldots, A_r)$ is the dense branch of $T_i$, 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 $S_i \triangleq \{B_1, \ldots, B_{r_i+1}, A_{r_i+1}\}$ to be the set of critical nodes of $P_i$. Each node $N \in S_i$ is a critical node.
We remark that each critical node \( N \in S_i \) \((1 \leq i \leq \ell)\) is an internal node of maximum size in \( T_i \) that is not in the dense branch. Moreover, each \( S_i \) is a partition of \( P_i \). Based on critical nodes, we present an improved decomposition algorithm, which is similar to the one in Lemma 4.1, and prove that the output quality of our algorithm can be significantly strengthened for hierarchical clustering. Specifically, in addition to satisfying (A1) and (A2), we prove that the total weight between each critical node \( N \in S_i \) and all the other clusters \( P_j \) for all \( i \neq j \) can be upper bounded. We highlight that this is one of the key properties that allows us to obtain our main result, and also suggests that the original decomposition algorithm in [GT14] might not suffice for our purpose.

**Lemma 4.4 (Strong Decomposition Lemma).** Let \( G = (V, E, w) \) be a graph such that \( \lambda_{k+1} > 0 \) and \( \lambda_k < \left( \frac{1}{2c_0(k+1)^2} \right)^2 \), where \( c_0 \) is the constant in Lemma 2.5. Then, there is a polynomial-time algorithm that finds an \( \ell \)-partition of \( V \) into sets \( \{ P_i \}_{i=1}^{\ell} \), for some \( \ell \leq k \), such that for every \( 1 \leq i \leq \ell \) and every critical node \( N \in S_i \) the following properties hold:

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

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

(A3) \( w(N, V \setminus P_i) \leq 6(k+1) \cdot \text{vol}_{G[P_i]}(N) \).

To underline the importance of (A3), recall that, in general, each subtree \( T_i \) cannot be directly used to construct an \( O(1) \)-approximate HC tree of \( G \) because of the potential high cost of the crossing edges \( E(P_i, V \setminus P_i) \). However, if the internal cost of \( T_i \) is high enough to compensate for the cost introduced for the crossing edges \( E(P_i, V \setminus P_i) \), then one can safely use this \( T_i \) as a building block. This is one of the most crucial insights that leads us to design our final algorithm \textbf{PruneMerge}.

### 4.2 The algorithm for well-clustered graphs

Now we are ready to describe the algorithm \textbf{PruneMerge}, and we refer the reader to Algorithm 2 for the formal presentation. At a high-level, our algorithm consists of three phases: \textbf{Partition}, \textbf{Prune} and \textbf{Merge}. In the \textbf{Partition} phase (Lines 1–2), the algorithm invokes Lemma 4.4 to partition \( V(G) \) into sets \( \{ P_i \}_{i=1}^{\ell} \), and applies Algorithm 1 to obtain the corresponding trees \( \{ T_i \}_{i=1}^{\ell} \). The \textbf{Prune} phase (Lines 4–11) consists of a repeated pruning process: for every such tree \( T_i \), the algorithm checks in Line 7 if the maximal possible cost of the edges \( E(P_i, V \setminus P_i) \) (i.e., the LHS of the inequality in the \textbf{if}-condition) can be bounded by the internal cost of the critical nodes \( N \in S_i \), up to a factor of \( O(k) \).

- If so, the algorithm uses \( T_i \) as a building block and adds it to a global set of trees \( T \);
- Otherwise, the algorithm prunes the subtree \( T_i[N] \), where \( N \in S_i \) is the critical node closest to the root in \( T_i \), and adds \( T_i[N] \) to \( T \) (Line 11).

The process is repeated with the pruned \( T_i \) until either the condition in Line 7 is satisfied, or \( T_i \) is completely pruned. Finally, in the \textbf{Merge} phase (Lines 13–15) the algorithm combines the trees in \( T \) in a “caterpillar style” according to an increasing order of their sizes. The performance of this algorithm is summarised as follows:

**Theorem 4.** Let \( G = (V, E, w) \) be a graph, and \( k > 1 \) such that \( \lambda_{k+1} > 0 \) and \( \lambda_k < \left( \frac{1}{2c_0(k+1)^2} \right)^2 \), where \( c_0 \) is the constant in Lemma 2.5. The algorithm \textbf{PruneMerge} runs in polynomial-time and constructs an HC tree \( T_{PM} \) of \( G \) satisfying \( \text{cost}_G(T_{PM}) = O\left(k^{22}/\lambda_{k+1}^{10}\right) \cdot \text{OPT}_G \). In particular, when \( \lambda_{k+1} = \Omega(1) \) and \( k = O(1) \), the algorithm’s constructed tree \( T_{PM} \) satisfies that \( \text{cost}_G(T_{PM}) = O(1) \cdot \text{OPT}_G \).
In this subsection we will prove our main result, i.e., Theorem 4. We assume that the constructed trees by Algorithm 2 would always satisfy our promised approximation ratio.

\[ \text{Lemma 4.4:} \]

For every cluster \( P_i \) of the partitioning algorithm, we can apply the standard technique of running Algorithm 2 for different values of \( k \) and return the tree of lowest cost. By introducing a factor of \( O(k) \) to the algorithm’s runtime, this ensures that one of the constructed trees by Algorithm 2 would always satisfy our promised approximation ratio.

### 4.3 Analysis of the algorithm

In this subsection we will prove our main result, i.e., Theorem 4. We assume that the Partition phase of Algorithm 2 (Line 1) has finished and \( V(G) \) is decomposed into disjoint sets \( \{P_i\}_{i=1}^\ell \), for some \( \ell \leq k \), such that the following properties of Lemma 4.4 hold for all \( 1 \leq i \leq \ell \) and every critical node \( N \in S_i \):

\[
\begin{align*}
\Phi(P_i) & = O(k^6 \sqrt{\lambda_k}); \\
\Phi_G(P_i) & = \Omega(\lambda_{k+1}^2 / k^4); \\
w(N, V \setminus P_i) & \leq 6(k + 1) \cdot \text{vol}_{G[P_i]}(N).
\end{align*}
\]

Also, for some parameter \( \phi_{in} = \Theta(\lambda_{k+1}/k^2) \), we have that \( \Phi_G(P_i) = \Omega(\phi_{in}^2) \) holds for any \( 1 \leq i \leq \ell \). Let \( T_i = T_{\text{deg}}(G[P_i]) \) with corresponding set of critical nodes \( S_i \), for all \( 1 \leq i \leq \ell \), and let \( S = \bigcup_{i=1}^\ell S_i \) be the set of all critical nodes. In order to prove Theorem 4, we adopt the following strategy. We group the edges of \( G \) into two categories: let \( E_1 \) be the set of edges in the induced subgraphs \( G[P_i] \) for all \( 1 \leq i \leq \ell \), i.e.,

\[ E_1 \triangleq \bigcup_{i=1}^\ell E(G[P_i]), \]

and let \( E_2 \) be the remaining crossing edges. Therefore, we can write the cost of our tree \( T_{PM} \) as

\[ \text{cost}(T_{PM}) = \sum_{e \in E_1} \text{cost}_{T_{PM}}(e) + \sum_{e \in E_2} \text{cost}_{T_{PM}}(e), \quad (4.1) \]
and we will ultimately bound each sum individually in Lemmas 4.6 and 4.7. However, before presenting these two technical lemmas, we first introduce the notions of pruned and unpruned critical nodes, and analyse their properties.

Consider an arbitrary cluster $P_i$ with corresponding induced tree $T_i$. We say that a critical node $N \in S_i$ is pruned if $N$ was eventually cut from the tree $T_i$ and the subtree $T_i[N]$ was added to $T$ (Lines 10–11 of Algorithm 2). Otherwise, we say that $N$ is unpruned. We denote the set of all pruned nodes by PRUNED and the set of unpruned nodes by UNPRUNED.

Our first result bounds the size of the parent of a pruned node $N \in S_i$ in $T_{PM}$, with respect to the size of its parent in the tree $T_i$. This result will be extensively used when bounding the cost of the edges adjacent to $N$.

**Lemma 4.5.** It holds for every $1 \leq i \leq \ell$ and every pruned critical node $N \in S_i \cap \text{PRUNED}$ that

$$|\text{parent}_{T_{PM}}(N)| \leq 6k \cdot |\text{parent}_{T_i}(N)|.$$  

**Proof.** Suppose the dense branch of $T_i$ is $(A_0, \ldots, A_k)$ for some $k \in \mathbb{Z}_{\geq 0}$, with $B_j$ being the sibling of $A_j$ and $A_k$ having children $A_{k+1}, B_{k+1}$. Recall that the set of critical nodes is $S_i = \{B_1, \ldots, B_{k+1}, A_{k+1}\}$. By construction, it holds for all $1 \leq j \leq k$ 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 $j \geq 2$. Thus, we conclude that for every interval $(2^{s-1}, 2^s)$, for some $s \in \mathbb{Z}_{\geq 0}$, there are at most 3 critical nodes

$$N \in S_i$$

of size $|N| \in (2^{s-1}, 2^s]$. Now let us fix $N \in S_i \cap \text{PRUNED}$. By construction, we have that

$$|\text{parent}_{T_i}(N)| \geq 2 \cdot |N|.$$  

(4.2)

On the other hand, by the construction of $T_{PM}$ we have that

$$|\text{parent}_{T_{PM}}(N)| = \sum_{j=1}^{\ell} \sum_{M \in S_i, |M| \leq |N|} |M| \leq \sum_{s=0}^{\log |N|} \sum_{j=1}^{\ell} \sum_{M \in S_i, 2^{s-1} \leq |M| \leq 2^s} |M|$$

$$\leq \sum_{j=1}^{\ell} \sum_{s=0}^{\log |N|} 3 \cdot 2^s \leq 3 \cdot 2^{\log |N|} + 1 \leq 12k \cdot |N|.$$  

(4.3)

By combining (4.2) and (4.3), we have that

$$|\text{parent}_{T_{PM}}(N)| \leq 12k \cdot |N| \leq 6k \cdot |\text{parent}_{T_i}(N)|,$$

which proves the statement.

We are now ready to prove the two main technical lemmas of this subsection.

**Lemma 4.6.** It holds that

$$\sum_{e \in E_1} \text{cost}_{T_{PM}}(e) = O\left(\frac{k}{\phi_{in}}\right) \cdot \text{OPT}_G.$$  

**Proof.** Notice that

$$\sum_{e \in E_1} \text{cost}_{T_{PM}}(e) = \sum_{i=1}^{\ell} \sum_{e \in E[G[P_i]]} \text{cost}_{T_{PM}}(e).$$

We will prove that, for every $1 \leq i \leq \ell$ and $e \in E[G[P_i]]$, the cost of $e$ in $T_i$ differ by at most a factor of $O(k)$. Combining this with Theorem 3 will prove the lemma.

To prove this $O(k)$-factor bound, we fix any $1 \leq i \leq \ell$ and let $S_i$ be the set of critical nodes of $P_i$. As the nodes of $S_i$ form a partition of the vertices of $G[P_i]$, any edge $e \in E[G[P_i]]$ satisfies exactly one of the following conditions: (i) $e$ is inside a critical node; (ii) $e$ is adjacent to a pruned node; or (iii) $e$ crosses two unpruned nodes. Formally, it holds that

$^4$We remark that, in the worst case, all three nodes $B_1, B_{k+1}$ and $A_{k+1}$ could have size in $(2^{s-1}, 2^s]$. 

19
\[
\sum_{e \in E(G[P_i])} \text{cost}_{T_{PM}}(e) = \sum_{N \in S_i} \sum_{e \in E(N,N)} \text{cost}_{T_{PM}}(e) + \sum_{N \in S_i \cap \text{PRUNED}} \sum_{M \in S_i \setminus \{N\}} \sum_{e \in E(N,M)} \text{cost}_{T_{PM}}(e) + \sum_{N,M \in S_i \cap \text{UNPRUNED}} \sum_{N \neq M} \sum_{e \in E(N,M)} \text{cost}_{T_{PM}}(e)
\]

For Cases (i) and (iii), the costs of \(e\) in both trees are the same, since we do not change the structure of the tree inside any critical node nor alter the inner structure of the pruned trees \(T_i\) that contain only unpruned nodes, i.e.,

\[
\sum_{N \in S_i} \sum_{e \in E(N,N)} \text{cost}_{T_{PM}}(e) = \sum_{N \in S_i} \text{cost}_{T_i}(e),
\]

\[
\sum_{N,M \in S_i \setminus \text{UNPRUNED}} \sum_{e \in E(N,M)} \text{cost}_{T_{PM}}(e) = \sum_{N,M \in S_i \setminus \text{UNPRUNED}} \sum_{e \in E(N,M)} \text{cost}_{T_i}(e).
\]

For Case (ii), the cost of any such edge increases by at most a factor of \(O(k)\) due to Lemma 4.5 and the construction of \(T_{PM}\). Formally, let \(N \in S_i \cap \text{PRUNED}\) be an arbitrary pruned node and let \(M \in S_i \setminus \{N\}\) be a critical node such that \(|\text{parent}_{T_i}(M)| \leq |\text{parent}_{T_i}(N)|\). Firstly, notice that if parent_{\(T_i\)}(\(N\)) is the root node of \(T_i\), then for any edge \(e \in E(N, M)\) it holds that \(\text{cost}_{T_i}(e) = w_e \cdot |T_i|\). On the other side, by the construction of \(T_{PM}\), we know that \(\text{cost}_{T_{PM}}(e) \leq 6k \cdot w_e \cdot |T_i|\), so we conclude that \(\text{cost}_{T_{PM}}(e) \leq 6k \cdot \text{cost}_{T_i}(e)\). Secondly, if parent_{\(T_i\)}(\(N\)) is not the root node of \(T_i\) and since \(|\text{parent}_{T_i}(M)| \leq |\text{parent}_{T_i}(N)|\), we know that \(|M| \leq |N|\). Therefore it holds for any edge \(e \in E(N, M)\) that
\[
\text{cost}_{T_{PM}}(e) = w_e \cdot |\text{parent}_{T_{PM}}(N)| \leq 6k \cdot w_e \cdot |\text{parent}_{T_i}(N)| = 6k \cdot \text{cost}_{T_i}(e),
\]

where the inequality follows by Lemma 4.5. Combining the above observations, we have that

\[
\sum_{e \in E_1} \text{cost}_{T_{PM}}(e) \leq \sum_{i=1}^{\ell} 6k \cdot \sum_{e \in E(G[P_i])} \text{cost}_{T_i}(e) \leq \sum_{i=1}^{\ell} 6k \cdot \text{cost}(T_i). \tag{4.4}
\]

On the other side, let \(T^*\) be any optimal HC tree of \(G\) with cost \(\text{OPT}_G\), and it holds that
\[
\text{OPT}_G = \text{cost}_G(T^*) \geq \sum_{i=1}^{\ell} \sum_{e \in E(G[P_i])} \text{cost}_{T_i}(e) \geq \sum_{i=1}^{\ell} \text{OPT}_{G[P_i]} = \sum_{i=1}^{\ell} \Omega \left(\Phi_{G[P_i]}^4\right) \cdot \text{cost}(T_i) = \sum_{i=1}^{\ell} \Omega \left(\phi_{\text{in}}^8\right) \cdot \text{cost}(T_i), \tag{4.5}
\]

where the last equality follows by Property (A2) of Lemma 4.4 and Theorem 3 applied to every \(G[P_i]\). Finally, by combining (4.4) and (4.5) we have that
\[
\sum_{e \in E_1} \text{cost}_{T_{PM}}(e) = O(k/\phi_{\text{in}}^8) \cdot \text{OPT}_G,
\]

which proves the lemma.

\[\square\]

\textbf{Lemma 4.7.} It holds that \(\sum_{e \in E_2} \text{cost}_{T_{PM}}(e) = O \left(k^2/\phi_{\text{in}}^{10}\right) \cdot \text{OPT}_G\).
Proof. For the edges $e \in E_2$, we can subdivide them into (i) edges adjacent to pruned nodes and (ii) edges adjacent to unpruned nodes. For Case (i), we will bound the cost with the help of Lemma 4.5 similar as before. For Case (ii) we will upper bound the cost based on the fact that, for unpruned nodes, the condition in Line 7 of Algorithm 2 is satisfied. Specifically, we have that

$$\sum_{e \in E_2} \text{cost}_{PM}(e) \leq \sum_{i=1}^{\ell} \sum_{N \in \mathcal{S}_i, |M| \leq |N|} \sum_{e \in E(N,M)} \text{cost}_{PM}(e)$$

$$\leq \sum_{i=1}^{\ell} \sum_{N \in \mathcal{S}_i \cap \text{UNPRUNED}} \sum_{M \in \mathcal{S}_i} \text{cost}_{PM}(e) + \sum_{i=1}^{\ell} \sum_{N \in \mathcal{S}_i \cap \text{PRUNED}} \sum_{M \in \mathcal{S}_i \setminus \mathcal{S}_j} \text{cost}_{PM}(e)$$

$$\leq \sum_{i=1}^{\ell} \sum_{N \in \mathcal{S}_i \cap \text{UNPRUNED}} n \cdot w(N, V \setminus P_i) + \sum_{i=1}^{\ell} \sum_{N \in \mathcal{S}_i \cap \text{PRUNED}} |\text{parent}_{PM}(N)| \cdot w(N, V \setminus P_i)$$

$$\leq \sum_{i=1}^{\ell} \sum_{N \in \mathcal{S}_i \cap \text{UNPRUNED}} 6(k+1) \cdot |\text{parent}_{T_i}(N)| \cdot \text{vol}_{G[P_i]}(N) \quad (4.6)$$

$$+ \sum_{i=1}^{\ell} \sum_{N \in \mathcal{S}_i \cap \text{PRUNED}} 36k(k+1) \cdot |\text{parent}_{T_i}(N)| \cdot \text{vol}_{G[P_i]}(N) \quad (4.7)$$

$$\leq 36k(k+1) \cdot \sum_{i=1}^{\ell} \sum_{N \in \mathcal{S}_i} |\text{parent}_{T_i}(N)| \cdot \text{vol}_{G[P_i]}(N)$$

$$\leq 36k(k+1) \cdot \sum_{i=1}^{\ell} \frac{\Phi_{G[P_i]}}{4} \cdot \text{cost}_{G[P_i]}(T_i) \quad (4.8)$$

$$= O(k^2) \cdot \sum_{i=1}^{\ell} \frac{1}{\Phi_{G[P_i]}} \cdot \text{OPT}_{G[P_i]} \quad (4.9)$$

where (4.6) follows by fact that the unpruned nodes satisfy the if-condition in Line 7 of Algorithm 2, (4.7) follows from Property (A3) of Lemma 4.4 and Lemma 4.5, (4.8) follows by Lemma 3.5, and (4.9) follows by Theorem 3 applied to every induced subgraph $G[P_i]$. \hfill \square

Finally, we are ready to prove Theorem 4.

Proof of Theorem 4. Let $T_{PM}$ be the HC tree obtained from Algorithm 2. We have that

$$\text{cost}(T_{PM}) = \sum_{e \in E_1} \text{cost}_{PM}(e) + \sum_{e \in E_2} \text{cost}_{PM}(e)$$

$$= O(k^2) \cdot \Phi_{in}^{10} \cdot \text{OPT}_{G} + O(k^2) \cdot \Phi_{in}^{10} \cdot \text{OPT}_{G} = O(k^2) \cdot \Phi_{in}^{10} \cdot \text{OPT}_{G} = O(k^2) \cdot \Phi_{in}^{10} \cdot \text{OPT}_{G}$$

where the second equality follows by Lemmas 4.6 and 4.7, and the last equality follows by the definition of $\Phi_{in}$.

Next we analyse the runtime of our algorithm. The Partition phase (Lines 1–2) runs in polynomial time by Lemma 4.4. In the Prune phase (Lines 4–11), the algorithm goes through all $\ell \leq k$ trees $T_i$, and for each tree the algorithm attempts to prune the critical node closest to the root. If the pruning happens, the process is repeated recursively to the pruned tree. Since
there are \( O(\log n) \) critical nodes for each tree \( T_i \) by construction, the \textbf{if}-condition in Line 7 of Algorithm 2 will be checked \( O(\log n) \) number of times for each tree \( T_i \). Hence we conclude that the \textbf{Prune} phase of the algorithm runs in polynomial time. Finally, in the \textbf{Merge} phase (Lines 13–15) the algorithm goes through the global set of trees \( T \), and successively merges the trees \( T_i \in T \) to form the final tree \( T_{PM} \). Since \( |T| = O(\ell \cdot \log n) \), we conclude that this step will be finished in polynomial time.

\[ \square \]

4.4 Proof of Lemma 4.4

Finally, we finish the section by presenting the proof of Lemma 4.4. We first describe the underlying algorithm and show a sequence of claims, which are used to prove Lemma 4.4 in the end of the subsection. At the very high level, our algorithm for computing a stronger decomposition of a well-clustered graph can be viewed as an extension to Algorithm 3 in [GT14], whose main idea can be summarised as follows: the algorithm starts with the trivial 1-partition of \( G \), i.e., \( P_1 = V \); in every iteration, the algorithm applies the \textbf{Spectral Partitioning} algorithm for every graph in \( \{G[P_i]\}_{i=1}^r \), and tries to find a sparse cut \((S, P_i \setminus S)\) for some \( S \subseteq P_i \).

\[ ^5 \text{We will denote by } r \text{ the number of clusters in the current run of the algorithm, and denote by } \ell \text{ the final number of clusters output by the algorithm.} \]

- If such a cut is found, the algorithm uses this cut to either introduce a new partition set \( P_{r+1} \) of small conductance, or refine the current partition \( \{P_i\}_{i=1}^r \);
- If no such cut is found, the algorithm checks if it is possible to perform a local refinement of the partition sets \( \{P_i\}_{i=1}^r \) in order to reduce the overall weight of the crossing edges, i.e. \( \sum_{i \neq j} w(P_i, P_j) \). 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 4.4.

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 [GT14]. We write \( \{P_i\}_{i=1}^r \) as a partition of \( V \) for some integer \( r \geq 1 \), i.e., \( P_i \cap P_j = \emptyset \) for \( i \neq j \), and \( \cup_{i=1}^r P_i = V \). In addition, every partition set \( P_i \) contains some core set denoted by \( \text{core}(P_i) \subseteq P_i \). For an arbitrary subset \( S \subseteq P_i \), we define \( S^+ \triangleq S \cap \text{core}(P_i) \), and \( S^- \triangleq S \setminus S^+ \). We further define \( \overline{S^+} \triangleq \text{core}(P_i) \setminus S \), and \( \overline{S^-} \triangleq P_i \setminus (S \cup \text{core}(P_i)) \), as illustrated in Figure 6. Note that \( \{S^+, \overline{S^+}\} \) forms a partition of \( \text{core}(P_i) \), and \( \{S^-, \overline{S^-}\} \) forms a partition of \( P_i \setminus \text{core}(P_i) \). 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 \( S \subseteq P \subseteq V \), we follow [GT14] and define the relative conductance as

\[ \varphi(S, P) \triangleq \frac{w(S \to P)}{\frac{\text{vol}(P \setminus S)}{\text{vol}(P)} \cdot w(S \to V \setminus P)}, \]

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

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

Combining the above observations, one could conclude that $w(S \to P) \gtrsim w(S \to V \setminus P)$ if $P$ is a good cluster, which means that $\varphi(S, P)$ is lower bounded by a constant. Moreover, Gharan and Trevisan [GT14] showed a converse of this fact: if $\varphi(S, P)$ is large for all $S \subset P$, then $P$ has high inner conductance. These facts suggest that the relative conductance provides a good quantitative measure for 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 [GT14]), and obtains an intermediate partition $\{P_i\}_{i=1}^r$ (Lines 4–17). For every $P_i$ ($1 \leq i \leq r$), the algorithm computes the induced trees $T_i = T_{\text{deg}}(G[P_i])$ (Line 18). For every tree $T_i$ ($1 \leq i \leq r$) with the corresponding set of critical nodes $S_i$, the algorithm further checks if the following conditions are satisfied:

1. $A_4$ For every critical node $N \in S_i$ with $\text{vol}(N) \leq \text{vol}(\text{core}(P_i)) / 2$, it holds that
   \[ \varphi(N, \text{core}(P_i)) \geq \frac{1}{3(k+1)}; \]
2. $A_5$ For every critical node $N \in S_i$ with $\text{vol}(N) \leq \text{vol}(P_i) / 2$, it holds that
   \[ w(N \to P_i) \geq w(N \to V \setminus P_i) \cdot \frac{1}{k+1}. \]

If $(A_4)$ is violated by some critical node $N \in S_i$ for some $i$, then the algorithm uses $N$ to refine the core set $\text{core}(P_i)$ (Line 22). If $(A_5)$ is not satisfied, then the algorithm further refines the partition (Line 24). 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 2.5, and

\[ \phi_{\text{in}} \triangleq \frac{\lambda_{k+1}}{140(k+1)^2}, \quad \phi_{\text{out}} \triangleq 90c_0 \cdot (k+1)^6 \sqrt{\lambda_k}. \]

(4.10)
Notice that, by assuming $\lambda_k < \left(\frac{1}{2r_0-c_0(k+1)^3}\right)^2$ in Lemma 4.4, it holds that $\phi_{\text{out}} < 1/3$. This fact will be used several times in our analysis.

**Algorithm 3:** Algorithm for partitioning $G$ into $\ell \leq k$ clusters

**Input:** $G = (V, E, w)$, $k > 1$ such that $\lambda_{k+1} > 0$;

**Output:** A $\left(\phi_{\text{in}}^2/4, \phi_{\text{out}}\right)$-partition $\{P_i\}_{i=1}^\ell$ of $G$ satisfying (A1) – (A3), for some $\ell \leq k$;

1. Let $r = 1$, $\text{core}(P_1) = P_1 = V$;
2. Let $\phi_{\text{in}} = \frac{\lambda_{k+1}}{140(k+1)^3}$, and $\phi_{\text{out}} = 90c_0 \cdot (k+1)^6 \sqrt{\lambda_k}$;
3. Let $\rho^* = \min \left\{\frac{\lambda_{k+1}}{10}, 30c_0 \cdot (k+1)^5 \cdot \sqrt{\lambda_k}\right\}$;
4. while At least one of the following conditions holds
   1. $\exists 1 \leq i \leq r$ such that $w(P_i \setminus \text{core}(P_i)) \rightarrow P_i < w(P_i \setminus \text{core}(P_i)) \rightarrow P_j$ for some $j \neq i$;
   2. **Spectral Partitioning** finds $S \subseteq P_i$ with $6 \cdot \text{vol}(S^+ \leq \text{vol}(\text{core}(P_i))/2$, such that max $\{\Phi_{G[P_i]}(S), \Phi_{G[P_i]}(P_i \setminus S)\} < \phi_{\text{in}}$;
5. do
6. Order the sets $P_1, \ldots, P_r$ such that $\lambda_2(G[P_1]) \leq \ldots \leq \lambda_2(G[P_r])$;
7. Let $1 \leq i \leq r$ the smallest index for which item 2 of the while-condition is satisfied, and let $S \subset P_i$ be the corresponding set;
8. if $\max \{\Phi_G(S^+), \Phi_G(\overline{S^+})\} \leq \left(1 + \frac{1}{k+1}\right)^{r+1} \cdot \rho^*$ then
9. Let $P_i = P_i \setminus \overline{S^+}$, $\text{core}(P_i) = S^+$, $P_{r+1} = \text{core}(P_{r+1}) = \overline{S^+}$, $r = r + 1$ and go to Line 4;
10. if $\min \{\varphi(S^+, \text{core}(P_i)), \varphi(\overline{S^+}, \text{core}(P_i))\} \leq \frac{1}{3(k+1)}$ then
11. Update $\text{core}(P_i)$ to either $S^+$ or $\overline{S^+}$ with the lower conductance, and go to Line 4;
12. if $\Phi_G(S^-) \leq \left(1 + \frac{1}{k+1}\right)^{r+1} \cdot \rho^*$ then
13. Let $P_i = P_i \setminus S^-$, $P_{r+1} = \text{core}(P_{r+1}) = S^-$, $r = r + 1$ and go to Line 4;
14. if $w(P_i \setminus \text{core}(P_i)) \rightarrow P_j < w(P_i \setminus \text{core}(P_i)) \rightarrow P_j$ for some $j \neq i$ then
15. Let $P_i = \text{core}(P_i)$, merge $(P_i \setminus \text{core}(P_i))$ with $\text{argmax}_{P_j} w(P_i \setminus \text{core}(P_i) \rightarrow P_j)$, and go to Line 4;
16. if $w(S^- \rightarrow P_j) < w(S^- \rightarrow P_j)$ for some $j \neq i$ then
17. Let $P_i = P_i \setminus S^-$, merge $S^-$ with $\text{argmax}_{P_j} w(S^- \rightarrow P_j)$, and go to Line 4;
18. For every partition set $P_i$ with HC tree $T_i$, let $S_i$ be the set of critical nodes;
19. if $\exists N \in S_i$ such that $\max \{\Phi_G(N^+), \Phi_G(\overline{N^+})\} \leq (1 + \frac{1}{k+1})^{r+1} \cdot \rho^*$ then
20. Let $P_i = P_i \setminus \overline{N^+}$, $\text{core}(P_i) = N^+$, $P_{r+1} = \text{core}(P_{r+1}) = \overline{N^+}$, $r = r + 1$ and go to Line 4;
21. if $\exists N \in S_i$ such that $\text{vol}(\text{vol}(N^+) \leq \text{vol}(\text{core}(P_i))/2$ and $\varphi(N^+, \text{core}(P_i)) \leq \frac{1}{3(k+1)}$ then
22. Update $\text{core}(P_i)$ to either $N^+$ or $\overline{N^+}$ with the lower conductance, and go to Line 4;
23. if $\exists N \in S_i$ such that $\text{vol}(N^-) \leq \text{vol}(P_i)/2$ and $w(N^- \rightarrow P_j) < w(N^- \rightarrow P_j)$ for some $j \neq i$ then
24. Let $P_i = P_i \setminus N^-$, merge $N^-$ with $\text{argmax}_{P_j} w(N^- \rightarrow P_j)$, and go to Line 4;
25. return the partition $\{P_i\}_{i=1}^r$.

Following the proof structure in [GT14], we will prove Lemma 4.4 via a sequence of claims. Notice that, during the entire execution of the algorithm, the sets $\{P_i\}_{i=1}^r$ always form a

---

6If the set $S \subset P_i$ returned by **Spectral Partitioning** has $\text{vol}(S^+) > \text{vol}(G)/2$, swap $S$ with $P_i \setminus S$.  

---

24
partition of $V$, and each core($P_i$) is a subset of $P_i$. Firstly, we show that, at any point during the execution of the algorithm, the core sets core($P_i$) ($1 \leq i \leq r$) always have low conductance.

Claim 4.8. Throughout the algorithm, we always have that

$$\max_{1 \leq i \leq r} \Phi_G(\text{core}(P_i)) \leq \rho^* \cdot \left(1 + \frac{1}{k+1}\right)^r.$$ 

The following result will be used in our proof:

Lemma 4.9 (Lemma 2.2, [GT14]). 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. $\varphi(S,W) \leq \varepsilon/3$ and
2. $\max\{\Phi_G(S), \Phi_G(W \setminus S)\} \geq (1 + \varepsilon) \cdot \Phi_G(W)$.

Then it holds that

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

Proof of Claim 4.8. 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 core($P_1$) = $P_1 = V$, which gives us that $\Phi_G(\text{core}(P_1)) = 0$; hence, the statement holds trivially. Secondly, for the inductive step, we assume that the statement holds for some fixed configuration of the core sets $\{\text{core}(P_i)\}_{i=1}^r$ and we prove that the statement holds after the algorithm updates the current configuration. Notice that $\{\text{core}(P_i)\}_{i=1}^r$ are updated through Lines 9, 11, 13, 20 and 22 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 9, 13 and 20, the algorithm introduces some new set core($P_{r+1}$) such that

$$\Phi_G(\text{core}(P_{r+1})) \leq \rho^* \cdot \left(1 + \frac{1}{k+1}\right)^{r+1}.$$ 

Combining this with the inductive hypothesis, which assumes the inequality holds for core($P_i$) ($1 \leq i \leq r$), we have that

$$\max_{1 \leq i \leq r+1} \Phi_G(\text{core}(P_i)) \leq \rho^* \cdot \left(1 + \frac{1}{k+1}\right)^{r+1}.$$ 

- The case for executing Lines 11 and 22 is similar, so we focus on Line 11 here and prove this by applying Lemma 4.9. When executing Line 11, we know that the if-condition in Line 8 does not hold, so we have that

$$\max\{\Phi_G(S^+), \Phi_G(\overline{S^+})\} > \left(1 + \frac{1}{k+1}\right)^{r+1} \cdot \rho^* \geq \left(1 + \frac{1}{k+1}\right) \cdot \Phi_G(\text{core}(P_i)),$$

where the last inequality follows by the inductive hypothesis. Moreover, when executing Line 11, we also know that the if-condition in Line 10 holds, i.e.,

$$\min\{\varphi(S^+, \text{core}(P_i)), \varphi(\overline{S^+}, \text{core}(P_i))\} \leq \frac{1}{3(k+1)}.$$ 

Therefore, by applying Lemma 4.9 with $S^+ \subset \text{core}(P_i)$ and $\varepsilon = 1/(k+1)$ and using the inductive hypothesis, we conclude that

$$\min\{\Phi_G(S^+), \Phi_G(\overline{S^+})\} \leq \Phi_G(\text{core}(P_i)) \leq \rho^* \cdot \left(1 + \frac{1}{k+1}\right)^r.$$
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 will show that the number of partition sets cannot exceed \( k \). This proof is identical to Claim 3.2 in [GT14], and we include the proof here for completeness.

\textbf{Claim 4.10.} The total number of clusters returned by the algorithm satisfies that \( \ell \leq k \).

\textit{Proof.} Suppose for contradiction that the number of clusters becomes \( r = k + 1 \) at some point during the execution of the algorithm. Then, since \( \text{core}(P_1), \ldots, \text{core}(P_{k+1}) \) are disjoint, by the definition of \( \rho(k + 1) \) and Claim 4.8 we have that

\[
\rho(k + 1) \leq \max_{1 \leq i \leq k+1} \Phi_G(\text{core}(P_i)) \leq \left(1 + \frac{1}{k+1}\right)^{k+1} \cdot \rho^* \leq e \cdot \rho^* \leq e \cdot \frac{\lambda_{k+1}}{10} < \frac{\lambda_{k+1}}{2},
\]

which contradicts Lemma 2.4. Therefore, the total number of clusters at any time satisfies \( r < k + 1 \), which means that the final number of clusters satisfies \( \ell \leq k \). This proves the claim.

Now we are ready to show that the output \( \{P_i\}_{i=1}^{\ell} \) of Algorithm 3 and its core sets \( \{\text{core}(P_i)\}_{i=1}^{\ell} \) satisfy Properties (A4) and (A5), which will be used in proving Lemma 4.4.

\textbf{Claim 4.11.} Let \( \{P_i\}_{i=1}^{\ell} \) be the output of Algorithm 3 with the corresponding core sets \( \{\text{core}(P_i)\}_{i=1}^{\ell} \). Then, the following hold for any \( 1 \leq i \leq \ell \):

1. \( \Phi_G(\text{core}(P_i)) \leq \phi_{\text{out}}/(k + 1) \);
2. \( \Phi_G(P_i) \leq \phi_{\text{out}}; \)
3. \( \Phi_G(P_i) \geq \phi_{\text{in}}^2/4 \).

Moreover, assuming that \( S_i \) is the set of critical nodes of \( T_i = \text{deg}(G[P_i]) \), the following two properties hold for any \( 1 \leq i \leq \ell \):

4. For every critical node \( N \in S_i \) with \( \text{vol}(N^+) \leq \text{vol}(\text{core}(P_i))/2 \), we have that
   \[
   \varphi(N^+, \text{core}(P_i)) \geq \frac{1}{3(k + 1)};
   \]
5. For every critical node \( N \in S_i \) with \( \text{vol}(N^-) \leq \text{vol}(P_i)/2 \), we have that
   \[
   w(N^- \rightarrow P_i) \geq w(N^- \rightarrow V \setminus P_i) \cdot \frac{1}{k + 1}.
   \]

\textit{Proof.} First of all, by Claim 4.8 we have for any \( 1 \leq i \leq \ell \) that

\[
\Phi_G(\text{core}(P_i)) \leq \rho^* \cdot \left(1 + \frac{1}{k+1}\right)^{\ell} \leq e \cdot \rho^* \leq 30 \cdot e \cdot c_0 \cdot (k + 1)^5 \sqrt{\lambda_k} \leq \frac{\phi_{\text{out}}}{k + 1},
\]

where the second inequality holds by the fact that \( \ell \leq k + 1 \), the third one holds by the choice of \( \rho^* \), and the last one holds by the choice of \( \phi_{\text{out}} \). This proves Item (1).

To prove Item (2), we notice that the first condition of the while-loop (Line 4) doesn’t hold when the algorithm terminates, hence we have for any \( 1 \leq i \neq j \leq \ell \) that

\[
w(P_i \setminus \text{core}(P_i) \rightarrow P_i) \geq w(P_i \setminus \text{core}(P_i) \rightarrow P_j).
\]

By applying the averaging argument, we have that

\[
w(P_i \setminus \text{core}(P_i) \rightarrow \text{core}(P_i)) = w(P_i \setminus \text{core}(P_i) \rightarrow P_i)
\]
Therefore, by the same averaging argument we have that

\[
\frac{\log \left( \frac{w(P_i \setminus \text{core}(P_i) \to V)}{\ell} \right)}{\log \left( \frac{w(P_i \setminus \text{core}(P_i) \to V \setminus P_i)}{k} \right)} \geq \frac{w(P_i \setminus \text{core}(P_i) \to V)}{\ell} \geq \frac{w(P_i \setminus \text{core}(P_i) \to V \setminus P_i)}{k}.
\] (4.11)

We apply the same analysis used in [GT14], and have that

\[
\Phi_G(P_i) = \frac{w(P_i \to V)}{\text{vol}(P_i)} \leq \frac{w(\text{core}(P_i) \to V) + w((P_i \setminus \text{core}(P_i) \to V \setminus P_i) - w(P_i \setminus \text{core}(P_i) \to \text{core}(P_i)))}{\text{vol}(\text{core}(P_i))} \\
\leq \Phi_G(\text{core}(P_i)) + \frac{(k-1) \cdot w(P_i \setminus \text{core}(P_i) \to \text{core}(P_i)))}{\text{vol}(\text{core}(P_i))} \\
\leq k \cdot \Phi_G(\text{core}(P_i)) \\
\leq \phi_{\text{out}},
\]

where the second inequality uses equation (4.11). This proves Item (2).

Next, we analyse Item (3). Again, we know that the second condition within the while-loop (Line 4) does not hold when the algorithm terminates. By the performance of the Spectral Partitioning algorithm (i.e., Lemma 2.3), it holds for any 1 \(\leq i \leq \ell \) that \(\Phi_{G(P_i)} \geq \phi_{\text{in}}^2/4 \). With this, we prove that Item (3) holds.

Similarly, when the algorithm terminates, we know that for any critical node \(N\) the if-condition in Line 21 does not hold. Hence, for any 1 \(\leq i \leq \ell \) and any \(N \in S_i\) with \(\text{vol}(N^+) \leq \text{vol}(\text{core}(P_i))/2\), we have that

\[
\varphi(N^+, \text{core}(P_i)) \geq \frac{1}{3(k+1)}.
\]

This shows that Item (4) holds as well.

Finally, since there is no \(N\) satisfying the if-condition in Line 23 of the algorithm, it holds for any 1 \(\leq i \neq j \leq \ell \) and every critical node \(N \in S_i\) that \(w(N^- \to P_i) \geq w(N^- \to P_j)\). Therefore, by the same averaging argument we have that

\[
w(N^- \to P_i) \geq \frac{w(N^- \to V)}{\ell} \geq \frac{w(N^- \to V \setminus P_i)}{k+1},
\]

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 5–17), at least one of the if-conditions will be satisfied, and some sets are updated accordingly. This fact, stated as Claim 4.13, is important, since otherwise the algorithm might end up in an infinite loop. The following result will be used in our proof.

**Lemma 4.12** (Lemma 2.6, [GT14]). Let \(\text{core}(P_i) \subseteq P_i \subseteq V\), and \(S \subseteq P_i\) be such that \(\text{vol}(S^+) \leq \text{vol}(\text{core}(P_i))/2\). Suppose that the following hold for some parameters \(\rho\) and 0 < \(\varepsilon < 1\):

1. \(\rho \leq \Phi_G(S^-)\) and \(\rho \leq \max\{\Phi_G(S^+), \Phi_G(S^+)^{\text{out}}\}\);
2. If \(S^- \neq \emptyset\), then \(w(S^- \to P_i) \geq w(S^- \to V)/k\);
3. If \(S^+ \neq \emptyset\), then \(\varphi(S^+, \text{core}(P_i)) \geq \varepsilon/3\) and \(\varphi(S^+, \text{core}(P_i)) \geq \varepsilon/3\).

Then, it holds that

\[
\Phi_{G(P_i)}(S) \geq \varepsilon \cdot \frac{\rho}{14k}.
\]
Claim 4.13. If at least one condition of the while-loop is satisfied, then at least one of the if-conditions (Lines 8, 10, 12, 14 or 16) is satisfied.

Proof. First of all, notice that if the first condition of the while-loop is satisfied, then the if-condition in Line 14 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 $\Phi_{G[P_i]}(S) \geq \phi_{in}$. The proof is structured in the following two steps:

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

**Step 1:** We prove this fact by applying Lemma 4.12 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 4.12 are satisfied, beginning with the first one. If $S^- = \emptyset$, then we trivially have that $1 = \Phi_G(S^-) \geq \rho$; so we assume that $S^- \neq \emptyset$. As the if-condition in Line 12 is not satisfied, we have that

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

and combining this with Claim 4.8 gives us that

$$
\Phi_G(S^-) = \max\{\Phi_G(\text{core}(P_1)), \ldots, \Phi_G(\text{core}(P_r)), \Phi_G(S^-)\} \geq \rho(r+1).
$$

Therefore, we have that

$$
\Phi_G(S^-) \geq \max\{\rho^*, \rho(r+1)\} = \rho. \quad (4.12)
$$

Similarly, if $S^+ = \emptyset$, then we trivially have that $1 = \Phi_G(S^+) \geq \rho$; so we assume that $S^+ \neq \emptyset$. Moreover, since we have chosen $S^+$ such that $\text{vol}(S^+) \leq \text{vol}(\text{core}(P_i))/2$, we know that $S^+ = \text{core}(P_i) \setminus S^+ \neq \emptyset$. As the if-condition in Line 8 is not satisfied, we have that

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

Combining this with Claim 4.8 gives us that

$$
\max\{\Phi_G(S^+), \Phi_G(S^+)\}
= \max\{\Phi_G(\text{core}(P_1)), \ldots, \Phi_G(\text{core}(P_{i-1})), \Phi_G(S^+), \Phi_G(\overline{S^+}), \Phi_G(\text{core}(P_{i+1})), \ldots, \Phi_G(\text{core}(P_r))\}
\geq \rho(r+1).
$$

Therefore we have that

$$
\max\{\Phi_G(S^+), \Phi_G(S^+)\} \geq \max\{\rho^*, \rho(r+1)\} = \rho. \quad (4.13)
$$

Combining (4.12) and (4.13), we see that the first condition of Lemma 4.12 is satisfied. Since the if-condition in Line 16 is not satisfied, it follows by an averaging argument that

$$
w(S^- \rightarrow P_i) \geq \frac{w(S^- \rightarrow V)}{k},
$$

28
which shows that the second condition of Lemma 4.12 is satisfied. Finally, since the if-condition in Line 10 is not satisfied, we know that
\[
\min \left\{ \varphi(S^+, \text{core}(P_i)), \varphi(S^+, \text{core}(P_i)) \right\} \geq \frac{1}{3(k + 1)},
\]
which shows that the third condition of Lemma 4.12 is satisfied as well. Hence, by Lemma 4.12 we conclude that
\[
\Phi_{G[P_i]}(S) \geq \varepsilon \cdot \frac{1}{4(k + 1)} = \frac{\max\{\rho^*, \rho(r + 1)\}}{4(k + 1)^2},
\]
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 (4.14) and Lemma 2.4, we have that
\[
\Phi_{G[P_i]}(S) \geq \frac{\rho^*}{4(k + 1)^2} = \frac{\rho^*}{14(k + 1)^2} \geq \frac{\lambda_{k+1}}{28(k + 1)^2} = \phi_{in},
\]
which leads to the desired contradiction.

**Case 2:** \( r < k \). Recall that the partition sets \( \{P_i\}_{i=1}^r \) are labelled such that \( \lambda_2(G[P_1]) \leq \ldots \leq \lambda_2(G[P_r]) \), and the algorithm has chosen the lowest index \( i \) for which the set \( S \subseteq P_i \) 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 \( S \subseteq P_1 \)). We combine the performance of the **Spectral Partitioning** algorithm (Lemma 2.3) with Lemma 2.5, and obtain that
\[
\Phi_{G[P_1]}(S) \leq \sqrt{2 \lambda_2(G[P_1])} = \min_{1 \leq j \leq r} \sqrt{2 \lambda_2(G[P_j])} \leq \sqrt{4c_0 \cdot k^6 \cdot \lambda_k}. \tag{4.15}
\]
Combining (4.14) and (4.15) we have that
\[
\rho^* \leq 28c_0 \cdot (k + 1)^5 \sqrt{\lambda_k}.
\]
Thus, by the definition of \( \rho^* \) we have that
\[
\rho^* = \frac{\lambda_{k+1}}{10}.
\]
We combine this with (4.14), and have that
\[
\Phi_{G[P_i]}(S) \geq \frac{\lambda_{k+1}}{140(k + 1)^2} = \phi_{in},
\]
which gives our desired contradiction.

**Case 2b:** \( i > 1 \) (i.e., the algorithm selects \( S \subseteq P_i \) for some \( i \geq 2 \). Let \( S_1 \subseteq P_1 \) be the set obtained by applying the **Spectral Partitioning** algorithm to the graph \( G[P_1] \). Since the algorithm did not select \( S_1 \subseteq P_1 \), we know that \( \Phi_{G[P_i]}(S_1) \geq \phi_{in} \). Combining the performance of the **Spectral Partitioning** algorithm (Lemma 2.3) with Lemma 2.5, we have that
\[
\phi_{in} \leq \Phi_{G[P_1]}(S_1) \leq \min_{1 \leq j \leq r} \sqrt{2 \lambda_2(G[P_j])} \leq 2c_0 \cdot k^3 \cdot \sqrt{\lambda_k}.
\]
This gives us that
\[
\frac{\lambda_{k+1}}{10} = 14(k + 1)^2 \cdot \phi_{in} < 30c_0 \cdot (k + 1)^5 \cdot \sqrt{\lambda_k},
\]

29
and it holds by the definition of $\rho^*$ that

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

Therefore, by (4.14) we have that

$$
\Phi_G[P_{i+1}](S) \geq \frac{\lambda_{k+1}}{140(k+1)^2} = \phi_{in}.
$$

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 4" is executed, is finite.

**Claim 4.14.** For any graph $G = (V, E, w)$ with the minimum weight $w_{\min}$ as the input, Algorithm 3 terminates after executing the while-loop $O(k \cdot n \cdot \text{vol}(G)/w_{\min})$ times.

**Proof.** Notice that the algorithm goes back to check the loop conditions (Line 4) right after any of Lines 9, 11, 13, 15, 17, 20, 22 and 24 is executed, and each of these commands changes the current structure of our partition $\{P_i\}_{i=1}^T$ with core sets $\text{core}(P_i) \subseteq P_i$. We classify these updates into the following three types:

1. The updates that introduce a new partition set $P_{r+1}$. These correspond to Lines 9, 13 and 20;
2. The updates that contract the core sets $\text{core}(P_i)$ to a strictly smaller subset $T \subset \text{core}(P_i)$. These correspond to Lines 11 and 22;
3. The updates that refine the partition sets $\{P_i\}_{i=1}^T$ by moving a subset $T \subseteq P_i \setminus \text{core}(P_i)$ from the partition set $P_i$ to a different partition set $P_j$, for some $P_i \neq P_j$. These correspond to Lines 15, 17, and 24.

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 4.10 that the algorithm outputs $\ell \leq k$ clusters. Secondly, for a fixed value of $\ell$, the second type of updates occurs at most $n$ times, since each update decreases the size of some core($P_i$) by at least one. Finally, for a fixed $\ell$ and a fixed configuration of core sets $\text{core}(P_i) \subseteq P_i$, 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., $\sum_{i \neq j} w(P_i, P_j)$, decreases by at least $w_{\min}$. Combining everything together proves the lemma.

Finally, we bring everything together and prove Lemma 4.4.

**Proof of Lemma 4.4.** 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 4.11 with the choices of $\phi_{in}, \phi_{out}$ in (4.10), we obtain for all $1 \leq i \leq \ell$ that $\Phi_G(P_i) \leq \phi_{out} = O(k^5 \cdot \sqrt{\lambda_\ell})$ and $\Phi_{G[P_{i+1}]} \geq \phi_{in}/4 = \Omega(\lambda_{k+1}^2/k^4)$. Hence, Properties (A1) and (A2) hold for every $P_i$.

To analyse Property (A3), we fix an arbitrary node $N \in S_i$ that belongs to the partition set $P_i$ with core set $\text{core}(P_i)$. By definition, we have that

$$
w(N, V \setminus P_i) = w(N^+, V \setminus P_i) + w(N^-, V \setminus P_i).
$$

We study $w(N^+, V \setminus P_i)$ and $w(N^-, V \setminus P_i)$ separately.
Bounding the value of $w(N^+, V \setminus P_i)$: We analyse $w(N^+, V \setminus P_i)$ by the following case distinction.

Case 1: $\text{vol}(N^+) \leq \text{vol}(\text{core}(P_i))/2$. By Item (4) of Claim 4.11 we know that

$$\varphi(N^+, \text{core}(P_i)) \geq \frac{1}{3(k+1)},$$

which is equivalent to

$$3(k+1) \cdot \frac{\text{vol}(\text{core}(P_i))}{\text{vol}(N^+)} \cdot w(N^+ \rightarrow \text{core}(P_i)) \geq w(N^+ \rightarrow V \setminus \text{core}(P_i)).$$

This implies that

$$6(k+1) \cdot w(N^+ \rightarrow \text{core}(P_i)) \geq w(N^+ \rightarrow V \setminus \text{core}(P_i)),$$

and we have that

$$w(N^+, V \setminus P_i) \leq w(N^+ \rightarrow V \setminus \text{core}(P_i)) \leq 6(k+1) \cdot w(N^+ \rightarrow \text{core}(P_i)) \leq 6(k+1) \cdot \text{vol}_{G[P_i]}(N^+).$$

Case 2: $\text{vol}(N^+) > \text{vol}(\text{core}(P_i))/2$. We have that

$$w(N^+, V \setminus P_i) \leq w(\text{core}(P_i), V \setminus P_i) \leq w(\text{core}(P_i), V \setminus \text{core}(P_i)) = \text{vol}(\text{core}(P_i)) \cdot \Phi_G(\text{core}(P_i)) \leq \text{vol}(\text{core}(P_i)) \cdot \frac{\phi_{\text{out}}}{k+1} \cdot \text{vol}(N^+),$$

where the third inequality follows by Item (1) of Claim 4.11. Therefore, we have that

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

where the last inequality follows by (4.16). We further combine (4.16) with (4.17), and obtain that

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

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

Therefore, combining the two cases above gives us that

$$w(N^+, V \setminus P_i) \leq 6(k+1) \cdot \text{vol}_{G[P_i]}(N^+).$$

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

Case 1: $\text{vol}(N^-) \leq \text{vol}(P_i)/2$. By Item (5) of Claim 4.11, we know that

$$w(N^- \rightarrow P_i) \geq w(N^- \rightarrow V \setminus P_i) \cdot \frac{1}{(k+1)},$$

which gives us that

$$w(N^-, V \setminus P_i) \leq (k+1) \cdot w(N^- \rightarrow P_i) \leq (k+1) \cdot \text{vol}_{G[P_i]}(N^-).$$

Case 2: $\text{vol}(N^-) > \text{vol}(P_i)/2$. In this case, we have that

$$w(N^-, V \setminus P_i) \leq w(P_i, V \setminus P_i) = \Phi_G(P_i) \cdot \text{vol}(P_i) \leq \phi_{\text{out}} \cdot \text{vol}(P_i) \leq 2\phi_{\text{out}} \cdot \text{vol}(N^-),$$

and we have that
where the second inequality follows by Item (2) of Claim 4.11. This implies that
\[
\text{vol}_{G[P_i]}(N^-) = \text{vol}(N^-) - w(N^-, V \setminus P_i) \geq (1 - 2\phi_{\text{out}}) \cdot \text{vol}(N^-),
\]  
where the last inequality follows by (4.19). Finally, combining (4.19) and (4.20) gives us that
\[
w(N^-, V \setminus P_i) \leq \frac{2\phi_{\text{out}}}{1 - 2\phi_{\text{out}}} \cdot \text{vol}_{G[P_i]}(N^-) \leq 2 \cdot \text{vol}_{G[P_i]}(N^-),
\]
where the last inequality follows by our assumption that \( \phi_{\text{out}} < 1/3 \). Therefore, combining the two cases together gives us that
\[
w(N^-, V \setminus P_i) \leq (k + 1) \cdot \text{vol}_{G[P_i]}(N^-).
\]  
(4.21)

Our claimed property (A3) follows by summing the inequalities in (4.18) and (4.21) and the fact that \( \text{vol}_{G[P_i]}(N) = \text{vol}_{G[P_i]}(N^+) + \text{vol}_{G[P_i]}(N^-) \).

Finally, we analyse the runtime of the algorithm. By Claims 4.13 and 4.14, 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 \( O(k \cdot n \cdot \text{vol}(G)/w_{\text{min}}) \). Notice that this quantity is upper bounded by \( O(\text{poly}(n)) \) given our assumption that \( w_{\text{max}}/w_{\text{min}} = O(\text{poly}(n)) \). This completes the proof. \( \square \)

5 Experiments

We experimentally evaluate the performance of our proposed algorithm, and compare it against the three well-known linkage heuristics for computing hierarchical clustering trees, and different variants of the algorithm proposed in [CAKMT17], i.e., \textbf{Linkage++}, on both synthetic and real-world data sets. At a high level, \textbf{Linkage++} consists of the following three steps:

(i) Project the input data points into a lower dimensional Euclidean subspace;

(ii) Run the \textbf{Single Linkage} algorithm [CAKMTM19] until \( k \) clusters are left;

(iii) Run a \textbf{Density} based linkage algorithm on the \( k \) clusters until one cluster is left.

Specifically, our algorithm \textbf{PruneMerge} will be compared against the following 6 algorithms:

- \textbf{Average Linkage}, \textbf{Complete Linkage}, and \textbf{Single Linkage}: the three well-known linkage algorithms studied in the literature. We refer the reader to [CAKMTM19] for a complete description;

- \textbf{Linkage++}, \textbf{PCA+} and \textbf{Density}: the algorithm proposed in [CAKMT17], together with two variants also studied in [CAKMT17]. The algorithm \textbf{PCA+} corresponds to running Steps (i) and (ii) of \textbf{Linkage++} until one cluster is left (as opposed to \( k \) clusters), while \textbf{Density} corresponds to running Steps (i) and (iii) of \textbf{Linkage++}.

All algorithms were implemented in Python 3.8 and the experiments were performed using an Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz processor, with 16 GB RAM. All of the reported costs below are averaged over 5 independent runs. For the parameter \( \phi_{\text{in}} \) used in the proof of Lemma 4.4 (see (4.10)), we set in our implementation \( \phi_{\text{in}} = \max \left\{ \frac{1}{140(k+1)^2}, 2 \cdot \lambda_k \right\} \). This value is used throughout the experiments reported here. Our code can be downloaded from

5.1 Experiments on synthetic data sets

We first compare the performance of our algorithm with the aforementioned other algorithms on synthetic data sets.

Clusters of the same size. Our first set of experiments employ input graphs generated according to random stochastic models, where all clusters have the same size. For our first experiment, we look at graphs generated according to the standard Stochastic Block Model (SBM). We first set the number of clusters as \( k = 3 \), and the number of vertices in each cluster \( \{P_i\}_{i=1}^3 \) as 1,000. We assume that any pair of vertices within each cluster is connected by an edge with probability \( p \), and any pair of vertices from different clusters is connected by an edge with probability \( q \). We fix the value \( q = 0.002 \), and consider different values of \( p \in [0.04, 0.2] \).

Our experimental results are illustrated in Figure 7(a).

For our second experiment, we consider graphs generated according to a hierarchical stochastic block model (HSBM) [CAKMTM19]. This model assumes the existence of a ground-truth hierarchical structure of the clusters. For the specific choice of parameters, we set the number of clusters as \( k = 5 \), and the number of vertices in each cluster \( \{P_i\}_{i=1}^5 \) as 600. For every pair of vertices \((u, v) \in P_i \times P_j\), 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 \( j \in \{4, 5\} \), \( q_{i,j} = q_{j,i} = q_{\text{min}} \); (ii) for \( i \in \{1, 2\} \), \( q_{i,3} = q_{3,i} = 2 \cdot q_{\text{min}} \); (iii) \( q_{4,5} = q_{5,4} = 2 \cdot q_{\text{min}} \); (iv) \( q_{1,2} = q_{2,1} = 3 \cdot q_{\text{min}} \). We fix the value \( q_{\text{min}} = 0.0005 \) and consider different values of \( p \in [0.04, 0.2] \). Our results are reported in Figure 7(b). We remark that this choice of parameters resembles similarities with [CAKMT17], and this ensures that the underlying graphs exhibit a ground truth hierarchical structure of clusters.

As reported in Figure 7, our experimental results for both sets of graphs are similar, and the performance of our algorithm is marginally better than Linkage++. This is well expected, as Linkage++ is specifically designed for the HSBM, in which all the clusters have the same inner density characterised by parameter \( p \), and their algorithm achieves a \((1 + o(1))\)-approximation for those instances.

![Figure 7](image-url): Results for clusters of the same size. 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 PruneMerge. Figure (a) corresponds to inputs generated according to the SBM, while Figure (b) corresponds to those generated according to the HSBM.

Clusters with non-uniform densities. Next we study graphs in which edges are present non-uniformly within each cluster (e.g., Figure 1(a) discussed earlier). Specifically, we set \( k = 3 \), \( |P_i| = 1000 \), \( q = 0.002 \), \( p = 0.06 \), and every pair of vertices \((u, v) \in P_i \times P_j\) is connected by an edge with probability \( p \) if \( i = j \) and probability \( q \) otherwise. Moreover, we choose a random set \( S_i \subset P_i \) of size \( |S_i| = c_p \cdot |P_i| \) from each cluster, and add edges to connect every
pair of vertices in each $S_i$ so that the vertices of each $S_i$ form a clique. By setting different values of $c_p \in [0.05, 0.4]$, the performance of our algorithm is about 20% – 50% better than Linkage++ with respect to the cost value of the constructed tree, see Figure 8(a) for detailed results. To explain the outperformance of our algorithm, notice that, by adding a clique into some cluster, the cluster structure is usually preserved with respect to $(\Phi_{\text{in}}, \Phi_{\text{out}})$ or similar eigen-gap assumptions on well-clustered graphs. However, the existence of such a clique within some cluster would make the vertices’ degrees highly unbalanced; as such many clustering algorithms that involve the matrix perturbation theory in their analysis might not work well.

**Clusters of different sizes.** To further highlight the significance of our algorithm on synthetic graphs of non-symmetric structures among the clusters, we study the graphs in which the clusters have different sizes. We choose the same set of $k$ and $q$ values as before ($k = 3, q = 0.002$), but set the sizes of the clusters to be $|P_1| = 1900, |P_2| = 900$ and $|P_3| = 200$. Every pair of vertices $u, v \in P_i$, for $i \in \{1, 2\}$ is connected by an edge with probability $p_1 = 0.06$, while pairs of vertices $u, v \in P_3$ are connected with probability $^7p_2 = 5 \cdot p_1 = 0.3$. We further plant a clique $S_i \subset P_i$ of size $|S_i| = c_p \cdot P_i$ for each cluster $P_i$, as in the previous set of experiments. By choosing different values of $c_p$ from $[0.05, 0.4]$, our results are reported in Figure 8(b), demonstrating that our algorithm performs better than the ones in [CAKMT17].

**5.2 Experiments on real-world data sets**

To evaluate the performance of our algorithm on real-world data sets, we follow the sequence of recent work on hierarchical clustering [ACAH19, CAKMT17, MRS+19, RP17], all of which are based on the following 5 data sets from the Scikit-learn library [PVG+11] as well as the UCI ML repository [UCI]: Iris, Wine, Cancer, Boston and Newsgroup$^8$. Similar with [RP17], for each data set we construct the similarity graph based on the Gaussian kernel, in which the $\sigma$-value is chosen according to the standard heuristic [NJW01]. As reported in Figure 9, our algorithm performs marginally worse than Linkage++ and significantly better than PCA++.

---

$^7$Such choice of $p_2$ is to compensate for the small size of cluster $P_3$, and this ensures that the outer conductance $\Phi_{\text{out}}(P_3)$ is low.

$^8$Due to the very large size of this data set, 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 8: Results for graphs with non-uniform densities, or different sizes. The x-axis represents the $c_p$-values, while the y-axis represents the cost of the algorithms’ constructed trees normalised by the cost of the ones constructed by PruneMerge. Figure (a) corresponds to inputs where all clusters have the same size, while in Figure (b) the clusters have different sizes.
Our experimental results on synthetic data sets demonstrate that our presented algorithm PruneMerge not only has excellent theoretical guarantees, but also produces output of lower cost than the previous algorithm Linkage++. In particular, the outperformance of our algorithm is best illustrated on graphs whose clusters have asymmetric internal structure and non-uniform densities. On the other side, our experimental results on real-world data sets show that the performance of PruneMerge is inferior to Linkage++ and especially to Average Linkage. We believe that developing more efficient algorithms for well-clustered graphs is a very meaningful direction for future work.

Finally, our experimental results indicate that the Average Linkage algorithm performs extremely well on all tested instances, when compared to PruneMerge and Linkage++. This leads to the open question whether Average Linkage achieves an $O(1)$-approximation for well-clustered graphs, although it fails to achieve this for general graphs [CAKMTM19]. In our point of view, the answer to this question could help us design more efficient algorithms for hierarchical clustering that not only work in practice, but also have rigorous theoretical guarantees.

References


