Approximating Dasgupta Cost in Sublinear Time from a Few Random Seeds

Michael Kapralov
EPFL

Akash Kumar
IIT Bombay

Silvio Lattanzi
Google Research

Aida Mousavifar
Google

Weronika Wrzos-Kaminska
EPFL

Abstract

Testing graph cluster structure has been a central object of study in property testing since the foundational work of Goldreich and Ron [STOC'96] on expansion testing, i.e. the problem of distinguishing between a single cluster (an expander) and a graph that is far from a single cluster. More generally, a (k, ϵ)-clusterable graph G is a graph whose vertex set admits a partition into k induced expanders, each with outer conductance bounded by ϵ. A recent line of work initiated by Czumaj, Peng and Sohler [STOC'15] has shown how to test whether a graph is close to (k, ϵ)-clusterable, and to locally determine which cluster a given vertex belongs to with misclassification rate $\approx \epsilon$, but no sublinear time algorithms for learning the structure of inter-cluster connections are known. As a simple example, can one locally distinguish between the 'cluster graph' forming a line and a clique?

In this paper, we consider the problem of testing the *hierarchical* cluster structure of (k, ϵ)-clusterable graphs in sublinear time. Our measure of hierarchical clusterability is the well-established Dasgupta cost, and our main result is an algorithm that approximates Dasgupta cost of a (k, ϵ)-clusterable graph in sublinear time, using a small number of randomly chosen *seed vertices* for which cluster labels are known. Our main result is an $O(\sqrt{\log k})$ approximation to Dasgupta cost of G in $\approx n^{1/2+O(\epsilon)}$ time using $\approx n^{1/3}$ seeds, effectively giving a sublinear time simulation of the algorithm of Charikar and Chatziafratis [SODA'17] on clusterable graphs. To the best of our knowledge, ours is the first result on approximating the hierarchical clustering properties of such graphs in sublinear time.

1 Introduction

Graph clustering is a central problem in data analysis, with applications in a wide variety of scientific disciplines from data mining to social science, statistics and more. The overall objective in these problems is to partition the vertex set of the graph into disjoint "well connected" subgraphs which are sparsely connected to each other. It is quite common in the practice of graph clustering that besides the graph itself one is given a list of vertices with correct cluster labels for them, and one must extend this limited amount of cleanly labeled data to a clustering of the entire graph. This corresponds to the widely used *seeded* model (see, e.g., [BBM02] and numerous follow up works, e.g., [DBE99, KBDM09, SK02, AB15]). The central question that we consider in this paper is

> What can be learned about the cluster structure of the input graph from a few seed nodes in sublinear time?

Formally, we work with the classical model for well-clusterable graphs [CPS15], where the input graph $G = (V, E)$ is assumed to admit a partitioning into a disjoint union of k induced expanders $C_1, \ldots, C_k$ with outer conductance bounded by $\epsilon \ll 1$ and inner conductance being $\Omega(1)$. We refer to such instances as $(k, \Omega(1), \epsilon)$-*clusterable* graphs, or (k, ϵ)-clusterable graphs for short. Such graphs have been the focus of significant attention in the property testing literature [CS07, KS08, NS10], starting from the seminal work of [GR11]. A recent line of work has shown how to design nearly optimal sublinear time clustering oracles for such graphs, i.e., algorithms that can consistently answer clustering queries on such a graph from a local exploration only. However, existing works do not show how to learn the structure of connections between the clusters. In particular, to the best of our knowledge, no approach in existing literature can resolve the following simple question:

> Distinguish between the clusters being arranged in a line and the clusters forming an (appropriately subsampled) clique (See Fig. 1) .

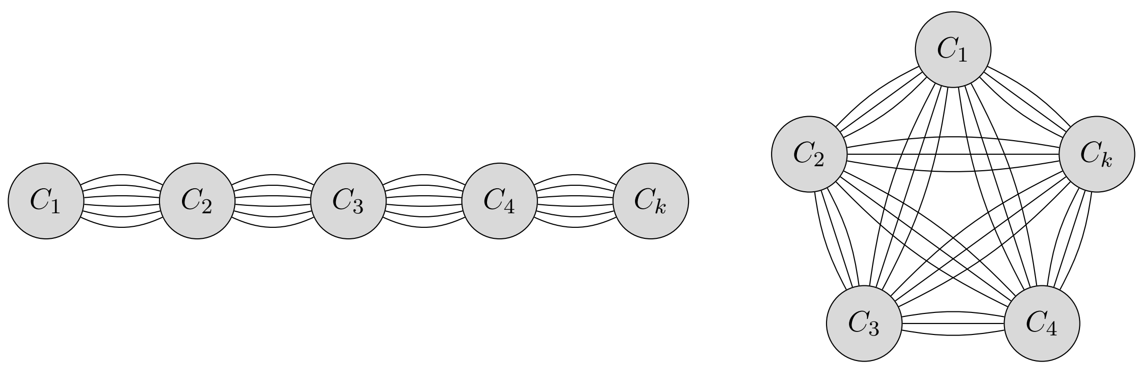

Figure 1: Clusters arranged in a line (Left); Clusters forming a clique (Right)

More generally, we would like to design a sublinear time algorithm that approximates the *hierarchical clustering* properties of k-clusterable graphs. Hierarchical clustering is a useful primitive in machine learning and data science with essential applications in information retrieval [MRS08, Ber06], social networks [GSZ+11] and phylogenetics [ESBB98]. Informally, in hierarchical clustering the objective is to construct a hierarchy of partitions that explain the cluster structure of the graph at different scales – note that such a partitioning looks very different in the two cases (line vs. clique) above. Formally, the quality of such a hierarchy of partitions is often evaluated using Dasgupta cost [Das16], and the main question studied in our paper is

> Is it possible to approximate the Dasgupta cost of a $(k, \Omega(1), \epsilon)$-clusterable graph using few queries to the input graph and a few correctly clustered seed vertices?

In practice, an algorithm operating in the seeded model [BBM02] most often does not have full control over the seeds, but rather is given a list generated by some external process. To model this, we assume that the seed vertices are sampled independently from the input graph, with probability proportional to their degrees: we refer to this model as the *random sample model*.

The case $k = 1$, i.e., approximating the Dasgupta cost of an expander. When $k = 1$, our input is a single expander, i.e., a single cluster, we approximate its Dasgupta cost in sublinear time using degree queries on the seeds. At first glance one might think that Dasgupta cost of an expander can be approximated well simply as a function of its number of vertices and average degree, but this is only the case for *regular* expanders. The irregular case is nontrivial, a poly$(1/\varphi)$ approximation was recently given by [MS21]. As our first result, we give an algorithm approximating Dasgupta cost of an (irregular) φ-expander using $\approx n^{1/3}$ seed vertices (and degree queries on these vertices). This, somewhat surprisingly, turns out to be a tight bound. Specifically, we show

Theorem 1.1 (Approximating Dasgupta cost of an expander)**.** *Dasgupta cost of a φ-expander can be approximated to within a poly$(1/\varphi)$ factor using degree queries on $\approx n^{1/3}$ seed vertices. Furthermore, the bound of $\approx n^{1/3}$ is tight up to polylogarithmic factors.*

The case $k > 1$ For $k > 1$ we leverage recent results on clustering oracles to decompose the problem of approximating the Dasgupta cost into two: **(1)** approximating Dasgupta cost of individual clusters and **(2)** approximating Dasgupta cost of the contracted graph, in which each cluster is contracted into a supernode. Such a decomposition is only possible for bounded degree graphs, see Example 4.2 in [MS21], so this is the setting we work in. We show that access to a few seed vertices is sufficient to obtain oracle access to the cut function (and, more generally, quadratic form of the Laplacian) of the contracted graph in time $\approx n^{1/2+O(\epsilon)}$. Our main result is Theorem 1.2 below:Theorem 1.2 below:

Theorem 1.2 (Informal version of Theorem 2.1)**.** *There exists an algorithm that for every $(k, \Omega(1), \epsilon)$-clusterable bounded degree graph $G = (V, E)$ estimates the Dasgupta cost of G up to $O(\sqrt{\log k})$ factor in the random sample model in time $\approx n^{1/2+O(\epsilon)} \cdot (d_{\max})^{O(1)}$.*

Remark 1.3. We remark that our algorithm for estimating Dasgupta cost from Theorem 2.1 can be made to provide an oracle access to a low cost hierarchical clustering tree.

Remark 1.4. One can verify by adapting the lower bound of $\Omega(n^{1/2})$ on expansion testing due to Goldreich and Ron [GR02] that at least $\Omega(\sqrt{n/k})$ queries are needed for a $o(k/\log k)$ approximation for constant k in this model. The proof is a rather direct adaptation of the classical result of Goldreich and Ron, and we therefore do not present it.

Remark 1.5. Recall that in our *random sample* model for seed vertices the seeds are sampled independently with probability proportional to their degrees. This model matches quite closely what happens in practice in the sense that the algorithm does not always have full control over the seeds [BBM02]. One can also consider the stronger model in which the algorithm can ask for correct label of *any* vertex of its choosing. This model is significantly stronger, and in particular, one can design an algorithm for obtaining the same approximation of Dasgupta cost as our Theorem 1.2 above, but with time complexity polynomial in d, $\log n$ and $1/\epsilon$.

We note that the currently best known approximation to the Dasgupta cost on n-vertex graphs is $O(\sqrt{\log n})$, achieved by the recursive sparsest cut algorithm of [CC17]. Our approximation is $O(\sqrt{\log k})$, matching what the Charikar and Chatziafratis algorithm achieves on k-node graphs. In fact, our main technical contribution is an efficient way of simulating this algorithm in sublinear time on k-clusterable graphs.

Related work on $(k, \Omega(1), \epsilon)$-clusterable graphs. Such graphs have been extensively studied in the property testing framework as well as local computation models. Its testing version, where one essentially wants to determine k, the number of clusters in G, in sublinear time, generalizes the well-studied problem of testing graph expansion, where one wants to distinguish between an expander (i.e. a good single cluster) and a graph with a sparse cut (i.e., at least two clusters). [GR11] showed that expansion testing requires $\Omega(n^{1/2})$ queries, then [CS07, KS08, NS10] developed algorithms to distinguish an expander from a graph that is far from a graph with conductance ϵ in time $\approx n^{1/2+O(\epsilon)}$, which the recent work of [CKK$^+$18] showed to be tight. The setting of $k > 2$ has seen a lot of attention recently [CPS15, CKK$^+$18, Pen20, GKL$^+$21a], with close to information theoretically optimal clustering oracles, i.e., small space data structures that provide quick access to an approximate clustering, obtained in [GKL$^+$21a]. More recently, [MS21] studied hierarchical clustering of k-clusterable graphs and developed a nearly linear time algorithm that approximates the Dasgupta cost of the graph up to a constant factor. However, their algorithm to work requires significantly stronger assumptions on the input data i.e., $\epsilon \ll 1/k^{O(1)}$, and their algorithm does not run in *sublinear time*. Note that the problem of estimating the Dasgupta cost becomes non-trivial when $\epsilon \gg \frac{1}{k}$, i.e., when the Dasgupta cost of the graph is dominated by the outgoing edges between different clusters[1].

The most closely related work on our setting is [KKLM23] where the authors provide a sublinear algorithm for hierarchical clustering. However, their algorithm works under significantly stronger assumptions on their input instance. They introduce the notion of hierarchically clusterable graphs, which assumes a planted hierarchical clustering structure not only at the bottom level of the hierarchy but at *every level*. Their result relies on several properties of such graphs. In contrast, we only assume that the input graph is k-clusterable. For this reason we cannot use the techniques developed in [KKLM23], and we need to develop a completely new approach.

Very recently, [AKLP22, ACL$^+$22] considered the problem of hierarchical clustering under Dasgupta objective in the streaming model. Both papers give a one pass $\widetilde{O}(n)$ memory streaming algorithm which finds a tree with Dasgupta cost within an $O(\sqrt{\log n})$ factor of the optimum in polynomial time. Additionally, [AKLP22] also considers this problem in the query model and presents an $O(\sqrt{\log n})$ approximate hierarchical clustering using $\widetilde{O}(n)$ queries without making any clusterability assumptions of the input graph. On the other hand, our algorithms assume the graph is k-clusterable and approximate the Dasgupta cost within an $O(\sqrt{\log k})$ in sublinear time.

Related work on hierarchical clustering. We briefly review developments in the area of algorithms for hierarchical clustering since the introduction of Dasgupta's objective function. Dasgupta designed an algorithm based on recursive sparsest-cut that provides $O(\log^{3/2} n)$ approximation for his objective function. This was improved by Charikar and Chatizafratis who showed that the recursive sparsest-cut algorithm already returns a tree with approximation guarantee $O(\sqrt{\log n})$ [CC17]. Furthermore, they showed that it's impossible to approximate the Dasgupta cost within a constant factor in general graphs under the Small-Set Expansion hypothesis. More recently, [CAKMTM18] studied this problem in a regime in which the input graph is sampled from a Hierarchical Stochastic Block Model [CAKMTM18]. They construct a tree in nearly linear time that approximates Dasgupta cost of the graph up to a constant factor. [CAKMTM18] uses a type of hierarchical stochastic block model, which generates close to regular expanders with high probability, and their analysis crucially relies on having dense clusters and large degrees. Our model allows for arbitrary expanders as opposed to dense random graphs and is more expressive in this sense.

Related work in semi-supervised active clustering. We note that our model is also related to the semi-supervised active clustering framework (SSAC) introduced in [AKB16]. In this model we are given a set X of n points and an oracle answering to same-cluster queries of the form "are these two points in the same cluster?". Thanks to its elegance and applications to crowdsourcing, the model received a lot of attention and has been extensively studied both in theory [ABJ18, ABJK18, BCBLP20, BCLP21, HMMP19, MP17, MS17a, MS17b, SS19, VRG19]

[1] For instance, in a d-regular, (k, φ, ϵ)-clusterable graph, one can easily show that the Dasgupta cost is at least $\Omega(\frac{\varphi \cdot d \cdot n^2}{k})$, simply because of the contribution of the k induced φ-expanders. On the other hand, the total number of edges running between the clusters is bounded by $\epsilon \cdot d \cdot n$, and therefore their total contribution to the Dasgupta cost is $O(\epsilon \cdot d \cdot n^2)$. Thus, the problem becomes non-trivial when $\epsilon \gg \frac{1}{k}$.

and in practice [FGSS18, GNK$^+$15, VGM15, VGMP17] — see also [EZK18] for other types of queries.

1.1 Basic definitions

Definition 1 (Inner and outer conductance). Let $G = (V, E)$ be a graph. For a set $C \subseteq V$ and a set $S \subseteq C$, let $E(S, C \setminus S)$ be the set of edges with one endpoint in S and the other in $C \setminus S$. The *conductance of S within C*, is $\phi^G_C(S) = \frac{|E(S,C\setminus S)|}{\mathrm{vol}(S)}$. The *outer conductance* of C is defined to be $\phi^G_{\mathrm{out}}(C) = \phi^G_V(C) = \frac{|E(C,V\setminus C)|}{\mathrm{vol}(C)}$. The *inner conductance* of $C \subseteq V$ is defined to be $\phi^G_{\mathrm{in}}(C) = \min_{S\subseteq C, 0<|S|\le \frac{\mathrm{vol}(C)}{2}} \phi^G_C(S)$ if $|C| > 1$ and one otherwise.

For Theorem 1.2, we assume that the degree of every vertex is maximal by adding self-loops, and use the notion of conductance corresponding to the graph with the added self-loops. We define *k-clusterable* graphs as a class of instances that can be partitioned into k expanders with small outer conductance:

Definition 2 ((k, φ, ϵ)-clustering). Let $G = (V, E)$ be a graph. A (k, φ, ϵ)-clustering of G is a partition of vertices V into disjoint subsets $C_1, \ldots, C_k$ such that for all $i \in [k]$, $\phi^G_{\mathrm{in}}(C_i) \ge \varphi$, $\phi^G_{\mathrm{out}}(C_i) \le \epsilon$ and for all $i, j \in [k]$ one has $\eta \coloneqq \frac{|C_i|}{|C_j|} \in O(1)$. A graph G is called (k, φ, ϵ)-clusterable if there exists a (k, φ, ϵ)-clustering for G.

Dasgupta cost. Hierarchical clustering is the task of partitioning vertices of a graph into nested clusters. The nested partitions can be represented by a rooted tree whose leaves correspond to the vertices of graph, and whose internal nodes represent the clusters of vertices. Dasgupta introduced a natural optimization framework for formulating hierarchical clustering tasks as an optimization problem [Das16]. We recall this framework now. Let T be any rooted tree whose leaves are vertices of the graph. For any node x of T, let $T[x]$ be the subtree rooted at x, and let $\textsc{leaves}(T[x]) \subseteq V$ denote the leaves of this subtree. For leaves $x, y \in V$, let $\mathrm{LCA}(x, y)$ denote the lowest common ancestor of x and y in T. In other words, $T[\mathrm{LCA}(x, y)]$ is the smallest subtree whose leaves contain both x and y.

Definition 3 (Dasgupta cost [Das16]). The Dasgupta cost of the tree T for the graph $G = (V, E)$ is defined to be $\mathrm{COST}_G(T) = \sum_{\{x,y\}\in E} |\textsc{leaves}(T[\mathrm{LCA}(x, y)])|$.

The random sample model for seed vertices. We consider a *random sample* model for seed vertices, in which the algorithm is given a (multi)set S of *seed* vertices, which are sampled independently with probability proportional to their degrees, together with their cluster label.

2 Technical overview

In this section, we give an overview of our main algorithmic result, stated below as Theorem 2.1 (formal version of Theorem 1.2). It postulates a sublinear time algorithm for estimating the Dasgupta cost of k-clusterable graphs. Here, we use O^*-notation to suppress $\mathrm{poly}(k)$, $\mathrm{poly}(1/\varphi)$, $\mathrm{poly}(1/\epsilon)$ and $\mathrm{polylog}\, n$-factors.

Theorem 2.1. *Let $k \ge 2$, $\varphi \in (0, 1)$ and $\frac{\epsilon}{\varphi^2}$ be a sufficiently small constant. Let $G = (V, E)$ be a bounded degree graph that admits a (k, φ, ϵ)-clustering $C_1, \ldots, C_k$. Let $|V| = n$.*

There exists an algorithm ($\textsc{EstimatedCost}(G)$*; Algorithm 1) that w.h.p. estimates the optimum Dasgupta cost of G within an $O\left(\frac{\sqrt{\log k}}{\varphi^{O(1)}}\right)$ factor in time $O^*\left(n^{1/2+O(\epsilon/\varphi^2)} \cdot (d_{\max})^{O(1)}\right)$ using $O^*\left(n^{O(\epsilon/\varphi^2)} \cdot (d_{\max})^{O(1)}\right)$ seed queries.*

Our algorithm consists of two main parts: First, we estimate the contribution from the inter-cluster edges to the Dasgupta cost. A natural approach is to contract the clusters $C_1, \ldots, C_k$ into supernodes, and use the Dasgupta cost of the contracted graph (defined below) as a proxy.

Definition 4 (Contracted graph). Let $G = (V, E)$ be a graph and let $\mathcal{C} = (C_1, \ldots, C_k)$ denote a partition of V into disjoint subsets. We say that the weighted graph $H = \left([k], \binom{[k]}{2}, W, w\right)$ is a contraction of G with respect to the partition $\mathcal{C}$ if for every $i, j \in [k]$ we have $W(i,j) = |E(C_i, C_j)|$, and for every $i \in [k]$ we have $w(i) = |C_i|$. We denote the contraction of G with respect to the partition $\mathcal{C}$ by $H = G/\mathcal{C}$.

The problem is of course that it is not clear how to get access to this contracted graph in sublinear time, and our main contribution is a way of doing so. Our approach amounts to first obtaining access to the quadratic form of the Laplacian of the contracted graph H, and then using the hierarchical clustering algorithm of [CC17] on the corresponding approximation to the contracted graph. Thus, we essentially show how to simulate the algorithm of [CC17] in sublinear time on (k, φ, ϵ)-clusterable graphs.

The procedure TotalClustersCost approximates the contribution from the internal cluster edges to the Dasgupta cost.

Algorithm 1 below presents our estimator for the Dasgupta cost of the graph.

Algorithm 1 EstimatedCost(G) time $\approx n^{1/2+O(\epsilon)}$

$\xi \leftarrow \left(\frac{\varphi}{k \cdot d_{\max}}\right)^{O(1)}$
$\mathcal{D} \leftarrow$ InitializeWeightedDotProductOracle(G, ξ) *# See Algorithm 7*
$\widetilde{H} \leftarrow$ ApproxContractedGraph($G, \xi, \mathcal{D}$) *#The Laplacian $\widetilde{\mathcal{L}}$ of $\widetilde{H}$ satisfies Equation (4)*
$\widetilde{T} \leftarrow$ WeightedRecursiveSparsestCut($\widetilde{H}$) *#Weighted version of Algorithm of [CC17]*
EST $\leftarrow O\left(\frac{1}{\varphi^2}\right) \cdot \mathrm{WCOST}_{\widetilde{H}}(\widetilde{T}) +$ TotalClustersCost(G) $+ O\left(\frac{\xi m n k^2}{\varphi^2}\right)$
return EST

Our algorithm uses a weighted definition of Dasgupta cost (Definition 9), which we denote WCOST, to relate the cost of G and the contracted graph H. Then, our estimate EST in Algorithm 1 simply sums the contribution from the weighted Dasgupta cost of the tree $\widetilde{T}$ on the contracted graph $\widetilde{H}$, with the contribution from the clusters. We want to ensure that the estimate always provides an upper bound on the optimal Dasgupta cost of G. To this end, we scale the weighted Dasgupta cost $\mathrm{WCOST}_{\widetilde{H}}(\widetilde{T})$ up by a factor of $O\left(\frac{1}{\varphi^2}\right)$ (to account for the multiplicative error), and add a term on the order of $\frac{\xi m n k^2}{\varphi^2}$ (to account for the additive error). That way we obtain an estimate EST such that

$$\mathrm{COST}(G) \leq \mathrm{EST} \leq O\left(\frac{\sqrt{\log k}}{\varphi^{O(1)}}\right) \mathrm{COST}(G),$$

where $\mathrm{COST}(G)$ denotes the optimum Dasgupta cost of G.

We outline the main ideas behind accessing the contracted graph in Section 2.2, and present the complete analysis in Section A. The TotalClustersCost procedure simply outputs a fixed value that depends on n, d, and k. We provide more details on this in Section 2.1 and present to full analysis in Section B.3.

We remark that with a little post-processing, our algorithms for estimating Dasgupta cost can be adapted to recover a low-cost hierarchical-clustering tree. To construct such a tree we first construct a tree $\widetilde{T}$ with k leaves on the contracted graph. The algorithm constructs $\widetilde{T}$ in sublinear time $\approx n^{1/2+O(\epsilon)}$. Then, for every cluster C_i one can construct a particular tree $\mathcal{T}^i_{\text{deg}}$ using (Algorithm 1 of [MS21]) on the vertices of C_i. Finally, we can extend the leaf i of the tree $\widetilde{T}$ by adding trees $\mathcal{T}^i$ as its direct child. Note that constructing $\mathcal{T}^i_{\text{deg}}$ explicitly takes time $O(|C_i|)$, however, this step is only required if one intends to output the full hierarchical-clustering tree of G. Otherwise, for only estimating $\text{COST}(G)$, we can estimate $\text{COST}(\mathcal{T}^i_{\text{deg}})$ as a function of the cluster size and the degree without explicitly constructing $\mathcal{T}^i_{\text{deg}}$.

2.1 Estimating Dasgupta cost of an expander using seed queries

In this section, we design an algorithm for estimating the Dasgupta cost of an irregular φ-expander up to $\text{poly}(1/\varphi)$ factor using $\approx n^{1/3}$ seed queries. We also prove that this is optimal (Theorem 2.4) in subsection B.4. Later in the paper (in Section B.3), we approximate the contribution of the clusters to the Dasgupta cost of a d-regular (k, φ, ϵ)-clusterable graph. There, a more basic approach suffices. In this section, we focus on a single but irregular φ-expander.

Theorem 2.2. *Let $G = (V, E)$ be a φ-expander (possibly with self-loops). Let T^* denote the tree with optimum Dasgupta cost for G. Then procedure* ClusterCost *(Algorithm 3), uses $O^*\left(n^{1/3}\right)$ seed queries and with probability $1 - n^{-101}$ returns a value such that:*

$$COST(T^*) \le \textsc{ClusterCost}(G) \le O\left(\frac{1}{\varphi^5}\right) \cdot COST(T^*).$$

We now outline the proof of Theorem 2.2.

Let G be a φ-expander, i.e., $\phi_{\text{in}}(G) \ge \varphi$. To estimate the Dasgupta cost of G, we use Theorem 2.3 from [MS21]. This result shows that there is a specific tree called $\mathcal{T}_{\text{deg}}$ on G that approximates the Dasgupta cost of G up to $O\left(\frac{1}{\varphi^4}\right)$. For completeness, we include the algorithm (Algorithm 2) for computing $\mathcal{T}_{\text{deg}}$ from [MS21]. Note that Algorithm 2 from [MS21] runs in time $O(m + n \log n)$, however, we don't need to explicitly construct $\mathcal{T}_{\text{deg}}$. Instead, we design an algorithm that estimates the cost of $\mathcal{T}_{\text{deg}}$ in time $n^{1/3}$.

Algorithm 2 $\textsc{HCwithDegrees}(G\{V\})$ [MS21]

Input: $G = (V, E, w)$ with the ordered vertices such that $d_{v_1} \ge \ldots \ge d_{v_{|V|}}$
Output: An HC tree $\mathcal{T}_{\text{deg}}(G)$
if $|V| = 1$ **then**
  return the single vertex V as the tree
else
  $i_{\max} \coloneqq \lfloor \log_2 |V| - 1 \rfloor$; $r \coloneqq 2^{i_{\max}}$; $A \coloneqq \{v_1, \ldots, v_r\}$; $B \coloneqq V \setminus A$
  Let $\mathcal{T}_1 \coloneqq \textsc{HCwithDegrees}(G\{A\})$; $\mathcal{T}_2 \coloneqq \textsc{HCwithDegrees}(G\{B\})$
  return $\mathcal{T}_{\text{deg}}$ with $\mathcal{T}_1$ and $\mathcal{T}_2$ as the two children
end if

Theorem 2.3 (Theorem 3 in [MS21])**.** *Given any graph $G = (V, E, w)$ with inner-conductance φ as input, Algorithm 2 runs in $O(m + n \log n)$ time, and returns an HC tree $\mathcal{T}_{\text{deg}}(G)$ that satisfies $COST_G(\mathcal{T}_{\text{deg}}(G)) = O(1/\varphi^4) \cdot OPT_G$.*

Our procedure for estimating the Dasgupta cost of the tree returned by Algorithm 2 is based on a simple expression for the (approximate) cost of this tree that we derive (and later show how to approximate by sampling).

Let $G = (V, E)$ be an arbitrary expander with vertices $x_1, x_2, \ldots x_n$ ordered such that $d_1 \geq d_2 \geq \ldots \geq d_n$, where $d_i = \deg(x_i)$. We denote by $\mathcal{T}_{\deg}$ the Dasgupta Tree returned by Algorithm 1 of [MS21]. Specifically, we show that the cost $\mathrm{COST}_G(\mathcal{T}_{\deg})$ is to within an $O(1/\varphi)$ factor approximated by

$$\sum_{i=1}^{n} i \cdot d_i = \sum_{x \in V} \mathrm{rank}(x) \cdot \deg(x), \tag{1}$$

where $\deg(x)$ is the degree of x and $\mathrm{rank}(x)$ is the rank of x in the ordering of vertices of V in non-increasing order of degrees. The proof is rather direct, and is presented in the appendix (Lemmas B.1 and B.2). Our task therefore reduces to approximating (1) in sublinear time. To achieve this, we partition the vertices into buckets according to their degree: For every d between 1 and n/φ that is a power of 2, let $B_d := \{x \in V : d \leq \deg(x) < 2d\}$. We will refer to B_d as the *degree class* of d. Let $n_d := |B_d|$ denote the size of the degree class, and let r_d denote the highest rank in B_d. Note that r_d is the number of vertices in G that have degree at least d, so we have $r_d = \sum_{t \geq d} n_t$.

The vertices in B_d have ranks $r_d, r_d - 1, \ldots, r_d - n_d + 1$ and degrees in $[d, 2d]$, which gives the bounds

$$\frac{d}{2} \cdot n_d \cdot r_d \leq \sum_{i=r_d-n_d+1}^{r_d} i \cdot d \leq \sum_{x \in B_d} \mathrm{rank}(x) \cdot \deg(x) \leq \sum_{i=r_d-n_d+1}^{r_d} i \cdot 2d \leq 2d \cdot n_d \cdot r_d, \tag{2}$$

so our task is further reduced to estimating the quantity

$$\sum_d d \cdot n_d \cdot r_d. \tag{3}$$

We do so by sampling: simply sample $\approx n^{1/3}$ vertices, and approximate the number of vertices n_d and the highest rank r_d of each degree class. This is summarized in Algorithm 3 below.

Algorithm 3 $\textsc{ClusterCost}(G, S, \hat{m})$
 # S is a (multi)set of size s of vertices in $G = (V, E)$
 # $\hat{m}$ is a constant factor estimate of $|E|$

1: **for** every d between 1 and n/φ that is a power of 2 **do**
2: $\quad \hat{n}_d \leftarrow \frac{2\hat{m}}{s}|\{v \in S : d \leq \deg(v) < 2d\}|$ *# Estimate the number of vertices by sampling*
3: $\quad \hat{r}_d \leftarrow \frac{2\hat{m}}{s}|\{v \in S : d \leq \deg(v)\}|$ *# Estimate the rank by sampling*
4: **end for**
5: **return** $\sum_d \hat{n}_d \cdot \hat{r}_d$

While the algorithm is simple, the analysis is quite interesting, and the bound of $n^{1/3}$ on the number of seeds is tight! We now outline the main ideas behind the analysis of the algorithm.

Ideally, we would like to estimate the number of vertices n_d and the highest rank r_d of every degree class d. However, this is hard to achieve, as some degree classes may be small. The crux of the analysis is showing that with $\approx n^{1/3}$ samples, we can approximate n_d and r_d for any degree class that contributes at least a $(1/\log n)$-fraction of the Dasgupta cost.

Recalling that our model assumes degree proportional sampling, the expected number of samples from any degree class B_t is

$$\frac{s}{2m} \sum_{x \in B_t} \deg(x) \approx \frac{s}{m} n_t \cdot t,$$

where s is the total number of samples. Thus, we can estimate n_t and r_t whenever $n_t \cdot t \geq \Omega^* \left(\frac{m}{s}\right)$.

Now, consider the degree class d with the highest degree mass. Since there are at most $\log n/\varphi$ different degree classes, we have $n_d \cdot d \geq \Omega^*(m)$. Thus, we can estimate the contribution to the Dasgupta cost of any degree class B_t which satisfies

$$\frac{n_d \cdot d}{n_t \cdot t} \leq O^*(s).$$

Using the degree class d as a reference, we show that any degree class t that has a significant contribution to the Dasgupta cost, must have a sufficiently large degree mass compared to d.

Specifically, if B_t is a degree class that contributes at least a $(1/\log n)$-fraction of the Dasgupta cost, i.e.

$$\sum_{x \in B_t} \operatorname{rank}(x) \deg(x) \geq \frac{1}{\log n} \sum_{x \in V} \operatorname{rank}(x) \deg(x),$$

then, by Equation (2), we have

$$2t \cdot r_t \cdot n_t \geq \sum_{x \in B_t} \operatorname{rank}(x) \deg(x) \geq \frac{1}{\log n} \sum_{x \in V} \operatorname{rank}(x) \deg(x) \geq \frac{1}{\log n} \sum_{i=1}^{r_t} i \cdot t \geq \frac{1}{\log n} \cdot \frac{r_t^2}{2} \cdot t.$$

From this, we conclude that $n_t \gtrapprox r_t$, allowing us to use the quantity $n_t^2 \cdot t$ as a further proxy for the contribution of B_t to the Dasgupta cost.

Furthermore, we show that if d is our high-degree-mass reference class and t is any degree class that contributes at least a $1/\log n$ fraction of the Dasgupta cost, then $n_t^2 \cdot t \geq n_d^2 \cdot d$. Intuitively, this is because the contribution from B_t is no smaller than the contribution from B_d.

Therefore, the following optimization problem provides an upper bound on the sufficient number of samples.

$$\begin{aligned}
&\max_{t, n_t, d, n_d} \frac{n_d \cdot d}{n_t \cdot t} \\
&\quad \text{such that} \\
&\qquad n_t^2 \cdot t \geq n_d^2 \cdot d && \#B_t \text{ has large contribution to the Dasgupta Cost} \\
&\qquad n_t, n_d \leq n && \#\text{at most } n \text{ vertices} \\
&\qquad n_t, t, d \geq 1 && \#B_t \text{ is non-empty and degrees are non-zero} \\
&\qquad n_d \geq 0.
\end{aligned}$$

However, the above optimization problem is too weak. For example, setting $n_d = n^{1/2}$, $d = n$, $n_t = n$, $t = 1$ gives a feasible solution with value $n^{1/2}$. But this solution would correspond to having $n^{1/2}$ vertices of degree n and n vertices of degree 1, which is impossible in an actual graph. We remedy this by adding an additional constraint that encodes that t, n_t, d, n_d arise from a valid graph.

$$\begin{aligned}
&\max_{t, n_t, d, n_d} \frac{n_d \cdot d}{n_t \cdot t} \\
&\quad \text{such that} \\
&\qquad n_t^2 \cdot t \geq n_d^2 \cdot d && \#B_t \text{ has large contribution to the Dasgupta cost} \\
&\qquad d \leq n_d && \#B_d \text{ does not have too many edges to } V \setminus B_d \\
&\qquad n_t, n_d \leq n && \#\text{at most } n \text{ vertices} \\
&\qquad n_t, t, d \geq 1 && \#B_t \text{ is non-empty and degrees are non-zero} \\
&\qquad n_d \geq 0.
\end{aligned}$$

A priori, there is no reason why the constraint $d \leq n_d$ should be satisfied by our reference class B_d. However, we show that for any graph, it is possible to find a reference class B_d which satisfies $d \leq n_d$ and contributes a large fraction of the degree mass. Intuitively, this is because if all the high-degree-mass classes had $d > n_d$, then they would require too many edges to be routed outside of their degree class, eventually exhausting the available vertices. See proof of Lemma B.4 in Section B.2 for the details.

Finally, we prove that the refined optimization problem has optimal value $\approx n^{1/3}$. Therefore $\approx n^{1/3}$ samples suffice to discover any degree class t with a non-trivial contribution to the Dasgupta cost. The full analysis is presented in Appendix B.

We also show that $\Omega(n^{1/3})$ seeds are necessary to approximate $\sum_{x \in V} \mathrm{rank}(x) \cdot \deg(x)$ to within any constant factor:

Theorem 2.4. *For every positive constant $\alpha > 1$ and n sufficiently large, there exists a pair of expanders G and G' such that $\sum_{i=1}^{n} i \cdot d_i \leq n^2$, $\sum_{i=1}^{n} i \cdot d_i' \geq \alpha n^2$ and at least $\Omega(n^{1/3})$ vertices need to be queried in order to have probability above $2/3$ of distinguishing between them (where $d_1 \geq ... \geq d_n \geq 1$ is the degree sequence in G and $d_1' \geq ... \geq d_n' \geq 1$ is the degree sequence in G').*

Figure 2 below illustrates the graphs G and G' from Theorem 2.4. Graph G has of a set A of $n^{2/3}$ vertices of degree $n^{2/3}$, and the remaining vertices have degree 1. Graph G' has a set A' of $n^{2/3}$ vertices of degree $n^{2/3}$, but the remaining vertices have degree α. In order to distinguish the two graphs, we need to query a vertex outside of A or A', but this requires $\Omega(n^{1/3})$ queries in expectation. The proof of Theorem 2.4 is straightforward, and is included in Section B.4.

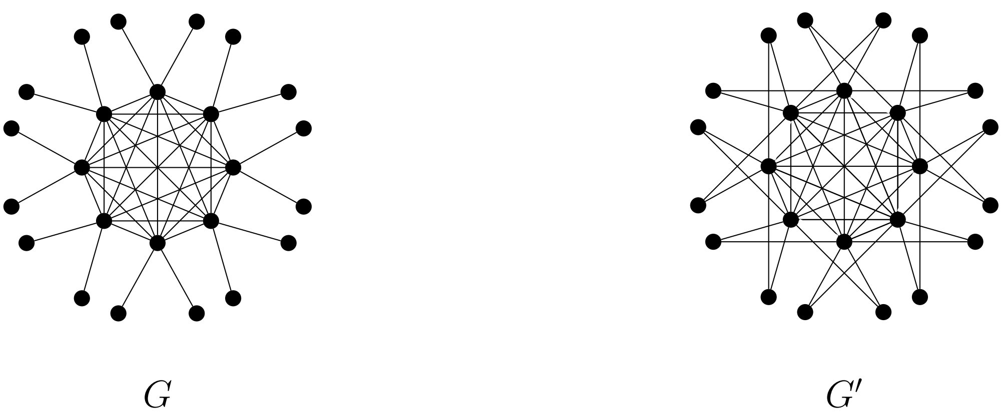

Figure 2: Illustration of the two instances in Theorem 2.4.

Finally, we describe our procedure TotalClustersCost for approximating the total contribution of the clusters to the Dasgupta cost of a d-regular (k, φ, ϵ)-clusterable graph. To approximate the contribution of a single cluster C_i, it suffices to apply the formula $\sum_{x \in C_i} \mathrm{rank}(x) d = \frac{|C_i|(|C_i|+1)}{2} d \approx |C_i|^2 d$. The procedure TotalClustersCost simply sums up these contributions from all the clusters. See Section B.3 for more details.

2.2 Sublinear time access to the contracted graph

In this section, we consider a (k, φ, ϵ)-clusterable graph with bounded maximum degree. For simplicity of presentation, we assume without loss of generality that the graph is d-regular. This is because, by a standard reduction, we can convert a degree d-bounded graph into a d-regular graph by adding self-loops to each vertex.

We denote the Laplacian of G by $\mathcal{L}_G$ and the normalized Laplacian of G by L_G. We will use the following notation for additive-multiplicative approximation.

Definition 5. For $x, y \in \mathbb{R}$, write

$$x \approx_{a,b} y \qquad\qquad \text{if } a^{-1} \cdot y - b \leq x \leq a \cdot y + b.$$

For matrices $X, Y \in \mathbb{R}^{k \times k}$, write

$$X \approx_{a,b} Y \qquad\qquad \text{if } a^{-1} \cdot Y - b \cdot m \cdot I_k \preceq X \preceq a \cdot Y + b \cdot m \cdot I_k.$$

Let $H = G/\mathcal{C}$ be the contraction of G with respect to the underlying clustering $\mathcal{C} = (C_1, C_2, \ldots C_k)$ (Definition 4). We write $H = ([k], \binom{[k]}{2}, W)$ to emphasize that the vertex set of the contracted graph H corresponds to the clusters of G and for $i, j \in [k]$, the pair (i, j) is an edge of H with weight $W(i, j) = |E(C_i, C_j)|$. If we were explicitly given the adjacency/Laplacian matrix of the contracted graph H, then finding a good Dasgupta tree for H can be easily done by using the algorithm of [CC17] which gives a $O(\sqrt{\log k})$ approximation to the optimal tree for H (and an $\sqrt{\log k}/\varphi^{O(1)}$ approximation to the optimal tree for G as shown in Theorem 2.1).

The problem is that we do not have explicit access to the Laplacian of the contracted graph (denoted by $\mathcal{L}_H$). However, to get a good approximation to the Dasgupta Cost of H, it suffices to provide explicit access to a Laplacian $\widetilde{L}$ (which corresponds to a graph $\widetilde{H}$) where cuts in $\widetilde{H}$ approximate sizes of corresponding cuts in H in the following sense: $\exists \alpha > 1, \beta > 0$ and such that for all $S \subseteq [k]$,

$$|\widetilde{E}(S, V \setminus S)| \approx_{\alpha,\beta} |E(S, V \setminus S)|.$$

Motivated by this observation, we simulate this access approximately by constructing a matrix $\widetilde{\mathcal{L}}$ which spectrally approximates $\mathcal{L}_H$ in the sense that

$$\mathcal{L}_H \approx_{a,\xi} \widetilde{\mathcal{L}} \tag{4}$$

in time $\approx m^{1/2+O(\epsilon/\varphi^2)} \cdot \text{poly}(1/\xi)$ for some $0 < \xi < 1 < a$ (see Theorem A.2). So, our immediate goal is to spectrally approximate $\mathcal{L}_H$. We describe this next.

Spectrally approximating $\mathcal{L}_H$: The key insight behind our spectral approximation $\widetilde{\mathcal{L}}$ to $\mathcal{L}_H$ comes from considering the case where our graph is a collection of k disjoint expanders each on n/k vertices. To understand this better, let $L_G = U \Lambda U^T$ denote the eigendecomposition of the Laplacian and let $M = U \Sigma U^T$ denote the eigendecomposition of the lazy random walk matrix. Letting $U_{[k]} \in \mathbb{R}^{n \times k}$ denote a matrix whose columns are the first k columns of U, we will use random sampling to obtain our spectral approximation $\widetilde{\mathcal{L}}$ to the matrix $(I - U_{[k]} \Sigma_{[k]} U_{[k]}^T)$. Indeed, for the instance consisting of k-disjoint equal sized expanders, note that $I - U_{[k]} \Sigma_{[k]} U_{[k]}^T = U_{-[k]} \Sigma_{-[k]} U_{-[k]}^T$ where $U_{[-k]} \in \mathbb{R}^{n \times (n-k)}$ is the matrix whose columns are the last $(n-k)$ columns of U. Using the information that $\lambda_n \geq \lambda_{n-1} \geq \cdots \geq \lambda_{k+1} \geq \varphi^2/2$, one can compare quadratic forms on $I - U_{[k]} \Sigma_{[k]} U_{[k]}^T$ and L_G (the normalized Laplacian of G) to show

$$\widetilde{\mathcal{L}} \approx_{O(1/\varphi^2),\xi} \mathcal{L}_H.$$

We will now describe this in more detail. First, we will show that the matrix $I - U_{[k]} \Sigma_{[k]} U_{[k]}^T$ approximates the quadratic forms of L_G multiplicatively. Then, we describe how this allows us to approximate the quadratic forms of $\mathcal{L}_H$. Finally, we will outline how to approximate the matrix $I - U_{[k]} \Sigma_{[k]} U_{[k]}^T$.

First, we introduce a central definition to this work, which is the notion of spectral embedding.

Definition 6 (k-dimensional spectral embedding). For every vertex x we let $f_x = U_{[k]}^T \mathbb{1}_x$ be the k-dimensional spectral embedding of vertex x.

The spectral embeddings of vertices in a graph provide rich geometric information which has been shown to be useful in graph clustering [LGT14, CPS15, CKK+18, GKL+21a]. The following remark asserts that the inner products between f_x and f_y are well-defined even though the choice for these vectors may not be basis free. First, we need the following standard result on eigenvalues of (k, φ, ϵ)-clusterable graphs [LGT14, CKK+18].

Lemma 1 ([GKL+21a]). Let $G = (V, E)$ be a d-regular graph that admits a (k, φ, ϵ)-clustering. Then we have $\lambda_k \leq 2\epsilon$ and $\lambda_{k+1} \geq \frac{\varphi^2}{2}$.

Remark 2.5. Take a (k, φ, ϵ)-clusterable graph G where ϵ/φ^2 smaller than a constant. Thus, the space spanned by the bottom k eigenvectors of the normalized Laplacian of G is uniquely defined, i.e. the choice of $U_{[k]}$ is unique up to multiplication by an orthonormal matrix $R \in \mathbb{R}^{k \times k}$ on the right. Indeed, by Lemma 1 it holds that $\lambda_k \leq 2\epsilon$ and $\lambda_{k+1} \geq \varphi^2/2$. Thus, since we assume that ϵ/φ^2 is smaller than an absolute constant, we have $2\epsilon < \varphi^2/2$ and thus, the subspace spanned by the bottom k eigenvectors of the Laplacian, i.e. the space of $U_{[k]}$, is uniquely defined, as required. We note that while the choice of f_x for $x \in V$ is not unique, but the dot product between the spectral embedding of $x \in V$ and $y \in V$ is well defined, since for every orthonormal $R \in \mathbb{R}^{k \times k}$ one has

$$\langle Rf_x, Rf_y \rangle = (Rf_x)^T (Rf_y) = (f_x)^T \left(R^T R\right) (f_y) = (f_x)^T \left(f_y\right).$$

Since G is (k, φ, ϵ)-clusterable, by Remark 2.5, the space spanned by the bottom k eigenvectors of the M is uniquely defined. Thus, for any $z \in \mathbb{R}^n$, $z^T (U_{[k]} \Sigma_{[k]} U_{[k]}^T) z$ is well defined.

Having observed this, we will now show that quadratic forms of $I - U_{[k]} \Sigma_{[k]} U_{[k]}^T$ approximate quadratic forms of L_G multiplicatively.

Lemma 2. Suppose that G is d-regular, and let L_G and M denote the normalized Laplacian and lazy random walk matrix of G. Let $M = U \Sigma U^T$ denote the eigendecomposition of M. Then for any vector $z \in \mathbb{R}^n$ with $||z||_2 = 1$ we have

$$\frac{1}{2} \cdot z^T L_G z \leq z^T \left(1 - U_{[k]} \Sigma_{[k]} U_{[k]}^T\right) z \leq \frac{3}{\varphi^2} \cdot z^T L_G z.$$

Proof. Recall that $U_{[k]} \in \mathbb{R}^{n \times k}$ is a matrix whose columns are the first k columns of U, and $\Sigma_{[k]} \in \mathbb{R}^{k \times k}$ is a matrix whose columns are the first k rows and columns of Σ. Let $U_{[-k]} \in \mathbb{R}^{n \times (n-k)}$ be matrix whose columns are the last $n - k$ columns of U, and $\Sigma_{[-k]} \in \mathbb{R}^{(n-k) \times (n-k)}$ be a matrix whose columns are the last $n - k$ rows and columns of Σ. Thus, the eigendecomposition of M is $M = U \Sigma U^T = U_{[k]} \Sigma_{[k]} U_{[k]}^T + U_{[-k]} \Sigma_{[-k]} U_{[-k]}^T$. Note that $M = I - \frac{L_G}{2}$, thus we have

$$z^T (U_{[k]} \Sigma_{[k]} U_{[k]}^T) z + z^T (U_{-[k]} \Sigma_{-[k]} U_{-[k]}^T) z = z^T M z = 1 - \frac{z^T L_G z}{2}, \tag{5}$$

which by rearranging gives

$$\frac{z^T L_G z}{2} \leq 1 - z^T (U_{[k]} \Sigma_{[k]} U_{[k]}^T) z = \frac{z^T L_G z}{2} + z^T (U_{-[k]} \Sigma_{-[k]} U_{-[k]}^T) z. \tag{6}$$

The first inequality gives $1 - z^T (U_{[k]} \Sigma_{[k]} U_{[k]}^T) z \geq \frac{z^T L_G z}{2}$ as desired.

To establish the second inequality above, we will show $z^T(U_{-[k]}\Sigma_{-[k]}U_{-[k]}^T)z \le \frac{2}{\varphi^2} z^T L_G z$. Let $z = \sum_{i=1}^n \alpha_i u_i$ be the eigendecomposition of vector z. Note that

$$z^T L_G z = \sum_{i=1}^n \lambda_i \alpha_i^2 \ge \lambda_{k+1} \sum_{i=k+1}^n \alpha_i^2.$$

By Lemma 1 we have $\lambda_{k+1} \ge \frac{\varphi^2}{2}$. This gives

$$\sum_{i=k+1}^n \alpha_i^2 \le \frac{z^T L_G z}{\lambda_{k+1}} \le \frac{2}{\varphi^2} \cdot z^T L_G z. \tag{7}$$

Finally, putting (6) and (7) together we get

$$1 - z^T(U_{[k]}\Sigma_{[k]}U_{[k]}^T)z = \frac{z^T L_G z}{2} + z^T(U_{-[k]}\Sigma_{-[k]}U_{-[k]}^T)z \le z^T L_G z \left(\frac{1}{2} + \frac{2}{\varphi^2}\right) \le \frac{3}{\varphi^2} \cdot z^T L_G z.$$

□

Now, we apply Lemma 2 to estimate the quadratic form of $\mathcal{L}_H$ on a vector $z \in \mathbb{R}^k$. To that effect, for $z \in \mathbb{R}^k$, we define $z_{\text{ext}} \in \mathbb{R}^n$ as the natural extension of z to $\mathbb{R}^n$: we let $z_{\text{ext}} \in \mathbb{R}^n$ be the vector such that for every $x \in V$, $z_{\text{ext}}(x) = z_i$, where C_i is the cluster that x belongs to.

Note that $z^T \mathcal{L}_H z = z_{\text{ext}}^T \mathcal{L}_G z_{\text{ext}} = d \cdot z_{\text{ext}}^T L_G z_{\text{ext}}$. Thus, to estimate $z^T \mathcal{L}_H z$ it suffices to design a good estimate for $z_{\text{ext}}^T L_G z_{\text{ext}}$, for which we use $z_{\text{ext}}^T(I - U_{[k]}\Sigma_{[k]}U_{[k]}^T)z_{\text{ext}}$, as per Lemma 2.

Finally, we briefly discuss how to estimate the quantity $z_{\text{ext}}^T(I - U_{[k]}\Sigma_{[k]}U_{[k]}^T)z_{\text{ext}}$. We have

$$z_{\text{ext}}^T(I - U_{[k]}\Sigma_{[k]}U_{[k]}^T)z_{\text{ext}} = ||z_{\text{ext}}||_2^2 - z_{\text{ext}}^T U_{[k]}\Sigma_{[k]}U_{[k]}^T z_{\text{ext}} = \sum_{i \in [k]} |C_i| z_i^2 - z_{\text{ext}}^T U_{[k]}\Sigma_{[k]}U_{[k]}^T z_{\text{ext}}.$$

Since the first term on the RHS can be easily approximated in the random sample model, we concentrate on obtaining a good estimate for the second term. We have

$$z_{\text{ext}}^T U_{[k]}\Sigma_{[k]}U_{[k]}^T z_{\text{ext}} = \sum_{i,j \in [k]} z_i z_j \sum_{\substack{x \in C_i \\ y \in C_j}} \left\langle f_x, \Sigma_{[k]} f_y \right\rangle, \tag{8}$$

and therefore in order to estimate $z_{\text{ext}}^T L_G z_{\text{ext}}$, it suffices to use a few random samples to estimate the sum above, as long as one is able to compute high accuracy estimates for $\left\langle f_x, \Sigma_{[k]} f_y \right\rangle, x, y \in V$, with high probability. We refer to such a primitive as a *weighted dot product oracle*, since it computes a weighted dot product between the k-dimensional spectral embeddings f_x and f_y for $x, y \in V$. Assuming such an estimator, which we denote by WeightedDotProductOracle, our algorithm ApproxContractedGraph (Algorithm 4 below) obtains an approximation $\mathcal{L}'$ to the Laplacian of the contracted graph.

Algorithm 4 ApproxContractedGraph$(G, \xi, \mathcal{D})$ time $m^{1/2+O(\epsilon)} \cdot \mathrm{poly}(1/\xi)$

$s \leftarrow O^*\left(m^{O(\epsilon/\varphi^2)} \cdot (1/\xi)^{O(1)}\right)$ *# See Theorem A.2 for the exact value*
$S \leftarrow$ (multi)set of s i.i.d random vertices together with their cluster label
$S_i \leftarrow S \cap C_i$, $\widetilde{w}(i) \leftarrow \frac{|S_i|}{s} \cdot n$, for all $i \in [k]$
for $i, j \in [k]$ **do**
  Assign $K_{i,j} = \frac{\widetilde{w}(i)}{|S_i|} \cdot \frac{\widetilde{w}(j)}{|S_j|} \cdot \sum_{\substack{x \in S_i \\ y \in S_j}} \langle f_x, \Sigma_{[k]} f_y \rangle_{\mathrm{apx}}$
end for
Assign $\mathcal{L}' = d \cdot (I - K)$.
Use SDP to round $\mathcal{L}'$ to a Laplacian $\widetilde{\mathcal{L}}$ s.t $\frac{\varphi^2}{3}\mathcal{L}' - \frac{\xi}{2} \cdot dn \cdot I_k \preceq \widetilde{\mathcal{L}} \preceq 2\mathcal{L}' + \frac{\xi}{2} \cdot dn \cdot I_k$.
$\widetilde{H} \leftarrow \left([k], \binom{[k]}{2}, \widetilde{W}, \widetilde{w}\right)$ *# $\widetilde{H}$ is the weighted graph with Laplacian $\widetilde{\mathcal{L}}$ and vertex weights $\widetilde{w}(i)$*
 # Note that $\widetilde{W}_e = -\widetilde{\mathcal{L}}_e$ for every $e \in \binom{[k]}{2}$
return $\widetilde{H}$

Estimating weighted dot products: Our construction of WeightedDotProductOracle (Algorithm 8) for estimating $\left\langle f_x, \Sigma_{[k]} f_y \right\rangle$ proceeds along the lines of [GKL⁺21a]. We run short random-walks of length $t \approx \log n/\varphi^2$ to obtain dot product access to the spectral embedding of vertices. Given $x \in V$, let m_x denote the probability distribution of endpoints of a t-step random-walks started from x.

We first show that one can estimate $\langle m_x, m_y \rangle$ in time $\approx m^{1/2+O(\epsilon/\varphi^2)} \cdot \mathrm{poly}(1/\xi)$ with probability $1 - n^{-100 \cdot k}$. Then, we construct a Gram matrix $\mathcal{G} \in \mathbb{R}^{s \times s}$ such that $\mathcal{G}_{x,y} = \langle m_x, m_y \rangle$ for every $x, y \in S$, where S is a small set of sampled vertices with $|S| = s = m^{O(\epsilon)}$. Next, we apply an appropriate linear transformation to the Gram matrix $\mathcal{G}$ and use it to estimate $\left\langle f_x, \Sigma_{[k]} f_y \right\rangle$ up to very tiny additive error $\frac{\xi}{n \cdot \mathrm{poly}(k)}$ (see Section C).

Using Semidefinite Programming to round $\mathcal{L}'$: As mentioned above, our proxy for the Laplacian $\mathcal{L}_H$ is obtained via an approximation to $I - U_{[k]}\Sigma_{[k]}U_{[k]}^T$. However, this approximator might not even be a Laplacian. To allay this, we first show that using calls to weighted dot product oracle, we can approximate all the entries of $I - U_{[k]}\Sigma_{[k]}U_{[k]}^T$ to within a very good precision. Starting off from such an approximation, one can use semidefinite programming methods to round the intermediate approximator to a bonafide Laplacian $\widetilde{\mathcal{L}}$. In some more detail, we show the following.

Theorem 2.6 (Informal version of Theorem A.2)**.** *The algorithm ApproxContractedGraph (Algorithm 4) when given a $(k, \Omega(1), \epsilon)$-clusterable graph as input, uses a data structure $\mathcal{D}$ obtained from $\approx m^{1/2+O(\epsilon)}$ time preprocessing routine, runs in time $\approx m^{1/2+O(\epsilon)}$, and finds a graph $\widetilde{H}$ with Laplacian $\widetilde{\mathcal{L}}$ such that with probability $1 - n^{-100}$:*

$$\widetilde{\mathcal{L}} \approx_{O(1/\varphi^2),\xi} \mathcal{L}_H.$$

Approximating the Dasgupta Cost of the contracted graph $\widetilde{H}$: Consider the graph $\widetilde{H} = \left([k], \binom{[k]}{2}, \widetilde{W}, \widetilde{w}\right)$ returned by the ApproxContractedGraph procedure (Algorithm 4).

Once Theorem 2.6 has been established, our estimation primitive EstimatedCost (Algorithm 1) uses a simple vertex-weighted extension of a result of [CC17] to find a tree $\widetilde{T}$ on $\widetilde{H}$.

Specifically, we need the following definitions.

Definition 7 (**Vertex-weighted sparsest cut problem**). Let $H = (V, E, W, w)$ be a vertex and edge weighted graph. For every set $S \in V$, we define the sparsity of cut $(S, V \setminus S)$ on graph H as

$$\mathrm{Sparsity}_H(S) = \frac{W(S, V \setminus S)}{w(S) \cdot w(V \setminus S)},$$

where $w(S) = \sum_{x \in S} w(x)$. The vertex-weighted sparsest cut of graph G is the cut with the minimum sparsity, i.e., $\arg\min_{S \subseteq V} \text{Sparsity}_H(S)$.

Definition 8 (**Vertex-weighted recursive sparsest cut algorithm (WRSC)**). Let $\alpha > 1$ and $H = (V, E, W, w)$ be a vertex and edge weighted graph. Let $(S, V \setminus S)$ be the vertex-weighted sparsest cut of H. The vertex-weighted recursive sparsest cut algorithm on graph H is a recursive algorithm that first finds a cut $(T, V \setminus T)$ such that $\text{Sparsity}_H(T) \leq \alpha \cdot \text{Sparsity}_H(S)$, and then recurs on the subgraph $H[T]$ and subgraph $H[V \setminus T]$.

Next, we first state results which help bound the Dasgupta Cost incurred by the tree one gets by using the *vanilla* recursive sparsest cut algorithm on any graph. Then, in Corollary 2, we present corresponding bounds for vertex-weighted graphs.

Theorem 2.7 (Theorem 2.3 from [CC17])**.** *Let $G = (V, E)$ be a graph. Suppose the RSC algorithm uses an α approximation algorithm for uniform sparsest cut. Then the algorithm RSC achieves an $O(\alpha)$ approximation for the Dasgupta cost of G.*

The following corollary from [CC17], follows using the $O(\sqrt{\log |V|})$ approximation algorithm for the uniform sparsest cut.

Corollary 1 ([CC17]). Let $G = (V, E)$ be a graph. Then algorithm RSC achieves an $O(\sqrt{\log |V|})$ approximation for the Dasgupta cost of G.

Since the clusters of G have different sizes, and since the Dasgupta Cost of a graph is a function of the size of the lowest common ancestor of the endpoints of the edges, we use weighted Dasgupta cost to relate the cost of G and the contracted graph H.

Definition 9 (**Weighted Dasgupta cost**). Let $G = (V, E, W, w)$ denote a vertex and edge weighted graph. For a tree T with $|V|$ leaves (corresponding to vertices of G), we define the weighted Dasgupta cost of T on G as

$$\text{WCOST}_G(T) = \sum_{(x,y) \in E} W(x, y) \cdot \sum_{z \in \text{LEAVES}(T[\text{LCA}(x,y)])} w(z).$$

We get the following guarantee on the Weighted Dasgupta Cost obtained by the WRSC algorithm.

Corollary 2. Let $H = (V, E, W, w)$ be a vertex and edge weighted graph. Then algorithm WRSC achieves an $O(\sqrt{\log |V|})$ approximation for the weighted Dasgupta cost of H.

Letting $\widetilde{T} = \text{WRSC}(\widetilde{H})$ be the tree computed by Algorithm 1, using Corollary 2, we show that the estimate

$$\text{EST} := O\left(\frac{1}{\varphi^2}\right) \cdot \text{WCOST}_{\widetilde{H}}(\widetilde{T}) + \textsc{TotalClustersCost}(G) + O\left(\frac{\xi m n k^2}{\varphi^2}\right)$$

computed by Algorithm 1 satisfies

$$\text{COST}(G) \leq \text{EST} \leq O\left(\frac{\sqrt{\log k}}{\varphi^{O(1)}}\right) \text{COST}(G).$$

The details are presented in Section A.3.

The rest of the paper is structured as follows: In Section A, we prove Theorem 2.1. In Section B we prove the guarantees of the TotalClustersCost procedure. Finally, in Section C, we prove the correctness of the WeightedDotProductOracle.

References

[AB15] Hassan Ashtiani and Shai Ben-David. Representation learning for clustering: A statistical framework. In Marina Meila and Tom Heskes, editors, *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI 2015, July 12-16, 2015, Amsterdam, The Netherlands*, pages 82–91. AUAI Press, 2015.

[ABJ18] Nir Ailon, Anup Bhattacharya, and Ragesh Jaiswal. Approximate correlation clustering using same-cluster queries. In *Proc. of LATIN*, pages 14–27, 2018.

[ABJK18] Nir Ailon, Anup Bhattacharya, Ragesh Jaiswal, and Amit Kumar. Approximate clustering with same-cluster queries. In *Proc. of ITCS*, volume 94, pages 40:1–40:21, 2018.

[ACL+22] Sepehr Assadi, Vaggos Chatziafratis, Jakub Lacki, Vahab Mirrokni, and Chen Wang. Hierarchical clustering in graph streams: Single-pass algorithms and space lower bounds. In *COLT*, volume 178 of *Proceedings of Machine Learning Research*, pages 4643–4702. PMLR, 2022.

[AKB16] Hassan Ashtiani, Shrinu Kushagra, and Shai Ben-David. Clustering with same-cluster queries. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3216–3224, 2016.

[AKLP22] Arpit Agarwal, Sanjeev Khanna, Huan Li, and Prathamesh Patil. Sublinear algorithms for hierarchical clustering. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *NeurIPS*, 2022.

[BBM02] Sugato Basu, Arindam Banerjee, and Raymond Mooney. Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. Citeseer, 2002.

[BCBLP20] Marco Bressan, Nicolò Cesa-Bianchi, Silvio Lattanzi, and Andrea Paudice. Exact recovery of mangled clusters with same-cluster queries. In *Advances in Neural Information Processing Systems*, volume 33, pages 9324–9334, 2020.

[BCLP21] Marco Bressan, Nicolò Cesa-Bianchi, Silvio Lattanzi, and Andrea Paudice. Exact recovery of clusters in finite metric spaces using oracle queries. In *Proc. of COLT*, volume 134, pages 775–803, 2021.

[Ber06] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping Multidimensional Data*, pages 25–71. Springer, 2006.

[CAKMTM18] Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. Hierarchical clustering: Objective functions and algorithms. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 378–397. SIAM, 2018.

[CC17] Moses Charikar and Vaggos Chatziafratis. Approximate hierarchical clustering via sparsest cut and spreading metrics. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 841–854. SIAM, 2017.

[CKK+18] Ashish Chiplunkar, Michael Kapralov, Sanjeev Khanna, Aida Mousavifar, and Yuval Peres. Testing graph clusterability: Algorithms and lower bounds. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 497–508. IEEE, 2018.

[CPS15] Artur Czumaj, Pan Peng, and Christian Sohler. Testing cluster structure of graphs. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 723–732, 2015.

[CS07] Artur Czumaj and Christian Sohler. Testing expansion in bounded-degree graphs. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 570–578. IEEE Computer Society, 2007.

[Das16] Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 118–127, 2016.

[DBE99] Ayhan Demiriz, Kristin P Bennett, and Mark J Embrechts. Semi-supervised clustering using genetic algorithms. *Artificial neural networks in engineering (ANNIE-99)*, pages 809–814, 1999.

[ESBB98] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[EZK18] Ehsan Emamjomeh-Zadeh and David Kempe. Adaptive hierarchical clustering using ordinal queries. In *Proc. of ACM-SIAM SODA*, pages 415–429. SIAM, 2018.

[FGSS18] Donatella Firmani, Sainyam Galhotra, Barna Saha, and Divesh Srivastava. Robust entity resolution using a crowd oracle. *IEEE Data Eng. Bull.*, 41(2):91–103, 2018.

[GKL+21a] Grzegorz Gluch, Michael Kapralov, Silvio Lattanzi, Aida Mousavifar, and Christian Sohler. Spectral clustering oracles in sublinear time. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, 2021.

[GKL+21b] Grzegorz Gluch, Michael Kapralov, Silvio Lattanzi, Aida Mousavifar, and Christian Sohler. Spectral clustering oracles in sublinear time. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, 2021.

[GNK+15] Anja Gruenheid, Besmira Nushi, Tim Kraska, Wolfgang Gatterbauer, and Donald Kossmann. Fault-tolerant entity resolution with the crowd. *CoRR*, abs/1512.00537, 2015.

[GR02] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.

[GR11] Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, pages 68–75. Springer, 2011.

[GSZ+11] Frédéric Gilbert, Paolo Simonetto, Faraz Zaidi, Fabien Jourdan, and Romain Bourqui. Communities and hierarchical structures in dynamic social networks: analysis and visualization. *Social Network Analysis and Mining*, 1(2):83–95, 2011.

[HMMP19] Wasim Huleihel, Arya Mazumdar, Muriel Médard, and Soumyabrata Pal. Same-cluster querying for overlapping clusters. In *Advances in Neural Information Processing Systems 32*, pages 10485–10495, 2019.

[KBDM09] Brian Kulis, Sugato Basu, Inderjit Dhillon, and Raymond Mooney. Semi-supervised graph clustering: a kernel approach. *Machine learning*, 74(1):1–22, 2009.

[KKLM23] Michael Kapralov, Akash Kumar, Silvio Lattanzi, and Aida Mousavifar. Learning hierarchical cluster structure of graphs in sublinear time. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 925–939. SIAM, 2023.

[KS08] Satyen Kale and C. Seshadhri. An expansion tester for bounded degree graphs. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, pages 527–538, 2008.

[LGT14] James R Lee, Shayan Oveis Gharan, and Luca Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):37, 2014.

[MP17] Arya Mazumdar and Soumyabrata Pal. Semisupervised Clustering, AND-Queries and Locally Encodable Source Coding. In *Advances in Neural Information Processing Systems 30*, pages 6489–6499, 2017.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[MS17a] Arya Mazumdar and Barna Saha. Clustering with noisy queries. In *Advances in Neural Information Processing Systems 30*, pages 5788–5799, 2017.

[MS17b] Arya Mazumdar and Barna Saha. Query complexity of clustering with side information. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4682–4693. Curran Associates, Inc., 2017.

[MS21] Bogdan-Adrian Manghiuc and He Sun. Hierarchical clustering: O(1)-approximation for well-clustered graphs. In *NeurIPS*, pages 9278–9289, 2021.

[NS10] Asaf Nachmias and Asaf Shapira. Testing the expansion of a graph. *Inf. Comput.*, 208(4):309–314, 2010.

[Pen20] Pan Peng. Robust clustering oracle and local reconstructor of cluster structure of graphs. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2953–2972. SIAM, 2020.

[SK02] Janne Sinkkonen and Samuel Kaski. Clustering based on conditional distributions in an auxiliary space. *Neural Computation*, 14(1):217–239, 2002.

[SS19] Barna Saha and Sanjay Subramanian. Correlation clustering with same-cluster queries bounded by optimal cost. *CoRR*, abs/1908.04976, 2019.

[VGM15] Vasilis Verroios and Hector Garcia-Molina. Entity resolution with crowd errors. In *Proc. of IEEE ICDE*, pages 219–230, 2015.

[VGMP17] Vasilis Verroios, Hector Garcia-Molina, and Yannis Papakonstantinou. Waldo: An adaptive human interface for crowd entity resolution. In *Proc. of ACM SIGMOD*, pages 1133–1148, 2017.

[VRG19] Fabio Vitale, Anand Rajagopalan, and Claudio Gentile. Flattening a hierarchical clustering through active learning. In *Advances in Neural Information Processing Systems 32*, pages 15263–15273, 2019.

Contents

A Proof of main result (Theorem 2.1)

Throughout this section, assume that $k \geq 2$, $\varphi \in (0, 1)$ and that $\frac{\epsilon}{\varphi^2}$ is smaller than a positive sufficiently small constant. We will always assume that $G = (V, E)$ is a graph that admits a (k, φ, ϵ)-clustering $\mathcal{C} = C_1, \ldots, C_k$ with $\eta \coloneqq \frac{\max_{i\in[k]} |C_i|}{\min_{i\in[k]} |C_i|} = O(1)$.

Given a set $S \subseteq V$, we let $G[S]$ denote the induced subgraph on S, and we use the notation $G\{S\}$ to denote the graph $G[S]$ with self loops added so that every vertex in $G\{S\}$ has the same degree as in G.

Recall that we use O^*-notation to suppress $\text{poly}(k, 1/\varphi, 1/\epsilon)$ and $\text{polylog}\, n$ -factors. For $i \in \mathbb{N}$ we use $[i]$ to denote the set $\{1, 2, \ldots, i\}$. For a vertex $x \in V$, we say that $\mathbb{1}_x \in \mathbb{R}^n$ is the indicator of x, that is, the vector which is 1 at index x and 0 elsewhere. For a (multi)set $S \subseteq V$, we say that $\mathbb{1}_S \in \mathbb{R}^n$ is the indicator of set S, i.e., $\mathbb{1}_S = \sum_{x\in S} \mathbb{1}_x$. For a multiset $I_S = \{x_1, \ldots, x_s\}$ of vertices from V we abuse notation and also denote by S the $n \times s$ matrix whose i^{th} column is $\mathbb{1}_{x_i}$.

Our algorithm and analysis use spectral techniques, and therefore, we setup the following notation. For a symmetric matrix A, we write $\nu_i(A)$ (resp. $\nu_{\max}(A), \nu_{\min}(A)$) to denote the i^{th} largest (resp. maximum, minimum) eigenvalue of A.

We also denote with A_G the adjacency matrix of G and let $\mathcal{L}_G = d \cdot I - A_G$ denote the Laplacian of G. Denote with L_G the *normalized Laplacian* of G where $L_G = I - \frac{A_G}{d}$. We denote the eigenvalues of L_G by $0 \leq \lambda_1 \leq \ldots \leq \lambda_n \leq 2$ and we write Λ to refer to the diagonal matrix of these eigenvalues in non-decreasing order. We also denote by $(u_1, \ldots, u_n)$ an orthonormal basis of eigenvectors of L_G and with $U \in \mathbb{R}^{n \times n}$ the matrix whose columns are the orthonormal eigenvectors of L_G arranged in non-decreasing order of eigenvalues. Therefore the eigendecomposition of L_G is $L_G = U\Lambda U^T$. For any $1 \leq k \leq n$ we write $U_{[k]} \in \mathbb{R}^{n \times k}$ for the matrix whose columns are the first k columns of U. Now, we introduce a central definition to this work, which is the notion of a spectral embedding.

Definition 6 (k-dimensional spectral embedding). For every vertex x we let $f_x = U_{[k]}^T \mathbb{1}_x$ be the k-dimensional spectral embedding of vertex x.

The spectral embeddings of vertices in a graph provide rich geometric information which has been shown to be useful in graph clustering [LGT14, CPS15, CKK+18, GKL+21a].

In this paper, we are interested in the class of graphs that admit a (k, φ, ϵ)-clustering. We would often need to refer to weighted graphs of the form $H = ([k], \binom{[k]}{2}, W, w)$ where W is a weight function on the edges of H and w is a weight function on the vertices of H. We denote the Laplacian of H as $\mathcal{L}_H = D_H - W$ where $D_H \in \mathbb{R}^{k \times k}$ is a diagonal matrix where for every $i \in [k]$, $D_H(i,i) = \sum_{j \in [k]} W(i,j)$. Our algorithms often require to estimate the inner product between spectral embeddings of vertices x and y denoted f_x and f_y. For pairs of vertices $x, y \in V$ we use the notation

$$\langle f_x, f_y \rangle := (f_x)^T (f_y)$$

to denote the dot product in the embedded domain. The following remark asserts that the inner products between f_x and f_y are well-defined even though the choice for these vectors may not be basis free.

First, we need the following standard result on eigenvalues of (k, φ, ϵ)-clusterable graphs [LGT14, CKK+18].

Lemma 1 ([GKL+21a]). Let $G = (V, E)$ be a d-regular graph that admits a (k, φ, ϵ)-clustering. Then we have $\lambda_k \leq 2\epsilon$ and $\lambda_{k+1} \geq \frac{\varphi^2}{2}$.

Now, we are ready to state the remark from before.

Remark 2.5. Take a (k, φ, ϵ)-clusterable graph G where ϵ/φ^2 smaller than a constant. Thus, the space spanned by the bottom k eigenvectors of the normalized Laplacian of G is uniquely defined, i.e. the choice of $U_{[k]}$ is unique up to multiplication by an orthonormal matrix $R \in \mathbb{R}^{k \times k}$ on the right. Indeed, by Lemma 1 it holds that $\lambda_k \leq 2\epsilon$ and $\lambda_{k+1} \geq \varphi^2/2$. Thus, since we assume that ϵ/φ^2 is smaller than an absolute constant, we have $2\epsilon < \varphi^2/2$ and thus, the subspace spanned by the bottom k eigenvectors of the Laplacian, i.e. the space of $U_{[k]}$, is uniquely defined, as required. We note that while the choice of f_x for $x \in V$ is not unique, but the dot product between the spectral embedding of $x \in V$ and $y \in V$ is well defined, since for every orthonormal $R \in \mathbb{R}^{k \times k}$ one has

$$\langle Rf_x, Rf_y \rangle = (Rf_x)^T (Rf_y) = (f_x)^T \left(R^T R\right) (f_y) = (f_x)^T (f_y).$$

We denote the transition matrix of the *random walk associated with* G by $M = \frac{1}{2} \cdot \left(I + \frac{A_G}{d}\right)$. From any vertex v, this random walk takes every edge incident on v with probability $\frac{1}{2d}$, and stays on v with the remaining probability which is at least $\frac{1}{2}$. Note that $M = I - \frac{L_G}{2}$. Observe that for all i, u_i is also an eigenvector of M, with eigenvalue $1 - \frac{\lambda_i}{2}$. We denote with Σ the diagonal matrix of the eigenvalues of M in descending order. Therefore the eigendecomposition of M is $M = U\Sigma U^T$. We write $\Sigma_{[k]} \in \mathbb{R}^{k \times k}$ for the matrix whose columns are the first k rows and columns of Σ. Furthermore, for any t, M^t is a transition matrix of random walks of length t. For any vertex x, we denote the probability distribution of a t-step random walk started from x by $m_x = M^t \mathbb{1}_x$. For a multiset $I_S = \{x_1, \ldots, x_s\}$ of vertices from V, let matrix $M^t S \in \mathbb{R}^{n \times s}$ is a matrix whose column are probability distribution of t-step random walks started from vertices in I_S. Therefore, the i-th column of $M^t S$ is m_{x_i}.

We recall standard (partial) ordering on n-by-n symmetric matrices.

Definition 10. Given two symmetric matrices $A, B \in \mathbb{R}^{n\times n}$, we say that matrix A precedes B in the Loewner (or the PSD) order if $B - A$ is a positive semidefinite matrix. This is denoted as $A \preceq B$ or as $B \succeq A$.

In this section, we present an algorithm to estimate the Dasgupta cost of (k, φ, ϵ)-clusterable graphs in sublinear time assuming oracle access to the underlying clustering, i.e., in the RandomSampleModel , which we formally define below:

Definition 11 (**The RandomSampleModel**). Let $G = (V, E)$ be graph that admits a (k, φ, ϵ)-clustering $C_1, \ldots, C_k$. In the RandomSampleModel, we assume that there exists an oracle that gives us a (multi)set S sampled independently with probability proportional to their degrees, together with their cluster label, i.e., for every vertex $x \in S$, a label $i \in [k]$ such that $C_i \ni x$.

The following definition is central to our algorithm and analysis:

Definition 4 (Contracted graph). Let $G = (V, E)$ be a graph and let $\mathcal{C} = (C_1, \ldots, C_k)$ denote a partition of V into disjoint subsets. We say that the weighted graph $H = \left([k], \binom{[k]}{2}, W, w\right)$ is a contraction of G with respect to the partition $\mathcal{C}$ if for every $i, j \in [k]$ we have $W(i, j) = |E(C_i, C_j)|$, and for every $i \in [k]$ we have $w(i) = |C_i|$. We denote the contraction of G with respect to the partition $\mathcal{C}$ by $H = G/\mathcal{C}$.

In fact, our algorithm for estimating the Dasgupta cost of G will first construct a low cost tree for the contracted graph H and use its cost as a proxy for the Dasgupta cost of G. The algorithm is reproduced in Section A.3 below (see Algorithm 5), and satisfies the following guaratees:

Theorem 2.1. *Let $k \geq 2$, $\varphi \in (0, 1)$ and $\frac{\epsilon}{\varphi^2}$ be a sufficiently small constant. Let $G = (V, E)$ be a bounded degree graph that admits a (k, φ, ϵ)-clustering $C_1, \ldots, C_k$. Let $|V| = n$.*

There exists an algorithm (EstimatedCost(G); Algorithm 1) that w.h.p. estimates the optimum Dasgupta cost of G within an $O\left(\frac{\sqrt{\log k}}{\varphi^{O(1)}}\right)$ factor in time $O^*\left(n^{1/2+O(\epsilon/\varphi^2)} \cdot (d_{\max})^{O(1)}\right)$ using $O^*\left(n^{O(\epsilon/\varphi^2)} \cdot (d_{\max})^{O(1)}\right)$ seed queries.

We start by designing a sublinear time estimator for the quadratic form of L_G, where L_G is the normalized Laplacian of the graph G. Later, we will use this estimator to obtain an approximation of the contracted graph. We can without loss of generality, assume that the graph is d-regular, by adding self-loops to the graph to make all the degrees d. Therefore, we prove the correctness of procedure ApproxContractedGraph (Algorithm 4) for *regular* graphs. Throughout this overview, in Section A.1 and Section A.2 we assume that graph G is d-regular.

Recall that M is the random walk matrix of G and let $M = U\Sigma U^T$ be the eigendecomposition of M. Let $U_{[k]}\Sigma_{[k]}U_{[k]}^T$ be a rank-k approximation to M obtained by truncating the terms corresponding to small eigenvalues. Since, G is (k, φ, ϵ)-clusterable, the space spanned by the bottom k eigenvectors of the M is uniquely defined (see Remark 2.5). Thus, for any $z \in \mathbb{R}^n$, $z^T(U_{[k]}\Sigma_{[k]}U_{[k]}^T)z$ is well defined. Lemma 2 reproduced below shows that the quadratic form of $I - U_{[k]}\Sigma_{[k]}U_{[k]}^T$ approximates the quadratic form of L_G multiplicatively.

Lemma 2. Suppose that G is d-regular, and let L_G and M denote the normalized Laplacian and lazy random walk matrix of G. Let $M = U\Sigma U^T$ denote the eigendecomposition of M. Then for any vector $z \in \mathbb{R}^n$ with $||z||_2 = 1$ we have

$$\frac{1}{2} \cdot z^T L_G z \leq z^T \left(1 - U_{[k]}\Sigma_{[k]}U_{[k]}^T\right) z \leq \frac{3}{\varphi^2} \cdot z^T L_G z.$$

Let G be a d-regular graph that admits a (k,φ,ϵ)-clustering $\mathcal{C} = C_1,\ldots,C_k$. Let $H = G/\mathcal{C}$ be the contraction of G with respect to the partition $\mathcal{C}$ (Definition 4), let $\mathcal{L}_H$ be the Laplacian of graph H. We apply Lemma 2 to estimate the quadratic form of $\mathcal{L}_H$ on a vector $z \in \mathbb{R}^k$. To that effect, for $z \in \mathbb{R}^k$, we define $z_{\text{ext}} \in \mathbb{R}^n$ as the natural extension of z to $\mathbb{R}^n$: we let $z_{\text{ext}} \in \mathbb{R}^n$ be the vector such that for every $x \in V$, $z_{\text{ext}}(x) = z_i$, where C_i is the cluster that x belongs to.

Definition 12 (**Extension z_{ext} of a vector** $z \in \mathbb{R}^k$). For a vector $z \in \mathbb{R}^k$ let $z_{\text{ext}} \in \mathbb{R}^n$ be the vector such that for every $x \in V$, $z_{\text{ext}}(x) = z_i$, where C_i is the cluster that x belongs to.

Note that $z^T\mathcal{L}_H z = z_{\text{ext}}^T \mathcal{L}_G z_{\text{ext}} = d \cdot z_{\text{ext}}^T L_G z_{\text{ext}}$. Thus, to estimate $z^T\mathcal{L}_H z$ it suffices to design a good estimate for $z_{\text{ext}}^T L_G z_{\text{ext}}$, for which we use $z_{\text{ext}}^T(I - U_{[k]}\Sigma_{[k]}U_{[k]}^T)z_{\text{ext}}$, as per Lemma 2:

$$z_{\text{ext}}^T(I - U_{[k]}\Sigma_{[k]}U_{[k]}^T)z_{\text{ext}} = ||z_{\text{ext}}||_2^2 - z_{\text{ext}}^T U_{[k]}\Sigma_{[k]}U_{[k]}^T z_{\text{ext}} = \sum_{i\in[k]} |C_i| z_i^2 - z_{\text{ext}}^T U_{[k]}\Sigma_{[k]}U_{[k]}^T z_{\text{ext}}.$$

Since the first term on the RHS can be easily approximated in the random sample model (Definition 11), we concentrate on obtaining a good estimate for the second term. We have

$$z_{\text{ext}}^T U_{[k]}\Sigma_{[k]}U_{[k]}^T z_{\text{ext}} = \sum_{i,j\in[k]} z_i z_j \sum_{\substack{x\in C_i\\ y\in C_j}} \left\langle f_x, \Sigma_{[k]} f_y \right\rangle, \tag{9}$$

and therefore in order to estimate $z_{\text{ext}}^T L_G z_{\text{ext}}$, it suffices to use few random samples to estimate the sum above, as long as one is able to compute high accuracy estimates for $\left\langle f_x, \Sigma_{[k]} f_y \right\rangle, x, y \in V$, with high probability. We refer to such a primitive as a *weighted dot product oracle*, since it computes a weighted dot product between the k-dimensional spectral embeddings f_x and f_y for $x, y \in V$ (as per Algorithm 7). Assuming such an estimator, which we denote by WeightedDotProductOracle, our algorithm (Algorithm ApproxContractedGraph) obtains an approximation $\widetilde{L} = \widetilde{L}(\widetilde{H})$ to the Laplacian of the contracted graph and is reproduced below.

Algorithm 4 ApproxContractedGraph$(G, \xi, \mathcal{D})$ time $n^{1/2+O(\epsilon)} \cdot \text{poly}(1/\xi)$

$s \leftarrow O^*\left(n^{O(\epsilon/\varphi^2)} \cdot (1/\xi)^{O(1)}\right)$ *# See Theorem A.2 for the exact value*
$S \leftarrow$ (multi)set of s i.i.d random vertices together with their cluster label
$S_i \leftarrow S \cap C_i$, $\widetilde{w}(i) \leftarrow \frac{|S_i|}{s} \cdot n$, for all $i \in [k]$
for $i, j \in [k]$ **do**
  Assign $K_{i,j} = \frac{\widetilde{w}(i)}{|S_i|} \cdot \frac{\widetilde{w}(j)}{|S_j|} \cdot \sum_{\substack{x\in S_i\\ y\in S_j}} \langle f_x, \Sigma_{[k]} f_y\rangle_{\text{apx}}$
end for
Assign $\mathcal{L}' = d \cdot (I - K)$.
Use SDP to round $\mathcal{L}'$ to a Laplacian $\widetilde{\mathcal{L}}$ s.t $\frac{\varphi^2}{3}\mathcal{L}' - \frac{\xi}{2} \cdot dn \cdot I_k \preceq \widetilde{\mathcal{L}} \preceq 2\mathcal{L}' + \frac{\xi}{2} \cdot dn \cdot I_k$.
$\widetilde{H} \leftarrow \left([k], \binom{[k]}{2}, \widetilde{W}, \widetilde{w}\right)$ *# $\widetilde{H}$ is the weighted graph with Laplacian $\widetilde{\mathcal{L}}$ and vertex weights $\widetilde{w}(i)$*
 # Note that $\widetilde{W}_e = -\widetilde{\mathcal{L}}_e$ for every $e \in \binom{[k]}{2}$
return $\widetilde{H}$

Remark A.1. A yet another natural choice for obtaining the estimator $\widetilde{L}$ uses the rank k SVD of L_G which is $U_{[k]}\Lambda_{[k]}U_{[k]}^T$. However, this does not provide a multiplicative approximation to the quadratic form of the Laplacian. To see this, consider the following instance. Let G consist of k disjoint expanders each with inner conductance φ. In this case, $\lambda_1 = \lambda_2 = \cdots = \lambda_k = 0$, and the rank k approximation above results in the estimator being 0.

The following theorem asserts that the procedure APPROXCONTRACTEDGRAPH finds a reasonably good approximation to $\mathcal{L}_H$.

Theorem A.2. *Let $\frac{1}{n^3} < \xi < 1$. Let $H = G/\mathcal{C}$ be the contraction of G with respect to $\mathcal{C}$ (Definition 4), and let $\mathcal{L}_H$ be the Laplacian of H.*

With probability at least $1 - n^{-100}$ over the initialization procedure (Algorithm 7) the following property is satisfied: APPROXCONTRACTEDGRAPH$(G, \xi, \mathcal{D})$ (Algorithm 4), runs in time $O^*\left(n^{1/2+O(\epsilon/\varphi^2)} \cdot \left(\frac{1}{\xi}\right)^{O(1)}\right)$, uses $O^*\left(n^{O(\epsilon/\varphi^2)} \cdot \left(\frac{1}{\xi}\right)^{O(1)}\right)$ seed queries, and finds a graph $\widetilde{H}$ with Laplacian $\widetilde{\mathcal{L}}$ such that with probability at least $1 - n^{-100}$, we have

$$\widetilde{\mathcal{L}} \approx_{O(1/\varphi^2),\xi} \mathcal{L}_H.$$

Once Theorem A.2 has been established, our algorithm for approximating Dasgupta cost of G simply uses the algorithm of [CC17] on the approximation $\widetilde{H}$ to the contracted graph H. Since the cluster sizes in G are different and the Dasgupta cost of a graph is a function of the size of the lowest common ancestor of endpoints of edges, to relate the cost of G and the contracted graph H, we recall the definition of weighted Dasgupta cost.

Definition 9 (**Weighted Dasgupta cost**). Let $G = (V, E, W, w)$ denote a vertex and edge weighted graph. For a tree T with $|V|$ leaves (corresponding to vertices of G), we define the weighted Dasgupta cost of T on G as

$$\mathrm{WCOST}_G(T) = \sum_{(x,y)\in E} W(x,y) \cdot \sum_{z\in \textsc{leaves}(T[\mathrm{LCA}(x,y)])} w(z).$$

Let $G = (V, E)$ be a d-regular graph that admits a (k, φ, ϵ)-clustering $\mathcal{C} = C_1, \ldots, C_k$, and let $H = G/\mathcal{C}$ be the contraction of G with respect to the partition $\mathcal{C}$. We denote the optimal Dasgupta tree for H as T_H^*. With this setup, we will now show the main result (Theorem 2.1) which is finally presented in Section A.3. We consider the graph $\widetilde{H} = \left([k], \binom{[k]}{2}, \widetilde{W}, \widetilde{w}\right)$ returned by the APPROXCONTRACTEDGRAPH procedure. Our estimation primitive ESTIMATEDCOST (Algorithm 5) uses a simple vertex-weighted extension of a result of [CC17] to find a tree $\widetilde{T}$ on $\widetilde{H}$. Specifically, we need the following definitions.

Definition 7 (**Vertex-weighted sparsest cut problem**). Let $H = (V, E, W, w)$ be a vertex and edge weighted graph. For every set $S \in V$, we define the sparsity of cut $(S, V \setminus S)$ on graph H as

$$\mathrm{Sparsity}_H(S) = \frac{W(S, V \setminus S)}{w(S) \cdot w(V \setminus S)},$$

where $w(S) = \sum_{x\in S} w(x)$. The vertex-weighted sparsest cut of graph G is the cut with the minimum sparsity, i.e., $\arg\min_{S\subseteq V} \mathrm{Sparsity}_H(S)$.

Definition 8 (**Vertex-weighted recursive sparsest cut algorithm (WRSC)**). Let $\alpha > 1$ and $H = (V, E, W, w)$ be a vertex and edge weighted graph. Let $(S, V \setminus S)$ be the vertex-weighted sparsest cut of H. The vertex-weighted recursive sparsest cut algorithm on graph H is a recursive algorithm that first finds a cut $(T, V \setminus T)$ such that $\mathrm{Sparsity}_H(T) \leq \alpha \cdot \mathrm{Sparsity}_H(S)$, and then recurs on the subgraph $H[T]$ and subgraph $H[V \setminus T]$.

Next, we first state results which help bound the Dasgupta Cost incurred by the tree one gets by using *vanilla* recursive sparsest cut algorithm on any graph. In Corollary 2, we present algorithms which present bounds on the Dasgupta Cost one obtains for vertex weighted graphs.

Theorem 2.7 (Theorem 2.3 from [CC17]). *Let $G = (V, E)$ be a graph. Suppose the RSC algorithm uses an α approximation algorithm for uniform sparsest cut. Then the algorithm RSC achieves an $O(\alpha)$ approximation for the Dasgupta cost of G.*

The following corollary from [CC17], follows using the $O(\sqrt{\log |V|})$ approximation algorithm for the uniform sparsest cut.

Corollary 1 ([CC17]). Let $G = (V, E)$ be a graph. Then algorithm RSC achieves an $O(\sqrt{\log |V|})$ approximation for the Dasgupta cost of G.

Corollary 2. Let $H = (V, E, W, w)$ be a vertex and edge weighted graph. Then algorithm WRSC achieves an $O(\sqrt{\log |V|})$ approximation for the weighted Dasgupta cost of H.

Let $\widetilde{T} = \mathrm{WRSC}(\widetilde{H})$. We denote the cost of $\widetilde{T}$ as $\mathrm{WCOST}_{\widetilde{H}}(\widetilde{T})$, we present an estimator (EST) for the Dasgupta cost of an optimal tree in G. In particular, we show $\mathrm{EST} \leq O\left(\frac{\sqrt{\log k}}{\varphi^{O(1)}}\right) \cdot \mathrm{COST}_G(T^*_G)$, where,

$$\mathrm{EST} := O\left(\frac{1}{\varphi^2}\right) \cdot \mathrm{WCOST}_{\widetilde{H}}(\widetilde{T}) + \textsc{TotalClustersCost}(G) + O\left(\frac{\xi m n k^2}{\varphi^2}\right).$$

The proof proceeds in two steps: in the first step, we prove Lemma 8 which upper bounds EST in terms of $\mathrm{WCOST}_H(T^*_H)$, where T^*_H is the optimum Dasgupta tree for H. Next, we use Lemma 12 to relate $\mathrm{WCOST}_H(T^*_H)$ with $\mathrm{COST}_G(T^*_G)$. We finally restate the $\textsc{EstimatedCost}$ procedure.

Algorithm 5 $\textsc{EstimatedCost}(G)$ time $\approx n^{1/2+O(\epsilon)}$

$\xi \leftarrow O\left(\frac{\varphi^2}{d_{\max} \cdot k^3 \cdot \sqrt{\log k}}\right)$

$\mathcal{D} \leftarrow \textsc{InitializeWeightedDotProductOracle}(G, \xi)$ *# See Algorithm 7*

$\widetilde{H} \leftarrow \textsc{ApproxContractedGraph}(G, \xi, \mathcal{D})$ *#The Laplacian $\widetilde{\mathcal{L}}$ of $\widetilde{H}$ satisfies Equation* (4)

$\widetilde{T} \leftarrow \textsc{WeightedRecursiveSparsestCut}(\widetilde{H})$ *#Weighted version of Algorithm of [CC17]*

$\mathrm{EST} \leftarrow O\left(\frac{1}{\varphi^2}\right) \cdot \mathrm{WCOST}_{\widetilde{H}}(\widetilde{T}) + \textsc{TotalClustersCost}(G) + O\left(\frac{\xi m n k^2}{\varphi^2}\right)$

return EST

The details are presented in Section A.3. The rest of this section is organized as follows. We first present the analysis of $\textsc{ApproxContractedGraph}$ in Section A.1. In turn, this analysis requires guarantees on how well the matrix $\mathcal{L}'$ obtained in Line 7 of Algorithm 4 approximates $\mathcal{L}_H$. This guarantee is presented in Section A.2. Finally, the proof of Theorem 2.1 is given in Section A.3. In Section A.4 we prove a few intermediate Lemmas used in the proof.

A.1 Correctness of $\textsc{ApproxContractedGraph}$ (Proof of Theorem A.2)

Throughout this section, we will assume that $G = (V, E)$ is a d-regular graph. The main result of this section is the correctness of $\textsc{ApproxContractedGraph}$ in the random sample (Theorem A.2).

We first collect the ingredients we need before proceeding further with the proof. As a first step, since $\textsc{ApproxContractedGraph}$ relies on our $\textsc{WeightedDotProductOracle}$, we will need correctness guarantees for the latter in our analysis. These are presented in Theorem A.3 below, and proved in Appendix C.

Theorem A.3. *Let M denote the random walk matrix of G, and let $M = U\Sigma U^T$ denote the eigendecomposition of M. With probability at least $1 - n^{-100}$ over the initialization procedure (Algorithm 7) the following holds:*

With probability at least $1 - 3 \cdot n^{-100 \cdot k}$ for all $x, y \in V$ we have

$$|\left\langle f_x, \Sigma_{[k]} f_y \right\rangle_{apx} - \left\langle f_x, \Sigma_{[k]} f_y \right\rangle| \leq \frac{\xi}{nk^2},$$

where $\left\langle f_x, \Sigma_{[k]} f_y \right\rangle_{apx} =$ WEIGHTEDDOTPRODUCTORACLE$(G, x, y, \xi, \mathcal{D})$.

Moreover, the running time of the procedures INITIALIZEWEIGHTEDDOTPRODUCTORACLE (Algorithm 7) and WEIGHTEDDOTPRODUCTORACLE (Algorithm 8) is $O^*\left(n^{1/2+O(\epsilon/\varphi^2)} \cdot \left(\frac{1}{\xi}\right)^{O(1)}\right)$.

The other ingredient in our proof comprises of the following two lemmas we state below.

Lemma 3. Let $\frac{1}{n^4} < \xi < 1$. Let $H = G/\mathcal{C}$ be the contraction of G with respect to $\mathcal{C}$ (Definition 4), and let $\mathcal{L}_H$ be the Laplacian of H. Let $K' \in \mathbb{R}^{k \times k}$ denote the matrix where $K'_{i,j} = \mathbb{1}_{C_i} U_{[k]}^T \Sigma_{[k]} U_{[k]} \mathbb{1}_{C_j}$. Then with probability at least $1 - n^{-50 \cdot k}$ we have

$$K' - \xi n/k \cdot I_k \preceq K \preceq K' + \xi n/k \cdot I_k.$$

The proof of Lemma 3 is deferred to Section A.2.

Lemma 4. Let $\frac{1}{n^4} < \xi < 1$. Let $H = G/\mathcal{C}$ be the contraction of G with respect to $\mathcal{C}$ (Definition 4), and let $\mathcal{L}_H$ be the Laplacian of H. Let $K' \in \mathbb{R}^{k \times k}$ denote the matrix where $K'_{i,j} = \mathbb{1}_{C_i} U_{[k]}^T \Sigma_{[k]} U_{[k]} \mathbb{1}_{C_j}$. Then

$$\mathcal{L}_H/2 \preceq d(I - K') \preceq 3/\varphi^2 \cdot \mathcal{L}_H.$$

Proof. Follows immediately from Lemma 2. □

We now prove Theorem A.2.

Proof. (Of Theorem A.2) Let $\mathcal{L}'$ denote the matrix obtained in Line 7 of Algorithm 4. We use the following SDP to round $\mathcal{L}'$ to a Laplacian $\widetilde{\mathcal{L}}$ in Line 8 of Algorithm 4:

(C1). Minimize 0 (Feasibility program, no objective)

(C2). $\widetilde{\mathcal{L}}$ symmetric and $\widetilde{\mathcal{L}} \succeq 0$

(C3). For every $i \in [k]$, $\sum_j \widetilde{\mathcal{L}}_{i,j} = 0$, and $\widetilde{\mathcal{L}}_{i,i} \geq 0$.

(C4). For every $i \neq j \in [k]$, $\widetilde{\mathcal{L}}_{i,j} \leq 0$

(C5). $\frac{\varphi^2}{3}\mathcal{L}' - \frac{\varphi^2}{3} \cdot \frac{\xi dn}{k} \cdot I_k \preceq \widetilde{\mathcal{L}} \preceq 2\mathcal{L}' + \frac{2\xi dn}{k} \cdot I_k$.

First, we will show that this program is feasible by showing that $\mathcal{L}_H$ is a feasible solution. Since $\mathcal{L}_H$ is a graph Laplacian, it satisfies constraints (C1), (C2), (C3) and (C4). Now, consider constraint (C5).

Using Lemma 3 and Lemma 4, we see that, except with probability at most n^{-100}, the following both hold for the matrix K defined in Line 5 of Algorithm 4.

- $K' - \xi n/k \cdot I_k \preceq K \preceq K' + \xi n/k \cdot I_k$, and
- $\mathcal{L}_H/2 \preceq d(I - K') \preceq 3/\varphi^2 \cdot \mathcal{L}_H$.

where K' is a k-by-k matrix defined as $K'_{i,j} = \mathbb{1}_{C_i} U_{[k]}^T \Sigma_{[k]} U_{[k]} \mathbb{1}_{C_j}$.

Let us assume that both of these conditions hold. Conditioned on this, the following is seen to hold

$$\frac{\varphi^2}{3} \cdot d(I-K) - \frac{\varphi^2}{3} \frac{\xi dn}{k} \cdot I_k \preceq \mathcal{L}_H \preceq 2 \cdot d(I-K) + 2\frac{\xi dn}{k} \cdot I_k. \tag{10}$$

This is precisely constraint (C5), which means that $\mathcal{L}_H$ indeed satisfies the constraints of the SDP and thus $\mathcal{L}_H$ is a feasible solution. Further, the set of feasible solutions contains an open ball: in particular all Laplacian matrices $\mathcal{L}'$ with $\|\mathcal{L}' - \mathcal{L}\|_F^2 \leq n^{-10}$ lie inside this ball. Therefore, in additional time $k^{O(1)} \cdot \log n$, the Ellipsoid Algorithm returns some feasible solution $\widetilde{\mathcal{L}}$. $\widetilde{\mathcal{L}}$ satisfies all these constraints as well. Writing $\mathcal{L}' = d(I-K)$, this means

$$\frac{\varphi^2}{3}\mathcal{L}' - \frac{\varphi^2}{3} \cdot \frac{\xi dn}{k} \cdot I_k \preceq \widetilde{\mathcal{L}} \preceq 2\mathcal{L}' + \frac{2\xi dn}{k} \cdot I_k. \tag{11}$$

Therfore, By (10), we have

$$\frac{\varphi^2}{6} \cdot \mathcal{L}_H - \varphi^2 \cdot \frac{\xi dn}{k} \cdot I_k \preceq \widetilde{\mathcal{L}} \preceq \frac{6}{\varphi^2} \cdot \mathcal{L}_H + \frac{6}{\varphi^2} \cdot \frac{\xi dn}{k} \cdot I_k.$$

Running time: The overall running time is dominated by the call to QuadraticOracle which finds the matrix K and thus it is at most $O^*\left(n^{1/2+O(\epsilon/\varphi^2)} \cdot \left(\frac{1}{\xi}\right)^{O(1)}\right)$.

Finally, we bound the number of seed queries issued. The algorithm estimates the cluster sizes $\widetilde{w}(i)$ within a multiplicative $(1 \pm \delta)$ factor with $\delta = \frac{\xi}{512 \cdot k^2 \cdot n^{40 \cdot \epsilon/\varphi^2}}$. By simple Chernoff bounds, this can be done using

$$s = \frac{400 \log n \cdot k^2}{\delta^2} \leq \frac{10^9 \cdot n^{80\epsilon/\varphi^2} \cdot k^6 \cdot \log n}{\xi^2}$$

seeds (see Lemma 7).

□

A.2 Proof of an intermediate Lemma used in Theorem A.2 (Lemma 3)

To finish the proof of Theorem A.2, we now prove Lemma 3. Throughout this section, we assume that $G = (V, E)$ is a d-regular graph.

First, we will need Lemma 5, which proves that for any pair of large enough subsets $A, B \subseteq V$ we can estimate the means of the weighted inner product between the spectral embedding of vertices in A and B i.e., $\frac{1}{|A| \cdot |B|} \sum_{\substack{x \in A \\ y \in B}} \langle f_x, \Sigma_{[k]} f_y \rangle$, by taking enough random samples from $S_A \subseteq A$ and $S_B \subseteq B$ and estimating the (weighted) inner product empirically. We prove Lemma 5 in Appendix A.4).

Lemma 5. Let $A, B \subseteq V$. Let $S_A \subseteq A$ and $S_B \subseteq B$ denote (multi)sets of vertices sampled independently and uniformly at random from A and B respectively, where $|S_A|, |S_B| \geq \frac{1600 \cdot k^3 \cdot n^{40\epsilon/\varphi^2} \cdot \log n}{\xi^2}$. Let M denote the lazy random walk matrix of G, and $M = U \Sigma U^T$ be the eigendecomposition of M. Then, with probability at least $1 - n^{-100 \cdot k}$ we have

$$\left| \mathbb{1}_A^T \cdot (U_{[k]} \Sigma_{[k]} U_{[k]}^T)\ \mathbb{1}_B\ - \frac{|A| \cdot |B|}{|S_A| \cdot |S_B|} \cdot \mathbb{1}_{S_A}^T (U_{[k]} \Sigma_{[k]} U_{[k]}^T) \mathbb{1}_{S_B} \right| \leq \xi \cdot n \tag{12}$$

Next, to prove the correctness of quadratic oracle we need the following lemma that bounds the ℓ_∞-norm on any unit vector in the eigenspace spanned by the bottom k eigenvectors of L_G, i.e. $U_{[k]}$ proved by [GKL+21a].

Lemma 6 ([GKL+21a]). Let $\varphi \in (0,1)$ and $\epsilon \leq \frac{\varphi^2}{100}$, and let $G=(V,E)$ be a d-regular graph that admits (k,φ,ϵ)-clustering $C_1,\ldots,C_k$. Let u be a normalized eigenvector of L with $||u||_2 = 1$ and with eigenvalue at most 2ϵ. Then we have

$$||u||_\infty \leq n^{20\cdot\epsilon/\varphi^2} \cdot \sqrt{\frac{160}{\min_{i\in[k]}|C_i|}}.$$

Finally, we need the following lemma which shows that Algorithm 4 receives enough samples from every cluster in the RandomSampleModel. Lemma 7 proves this fact. The proof relies on a simple Chernoff bound application and we defer it to Appendix A.4.

Lemma 7. Let $S \subseteq V$ denote a set of random vertices returned by the RandomSampleModel (Definition 11) in a regular graph. For every $i \in [k]$ let $S_i = S \cap C_i$. If $|S| \geq \frac{400\cdot\log n\cdot k^2}{\delta^2}$, then with probability at least $1-n^{-100\cdot k}$ for every $i\in[k]$ we have $|S_i| \in (1\pm\delta)\cdot|S|\cdot\frac{|C_i|}{n}$.

We are now ready to prove Lemma 3

Proof. (Of Lemma 3) We will first bound the entrywise difference between the matrices K and K'. In particular, we first show that for a fixed entry indexed by $i,j\in[k]$, we have $|K_{i,j}-K'_{i,j}| \leq \frac{\xi n}{2k^2}$. We will prove this via a repeated use of triangle inequality. For the reader's simplicity, we recall

- $K_{i,j} = \frac{\widetilde{w}(i)}{|S_i|}\cdot\frac{\widetilde{w}(j)}{|S_j|}\cdot\sum_{\substack{x\in S_i\\ y\in S_j}}\langle f_x, \Sigma_{[k]} f_y\rangle_{\text{apx}}$.
- $K'_{i,j} = \mathbb{1}_{C_i} U_{[k]}^T \Sigma_{[k]} U_{[k]} \mathbb{1}_{C_j}$

Now, we setup the stage to use triangle inequality as mentioned earlier. To this end, we introduce the following auxiliary quantities

- $\alpha_{i,j} = \frac{\widetilde{w}(i)}{|S_i|}\cdot\frac{|\widetilde{w}(j)}{|S_j|}\cdot\langle\sum_{x\in S_i} f_x, \Sigma_{[k]}\sum_{y\in S_j} f_y\rangle$.
- $\beta_{i,j} = \frac{|C_i|}{|S_i|}\cdot\frac{|C_j|}{|S_j|}\cdot\langle\sum_{x\in S_i} f_x, \Sigma_{[k]}\sum_{y\in S_j} f_y\rangle$

We know

$$|K_{i,j}-K'_{i,j}| \leq |K_{i,j}-\alpha_{i,j}| + |\alpha_{i,j}-\beta_{i,j}| + |\beta_{i,j}-K'_{i,j}| \tag{13}$$

We bound all the terms above. Towards the first term note that

$$\begin{aligned}
|K_{i,j}-\alpha_{i,j}| &\leq \frac{\widetilde{w}(i)}{|S_i|}\cdot\frac{\widetilde{w}(j)}{|S_j|}\cdot\left|\sum_{\substack{x\in S_i\\ y\in S_j}}\Big(\langle f_x, \Sigma_{[k]} f_y\rangle_{apx} - \langle f_x, \Sigma_{k]} f_y\rangle\Big)\right| \\
&\leq \frac{\widetilde{w}(i)}{|S_i|}\cdot\frac{\widetilde{w}(j)}{|S_j|}\cdot\frac{\xi}{nk^2} && \text{By Theorem A.3} \\
&\leq \frac{\xi n}{16k^4} && \text{By choice of } s.
\end{aligned}$$

The second line above holds with probability at least $1 - 3n^{-100k}$ for any fixed pair of vertices. By a union bound over all pairs of sampled vertices, it holds with probability at least $1 - n^{-50k}$.

Next, we bound the second term in Equation (13).

$$
\begin{aligned}
|\alpha_{i,j} - \beta_{i,j}| &\leq |\langle \mathbb{1}_{S_i}, \mathbb{1}_{S_j} \rangle| \left| \left(\frac{\widetilde{w}(i) \cdot \widetilde{w}(j)}{|S_i||S_j|} - \frac{|C_i||C_j|}{|S_i||S_j|} \right) \right| \\
&\leq \big| |C_i| \cdot |C_j| - (1-\delta)^2 \cdot |C_i| \cdot |C_j| \big| \cdot \max_{x \in V} ||f_x||_2^2 && \text{As } ||\Sigma_{[k]}||_2 = \max_{i \in [k]} \left(1 - \frac{\lambda_i}{2}\right) \leq 1, \text{ and Lemma 7} \\
&\leq 4 \cdot \delta \cdot |C_i| \cdot |C_j| \cdot \sum_{i=1}^{k} ||u_i||_\infty^2 && \text{As } ||f_x||_2^2 = \sum_{i=1}^{k} u_i^2(x) \\
&\leq 4 \cdot \delta \cdot |C_i| \cdot |C_j| \cdot k \cdot \frac{160 \cdot n^{40 \cdot \epsilon/\varphi^2}}{\min_{t \in [k]} |C_t|} && \text{By Lemma 6} \\
&\leq 320 \cdot \delta \cdot n^{40 \cdot \epsilon/\varphi^2} \cdot n \\
&\leq \xi n / 16k^2 && \text{As } \delta = \frac{\xi}{512 \cdot k^2 \cdot n^{40 \cdot \epsilon/\varphi^2}}
\end{aligned}
$$

Finally, we consider the last term in Equation (13). Here, we seek to bound $|\beta_{i,j} - K'_{i,j}|$. Since $|S_i| \geq \frac{1600k^7 n^{40\epsilon/\varphi^2} \cdot \log n}{\xi^2}$ for every i, it follows from Lemma 5 that $|\beta_{i,j} - K'_{i,j}| \leq \xi n/16k^2$.

Overall, by an application of triangle inequality, this means that in every single entry, we have $|K_{i,j} - K'_{i,j}| = err(i,j) \leq \xi n/5k^2$. Let $E \in \mathbb{R}^{k \times k}$ denote the matrix whose (i,j)-th entry is $err(i,j)$. Observe that the matrix $\xi n/k \cdot I_k - E$ is symmetric and diagonally dominant and therefore, in the psd order we have $E \preceq \xi n/k \cdot I_k$ which implies the lemma.

□

A.3 Correctness of EstimatedCost (Proof of Theorem 2.1)

Let $G = (V, E)$ be a d-regular graph that admits a (k, φ, ϵ)-clustering $\mathcal{C} = C_1, \ldots, C_k$, and let $H = G/\mathcal{C}$ be the of contraction of G with respect to the partition $\mathcal{C}$. We denote the optimal Dasgupta tree for H as T_H^*. With this setup, we will now show the main result (Theorem 2.1). Our estimation primitive EstimatedCost (Algorithm 5) uses a simple vertex-weighted extension of a result of [CC17] to find a tree $\widetilde{T}$ on $\widetilde{H}$. Let $\widetilde{T} = \text{WRSC}(\widetilde{H})$ (See Definition 8 and Corollary 2). We denote the cost of $\widetilde{T}$ as $\text{WCOST}_{\widetilde{H}}(\widetilde{T})$, we present an estimator (EST) for the Dasgupta cost of an optimal tree in G. In particular, we show $\text{EST} \leq O\left(\frac{\sqrt{\log k}}{\varphi^{O(1)}}\right) \cdot \text{COST}_G(T_G^*)$, where,

$$
\text{EST} := O\left(\frac{1}{\varphi^2}\right) \cdot \text{WCOST}_{\widetilde{H}}(\widetilde{T}) + \text{TotalClustersCost}(G) + O\left(\frac{\xi m n k^2}{\varphi^2}\right)
$$

The proof proceeds in two steps: in the first step, we prove Lemma 8 which upper bounds EST in terms of $\text{WCOST}_H(T_H^*)$, where T_H^* is the optimum Dasgupta tree for H. Next, we use Lemma 12 to relate $\text{WCOST}_H(T_H^*)$ with $\text{COST}_G(T_G^*)$.

A.3.1 Estimating the cost of the contracted graph

The main result of this section is Lemma 8. This lemma bounds $\text{WCOST}_{\widetilde{H}}(\widetilde{T})$ in terms of $\text{WCOST}_H(T_H^*)$ where T_H^* is the optimum Dasgupta tree for H. This is useful in relating EST with $\text{WCOST}_H(T_H^*)$.

Lemma 8. Let $\frac{1}{n^4} < \xi < 1$, $\xi' = \xi/16$. Let $H = G/\mathcal{C}$ be the contraction of G with respect to the partition $\mathcal{C}$ (Definition 4) and let T^*_H denote an optimum weighted Dasgupta tree for H. Let $\widetilde{H} = \left([k], \binom{[k]}{2}, \widetilde{W}, \widetilde{w}, \right)$ be the graph obtained by $\textsc{ApproxContractedGraph}(G, \xi', \mathcal{D})$ (Algorithm 4). Let $\widetilde{T} = \mathrm{WRSC}(\widetilde{H})$ denote a hierarchical clustering tree constructed on the graph $\widetilde{H}$ using the recursive sparsest cut algorithm. Then with probability at least $1 - 2 \cdot n^{-100}$ we have

$$\Omega\left(\varphi^2\right) \cdot \mathrm{WCOST}_H(T^*_H) - \xi mnk^2 \leq \mathrm{WCOST}_{\widetilde{H}}(\widetilde{T}) \leq O\left(\frac{\sqrt{\log k}}{\varphi^2} \cdot \mathrm{WCOST}_H(T^*_H) + \xi mnk^2\sqrt{\log k}\right)$$

To prove Lemma 8 we first present Definition 13 and Lemma 9 which is a variation of Claim 2.1. from [CC17].

Definition 13 (**Maximal clusters induced by a tree**). Let $H = (V, E, W_H, w_H)$ be a vertex and edge-weighted graph. Let T be a tree with $|V|$ leaves (corresponding to the vertices of H). For any node u of the tree T, we define the weight of the node u as the sum of the weight of those vertices of H that are leaves of the subtree $T[u]$:

$$w_T(u) = \sum_{x \in \textsc{leaves}(T[u])} w_H(x).$$

Let $T(s)$ be the maximal nodes in T such that their weight is at most s:

$$T(s) = \{u \in T : w_T(u) \leq s\}.$$

We refer to these nodes as maximal clusters of weight at most s. For convenience, we also define $T(s) = \textsc{leaves}(T)$ for every $s < \max_{x \in V} w(x)$. Note that $T(s)$ is a partition of V.

We denote by $E_T(s)$ the edges that are cut in $T(s)$, i.e. edges with end points in different clusters in $T(s)$. For convenience, we also define $E_T(s) = E$ for every $s < \max_{x \in V} w(x)$. Also, we let $W_{T(s)} = \sum_{(x,y) \in E_T(s)} W(x, y)$ denote the total weight of the cut edges in $T(s)$.

Lemma 9. Let $H = (V, E, W_H, w_H)$ be a vertex and edge-weighted graph. Let $\ell = \sum_{x \in V} w_H(x)$ be the total vertex weight of graph H. Let T be a tree with $|V|$ leaves (corresponding to the vertices of H). Then we have

$$\mathrm{WCOST}_H(T) = \sum_{s=0}^{\ell} W_T(s).$$

Proof. The proof is identical to the proof given in [CC17]. Consider any edge $(x, y) \in E$. Let $r = w_T(\mathrm{LCA}(x, y))$ be the weight of the lowest common ancestor of x, y. Then, by Definition 9 the contribution of the edge (x, y) to the LHS is $r \cdot W(x, y)$. Also, note that $(x, y) \in E_T(s)$ for all $0 \leq s \leq r - 1$. Hence, the contribution of the edge (x, y) to RHS is also $r \cdot W(x, y)$. $\square$

Next, we present Lemma 10 which shows that the Dasgupta cost of two graphs is close if the weight of every cut is similar in both graphs.

Lemma 10. Let $H = (V, E, W, w)$ and $H' = (V, E', W', w)$ be vertex and edge-weighted graphs. Let $\alpha, \beta > 0$. Suppose that for every set $S \subseteq V$ we have $W'(S, V \setminus S) \leq \beta \cdot W(S, V \setminus S) + \alpha$. Let T and T' denote the optimum Dasgupta tree of H and H' respectively. Then we have

$$\mathrm{WCOST}_{H'}(T') \leq \beta \cdot \mathrm{WCOST}_H(T) + \frac{\alpha}{2} \cdot |V| \cdot ||w||_1.$$

Proof. The main idea in the proof is to use Lemma 9 to relate the cost of the tree to the cost of cuts. Let $\ell = ||w||_1 = \sum_{x \in V} w(x)$. For every $0 \le s \le \ell$, let $B_1, B_2, \cdots, B_{t_s}$ denote the partition of maximal clusters of weight at most s induced by T in graph H (Definition 13). By Lemma 9 we have

$$\mathrm{WCOST}_H(T) = \sum_{s=0}^{\ell} W_{T(s)} = \frac{1}{2} \cdot \sum_{s=0}^{\ell} \sum_{i \in [t_s]} W(B_i, V \setminus \overline{B_i}). \tag{14}$$

Therefore, we have

$$\begin{aligned}
&\mathrm{WCOST}_H(T) \\
&= \frac{1}{2} \cdot \sum_{s=0}^{\ell} \sum_{i \in [t_s]} W(B_i, V \setminus \overline{B_i}) && \text{By (14)} \\
&\ge \frac{1}{2} \cdot \sum_{s=0}^{\ell} \sum_{i \in [t_s]} \frac{W'(B_i, V \setminus \overline{B_i}) - \alpha}{\beta} && \text{As for every } S \subseteq V,\ W'(S, V \setminus S) \le \beta \cdot W(S, V \setminus S) + \alpha \\
&= \frac{1}{\beta} \cdot \sum_{s=0}^{\ell} W'_{T(s)} - \frac{\alpha}{2 \cdot \beta} \cdot \sum_{s=0}^{\ell} t_s \\
&\ge \frac{1}{\beta} \cdot \sum_{s=0}^{\ell} W'_{T(s)} - \frac{\alpha}{2 \cdot \beta} \cdot \ell \cdot |V| && \text{As } t_s \le |\textsc{leaves}(T')| = |V| \\
&= \frac{1}{\beta} \cdot \mathrm{WCOST}_{H'}(T) - \frac{\alpha}{2 \cdot \beta} \cdot \ell \cdot |V| && \text{By Lemma 9} \\
&\ge \frac{1}{\beta} \cdot \mathrm{WCOST}_{H'}(T') - \frac{\alpha}{2 \cdot \beta} \cdot \ell \cdot |V| && \text{By optimality of } T' \text{ on } H'
\end{aligned}$$

Therefore, we get

$$\mathrm{WCOST}_{H'}(T') \le \beta \cdot \mathrm{WCOST}_H(T) + \frac{\alpha}{2} \cdot |V| \cdot ||w||_1.$$

□

Next, we prove the following lemma which is an important intermediate step towards Lemma 8.

Lemma 11. Let $H = G/\mathcal{C}$ be the contraction of G with respect to the partition $\mathcal{C}$ (Definition 4) and let T^*_H denote an optimum weighted Dasgupta tree for H. Let $\widetilde{H} = \left([k], \binom{[k]}{2}, \widetilde{W}, \widetilde{w}\right)$ be an approximation of H such that the following hold:

- for all $i \in [k]$, $\frac{w(i)}{2} \le \widetilde{w}(i) \le 2 \cdot w(i)$, and
- for all $S \subseteq [k]$, $a \cdot W(S, \overline{S}) - b \le \widetilde{W}(S, \overline{S}) \le a' \cdot W(S, \overline{S}) + b'$.

Let $\widetilde{T} = \mathrm{WRSC}(\widetilde{H})$ denote a hierarchical clustering tree constructed on the graph $\widetilde{H}$ using the recursive sparsest cut algorithm. Then

$$\frac{a}{2} \cdot \mathrm{WCOST}_H(T^*_H) - b \cdot n \cdot k \le \mathrm{WCOST}_{\widetilde{H}}(\widetilde{T}) \le O\left(a' \sqrt{\log k} \cdot \mathrm{WCOST}_H(T^*_H) + b' n \cdot k \cdot \sqrt{\log k}\right)$$

Proof. Let $T^*_{\widetilde{H}}$ be the optimum Dasgupta tree of $\widetilde{H}$. By Corollary 2 the tree $\widetilde{T}$ returned by the vertex-weighted recursive sparsest cut procedure satisfies

$$\mathrm{WCOST}_{\widetilde{H}}(T^*_{\widetilde{H}}) \le \mathrm{WCOST}_{\widetilde{H}}(\widetilde{T}) \le O(\sqrt{\log k}) \cdot \mathrm{WCOST}_{\widetilde{H}}(T^*_{\widetilde{H}}) \tag{15}$$

Recall $H = \left([k], \binom{[k]}{2}, W, w\right)$ is the contraction of G with respect to $\mathcal{C}$, i.e., $H = G/\mathcal{C} =$ (Definition 4). Recall that $\widetilde{H} = \left([k], \binom{[k]}{2}, \widetilde{W}, \widetilde{w}\right)$ where $\widetilde{H}$ satisfies items 1 and 2 in the premise. Now we define $\widehat{H} = \left([k], \binom{[k]}{2}, W, \widetilde{w}\right)$, where W is the matrix of edge weights in H, and $\widetilde{w}$ is the vector of vertex weights in $\widetilde{H}$. Let $T^*_{\widehat{H}}$ be the optimum Dasgupta tree of $\widehat{H}$. Note that the trees $T^*_{\widehat{H}}$ and $\widetilde{T}$ satisfy the pre-requisites of Lemma 10. Therefore, we have

$$a \cdot \mathrm{WCOST}_{\widehat{H}}(T^*_{\widehat{H}}) - \frac{b}{2} \cdot k \cdot ||\widetilde{w}||_1 \leq \mathrm{WCOST}_{\widetilde{H}}(T^*_{\widetilde{H}}) \leq a' \cdot \mathrm{WCOST}_{\widehat{H}}(T^*_{\widehat{H}}) + \frac{b'}{2} \cdot k \cdot ||\widetilde{w}||_1 \tag{16}$$

Note that the vertex weight function w of H satisfies $w_i = |C_i|$ for all $i \in [k]$. Also, recall from the premise, we have $\frac{1}{2} \cdot w(i) \leq \widetilde{w}(i) \leq 2 \cdot w(i)$. Therefore, by Definition 9 we have

$$\begin{aligned} \mathrm{WCOST}_{\widehat{H}}(T^*_{\widehat{H}}) &\leq \mathrm{WCOST}_{\widehat{H}}(T^*_{H}) && \text{By optimality of } T^*_{\widehat{H}} \text{ on } \widehat{H} \\ &\leq 2 \cdot \mathrm{WCOST}_{H}(T^*_{H}) && \text{As } \widetilde{w}(i) \leq 2 \cdot w(i) \end{aligned} \tag{17}$$

Similarly, we have

$$\begin{aligned} \mathrm{WCOST}_{\widehat{H}}(T^*_{\widehat{H}}) &\geq \frac{1}{2} \cdot \mathrm{WCOST}_{H}(T^*_{\widehat{H}}) && \text{As } \widetilde{w}(i) \geq \frac{1}{2} \cdot w(i) \text{ on } \widehat{H} \\ &\geq \frac{1}{2} \cdot \mathrm{WCOST}_{H}(T^*_{H}) && \text{By optimality of } T^*_{\widehat{H}} \end{aligned} \tag{18}$$

Therefore, by (16), (17), (18), and as $||\widetilde{w}||_2 \leq 2||w||_2 = 2n$, we have

$$\frac{a}{2} \cdot \mathrm{WCOST}_{H}(T^*_{H}) - b \cdot n \cdot k \leq \mathrm{WCOST}_{\widetilde{H}}(T^*_{\widetilde{H}}) \leq 2 \cdot a' \cdot \mathrm{WCOST}_{H}(T^*_{H}) + b' \cdot n \cdot k \tag{19}$$

Finally, by (15) and (19) we get

$$\frac{a}{2} \cdot \mathrm{WCOST}_{H}(T^*_{H}) - b \cdot n \cdot k \leq \mathrm{WCOST}_{\widetilde{H}}(\widetilde{T}) \leq O\left(a' \sqrt{\log k} \cdot \mathrm{WCOST}_{H}(T^*_{H}) + b' \cdot n \cdot k \sqrt{\log k}\right).$$

□

Finally, we prove Lemma 8.

Lemma 8. Let $\frac{1}{n^4} < \xi < 1$, $\xi' = \xi/16$. Let $H = G/\mathcal{C}$ be the contraction of G with respect to the partition $\mathcal{C}$ (Definition 4) and let T^*_H denote an optimum weighted Dasgupta tree for H. Let $\widetilde{H} = \left([k], \binom{[k]}{2}, \widetilde{W}, \widetilde{w},\right)$ be the graph obtained by ApproxContractedGraph$(G, \xi', \mathcal{D})$ (Algorithm 4). Let $\widetilde{T} = \mathrm{WRSC}(\widetilde{H})$ denote a hierarchical clustering tree constructed on the graph $\widetilde{H}$ using the recursive sparsest cut algorithm. Then with probability at least $1 - 2 \cdot n^{-100}$ we have

$$\Omega\left(\varphi^2\right) \cdot \mathrm{WCOST}_{H}(T^*_{H}) - \xi m n k^2 \leq \mathrm{WCOST}_{\widetilde{H}}(\widetilde{T}) \leq O\left(\frac{\sqrt{\log k}}{\varphi^2} \cdot \mathrm{WCOST}_{H}(T^*_{H}) + \xi m n k^2 \sqrt{\log k}\right)$$

Proof. Recall that $\widetilde{H} =$ ApproxContractedGraph$(G, \xi', \mathcal{D})$ and let $\widetilde{\mathcal{L}}$ be the Laplacian of $\widetilde{H}$. Therefore, by Theorem A.2 with probability at least $1 - n^{-100}$, we have

$$\Omega\left(\varphi^2\right) \mathcal{L}_H - \xi \cdot m \cdot I_k \preceq \widetilde{\mathcal{L}} \preceq O\left(\frac{1}{\varphi^2}\right) \cdot \mathcal{L}_H + \xi \cdot m \cdot I_k \tag{20}$$

Note that for every $S \subseteq [k]$, we have $\mathbb{1}_S^T \mathcal{L}_H \mathbb{1}_S = W(S, \overline{S})$, and $\mathbb{1}_S^T \widetilde{\mathcal{L}} \mathbb{1}_S = \widetilde{W}(S, \overline{S})$. Also, note that $||\mathbb{1}_S||_2^2 \leq k$. Therefore, by (20) with probability at least $1 - n^{-100}$ for every $S \in [k]$ we have

$$\Omega\left(\varphi^2\right) \cdot W(S, V \setminus S) - \xi \cdot m \cdot k \leq \widetilde{W}(S, V \setminus S) \leq O\left(\frac{1}{\varphi^2}\right) \cdot W(S, V \setminus S) + \xi \cdot m \cdot k \tag{21}$$

Moreover, note that by Lemma 7 with probability at least $1 - n^{-100 \cdot k}$ we have $\frac{1}{2} \cdot w(i) \leq \widetilde{w}(i) \leq 2 \cdot w(i)$. Now, apply Lemma 11. With probability at least $1 - n^{-100 \cdot k} - n^{-100} \geq 1 - 2 \cdot n^{-100}$, this gives

$$\Omega\left(\varphi^2\right) \cdot \mathrm{WCOST}_H(T_H^*) - \xi m n k^2 \leq \mathrm{WCOST}_{\widetilde{H}}(\widetilde{T}) \leq O\left(\frac{\sqrt{\log k}}{\varphi^2} \cdot \mathrm{WCOST}_H(T_H^*) + \xi m n k^2 \sqrt{\log k}\right)$$

□

A.3.2 Bounding the optimum cost of the graph with the contracted graph

The main result of this section is Lemma 12, that relates the cost of the optimum tree of the contracted graph with the cost of the optimum of G. This allows us to bound the estimator proposed in Algorithm 5 with the Dasgupta cost of the optimum tree of G.

Lemma 12. Let $H = G/\mathcal{C}$ be the contraction of G with respect to the partition $\mathcal{C}$ (Definition 4). Let T_H^* and T_G^* be optimum weighted Dasgupta trees for H and G respectively. Then we have

$$\mathrm{COST}_G(T_G^*) \leq \mathrm{WCOST}_H(T_H^*) + \textsc{TotalClustersCost}(G) \leq O\left(\frac{1}{\varphi^7}\right) \cdot \mathrm{COST}_G(T_G^*),$$

where $\textsc{TotalClustersCost}(G)$ is an output of Algorithm 6 which satisfies the guarantees of Theorem A.4

Theorem A.4. *Let $G = (V, E)$ be a d-regular (k, φ, ϵ)-clusterable graph. For every $i \in [k]$, let $G\{C_i\}$ denote the induced subgraph on C_i with added self loops so that the degrees in $G\{C_i\}$ and G are the same, and let T_i^* denote the tree with optimum Dasgupta cost for $G\{C_i\}$. Then procedure* $\textsc{TotalClustersCost}$ *(Algorithm 6) returns a value such that:*

$$\sum_{i \in [k]} COST_{G\{C_i\}}(T_i^*) \leq \textsc{TotalClustersCost}(G) \leq O\left(\frac{1}{\varphi^5}\right) \cdot \sum_{i \in [k]} COST_{G\{C_i\}}(T_i^*).$$

We prove Theorem A.4 in Section B.3. To prove Lemma 12, we show that there exists a tree on the contracted graph whose cost is not more than $\frac{1}{\varphi^{O(1)}}$ times the optimum cost of the graph (see Lemma 14). To show this, we exploit the structure of (k, φ, ϵ)-clusterable graphs and prove that in these graphs some *cluster respecting cut* has conductance comparable to the conductance of the sparsest cut (see Lemma 13).

Definition 14 (**Cluster respecting cuts**). Let G be a graph that admits a (k, φ, ϵ)-clustering $C_1, \ldots C_k$. We say that the the cut $(B, \overline{B})$ is a cluster respecting cut with respect to the partition $\mathcal{C}$ if both B and $\overline{B}$ are disjoint unions of the clusters. That is, there exists a subset $I \subseteq [k]$ such that $B = \cup_{i \in I} C_i$, and $\overline{B} = \cup_{i \notin I} C_i$.

Definition 15 (**Cluster respecting tree**). Let $G = (V, E)$ be a graph that admits a (k, φ, ϵ)-clustering $\mathcal{C} = C_1, \ldots C_k$. We say that tree T for G (with $|V|$ leaves) is cluster respecting with respect to the partition $\mathcal{C}$ if there exists a subtree $T_{[k]}$ of T (rooted at the root of T) with k leaves such that for every $i \in [k]$, then there exists a unique leaf ℓ_i (of $T_{[k]}$) such that the leaves in T which are descendants of ℓ_i are exactly the set C_i. We call the tree $T_{[k]}$ the contracted subtree of T.

Lemma 13 (Some cluster respecting cut has conductance comparable to the sparsest cut). Let $(S, V \setminus S)$ denote the sparsest cut of G. Then there exists a cluster respecting cut (Definition 14) $(B, V \setminus B)$ such that

$$\max(\phi^G_{\text{out}}(B), \phi^G_{\text{out}}(\overline{B})) \leq \frac{4 \cdot \phi^G_{\text{out}}(S)}{\varphi}.$$

Proof. Let $(S, \overline{S})$ denote the cut with the smallest conductance in G, where $\text{vol}(S) \leq \text{vol}(V \setminus S)$. If $(S, \overline{S})$ is cluster respecting, then we are already done. So, let us suppose it is not cluster respecting and let $\beta = \phi_{\text{in}}(G) = \phi_{\text{out}}(S)$. For each $i \in [k]$, define $X_i = S \cap C_i$ to be the vertices of S that belong to C_i. Similarly, for each $i \in [k]$, define $Y_i = \overline{S} \cap C_i$. Also, we use M_i to denote for each $i \in [k]$, the set with smaller volume between X_i and Y_i. That is, $M_i = \arg\min(\text{vol}(X_i), \text{vol}(Y_i))$. Define $M = \cup_{i \in [k]} M_i$ and let

$$B = (S \setminus M) \cup (\overline{S} \cap M) \textit{ and } \overline{B} = V \setminus B.$$

This makes $(B, \overline{B})$ a *cluster respecting cut*. This is because we move vertices of M from $\overline{S}$ to S to get B and the other way around to get $\overline{B}$. We will show that the sets B and $\overline{B}$ are both non-empty and we will upperbound $\max(\phi_{\text{out}}(B), \phi_{\text{out}}(\overline{B})) = \dfrac{|E(B, \overline{B})|}{\min(\text{vol}(B), \text{vol}(\overline{B}))}$. By definition of B, it holds that

$$|E(B, \overline{B})| \leq |E(S, \overline{S})| + \text{vol}(M) = \beta \text{vol}(S) + \text{vol}(M).$$

Also $\cup_{i \in [k]} E(X_i, Y_i) \subseteq E(S, \overline{S})$. Moreover, recalling that each cluster has $\phi_{\text{in}}(C_i) \geq \varphi$, it follows that $|E(S, \overline{S})| \geq \sum_{i \in [k]} |E(X_i, Y_i)| \geq \varphi \cdot \text{vol}(M)$. Therefore, $\varphi \cdot \text{vol}(M) \leq |E(S, \overline{S})| = \beta \cdot \text{vol}(S)$. This means

$$|E(B, \overline{B})| \leq \beta \cdot \text{vol}(S) + \frac{\beta \cdot \text{vol}(S)}{\varphi} \leq \frac{2\beta \cdot \text{vol}(S)}{\varphi} \tag{22}$$

We define an index set $I_{\text{small}} = \{i \in [k] : \text{vol}(X_i) \leq \text{vol}(Y_i)\}$ which indexes clusters where S contains smaller volume of C_i than $\overline{S}$. We show that if $\sum_{i \in I_{\text{small}}} \text{vol}(X_i) < \text{vol}(S)/10$, then $\min(\text{vol}(B), \text{vol}(\overline{B})) \geq 0.5\text{vol}(S)$. Moreover, we also show that the other case, with $\sum_{i \in I_{\text{small}}} \text{vol}(X_i) < \text{vol}(S)/10$ cannot occur. In all, this means $\max(\phi_{\text{out}}(B), \phi_{\text{out}}(\overline{B})) \leq \frac{4\beta}{\varphi} \cdot \phi_{\text{out}}(S)$. Let us consider the first situation above.

Case 1: Suppose $\sum_{i \in I_{\text{small}}} \text{vol}(X_i) < \text{vol}(S)/10$. Note that for any $i \notin I_{\text{small}}$, $B \supseteq X_i$. Thus, we have $\text{vol}(B) \geq 0.9\text{vol}(S) \geq 0.5\text{vol}(S)$ (and thus B is non empty). Also, note that in this case,

$$\text{vol}(\overline{B}) \geq \text{vol}(\overline{S}) - \text{vol}(M) \geq \text{vol}(S) - \text{vol}(M) \geq \text{vol}(S) - \frac{\beta}{\varphi}\text{vol}(S)$$

which is at least $0.5\text{vol}(S)$ as well (and in particular, $\overline{B}$ is also non empty). Thus, $\min(\text{vol}(B), \text{vol}(\overline{B})) \geq 0.5\text{vol}(S)$. This gives $\max(\phi_{\text{out}}(B), \phi_{\text{out}}(\overline{B}) \leq \frac{4\beta}{\varphi}$. In either case, we note that the cut $(B, \overline{B})$ is cluster respecting and satisfies that $\max(\phi_{\text{out}}(B), \phi_{\text{out}}(\overline{B})) \leq \frac{4\beta}{\varphi}$.

Case 2: Now, we rule out the case $\sum_{i \in I_{\text{small}}} \text{vol}(X_i) \geq \text{vol}(S)/10$. Note that all X_i's are disjoint (as they are contained in different clusters). Observe

$$|E(S, \overline{S})| \geq \sum_{i \in I_{\text{small}}} |E(X_i, Y_i)| \geq \varphi \sum_{i \in I_{\text{small}}} \text{vol}(X_i) \geq \varphi \cdot \frac{\text{vol}(S)}{10}.$$

The first step follows because for all $i \in I_{\text{small}}$, $E(X_i, Y_i) \subseteq E(S, \overline{S})$. Moreover, (X_i, Y_i) is a cut of the cluster C_i with $\phi_{\text{in}}(C_i) \geq \varphi$. However, this means that $\phi_{\text{out}}(S) \geq \varphi/10$. But G is

(k, φ, ϵ)-clusterable and S is the sparset cut in G. Thus, it better hold that $\phi_{\text{out}}(S) \leq \epsilon < \varphi/10$ which leads to a contradiction. $\square$

The following claim is an aside which proves the tightness of Lemma 13. This claim shows that the $O(1/\varphi)$ loss in approximation to conductance is inherent when we take a cluster respecting cut as opposed to the sparsest cut. The proof can be found in Appendix A.4.

Claim 1 (Tightness of Lemma 13). Let $d > 3$ be a constant. Then, there exist a $(2, \varphi, \epsilon)$ clusterable, d-regular graph G such that

$$\min\left(\phi_{\text{out}}(B), \phi_{\text{out}}(V \setminus B)\right) \geq \phi_{\text{in}}(G),$$

where $(B, \overline{B})$ is the unique cluster respecting cut of G.

The following observation shows that in a cluster respecting cut $(B, V \setminus B)$, the sets $B, \overline{B}$ induce (k', φ, ϵ)-clusterable graphs themselves. This is later used to prove Lemma 14.

Observation 1. Let $(B, V \setminus B)$ be a cluster respecting cut in G with respect to the partition $\mathcal{C}$ (Definition 14). Suppose that B contains $k' < k$ clusters in G. For every $S \subseteq V$, let $G\{S\}$ be a graph obtained by adding $d_x - d_x^S$ self-loops to every vertex $x \in S$, where d_x^S is the degree of vertex x in S and d_x denotes the original degree of x in G. Then, we have $G\{B\}$ is (k', φ, ϵ)-clusterable and $G\{V \setminus B\}$ is $(k - k', \varphi, \epsilon)$-clusterable.

Proof. We will show this observation for B. A similar argument holds for $V \setminus B$. For every cluster $C_i \subseteq B$, we have

$$\phi_{\text{out}}^{G\{B\}}(C_i) = \frac{|E(C_i, B \setminus C_i)|}{\text{vol}(C_i)} \leq \frac{\epsilon \cdot \text{vol}(C_i)}{\text{vol}(C_i)} = \epsilon.$$

Also, note that for every cluster $C_i \subseteq B$ and every $S \subseteq C_i$ with $\text{vol}(S) \leq \text{vol}(C_i)/2$ we have

$$\phi_{\text{in}}^{G\{B\}}(S) = \frac{|E(S, C_i \setminus S)|}{\text{vol}(S)} \geq \varphi.$$

Therefore, for every cluster $C_i \subseteq B$ we have $\phi_{\text{out}}^{G\{B\}}(C_i) \leq \epsilon$ and $\phi_{\text{in}}^{G\{B\}}(C_i) \geq \varphi$. Thus, B is (k', φ, ϵ)-clusterable. $\square$

As a corollary of Lemma 13, we get that there exists a cluster respecting cut which is approximately sparsest, as per Definition 8.

Corollary 3. [Some cluster respecting cut is an approximate vertex weighted sparest cut] Let $(S, V \setminus S)$ denote the cut minimizing the sparsity $\text{Sparsity}_G(S)$ (Definition 8). Then there exists a cluster respecting cut (Definition 14) $(B, V \setminus B)$ such that

$$\text{Sparsity}_G(B) \leq \frac{8}{\varphi} \cdot \text{Sparsity}_G(S).$$

Proof. Let $(S, \bar{S})$ denote the cut with the smallest conductance in G, with $|S| \leq |V \setminus S|$. Since the graph G is unweighted, from Definition 8, we obtain

$$\text{Sparsity}_G(S) = \frac{|E(S, V \setminus S)|}{|S| \cdot |V \setminus S|} \geq \frac{|E(S, V \setminus S)|}{|S| \cdot n} = \frac{d}{n} \cdot \frac{|E(S, V \setminus S)|}{d|S|} = \frac{d}{n} \phi_{\text{out}}^G(S). \tag{23}$$

By Lemma 13, there exists a cluster respecting cut $(B, \bar{B})$ with $|B| \leq |V \setminus B|$ such that

$$\phi_{\text{out}}^G(B) \leq \frac{4\phi_{\text{out}}^G(S)}{\varphi}. \tag{24}$$

By Definition 8 again, we obtain

$$\mathrm{Sparsity}_G(B) = \frac{|E(B, V \setminus B)}{|B| \cdot |V \setminus B|} \leq \frac{|E(B, V \setminus B)}{|B| \cdot n/2} = \frac{2d}{n} \cdot \frac{|E(B, V \setminus B)}{d|B|} = \frac{2d}{n} \phi^G_{\mathrm{out}}(B), \tag{25}$$

where the inequality follows since $|V \setminus B| \geq n/2$ by the assumption that $|B| \leq |V \setminus B|$.

Combining Equations (23), (24) and (25), gives

$$\mathrm{Sparsity}(B) \leq \frac{2d}{n} \cdot \phi^G_{\mathrm{out}}(B) \leq \frac{2d}{n} \cdot \frac{4\phi^G_{\mathrm{out}}(S)}{\varphi} \leq \frac{8}{\varphi} \cdot \mathrm{Sparsity}(S).$$

□

Next, we present Lemma 14 which uses Corollary 3 to show that there exists a cluster respecting tree whose cost is not more than $O\left(\frac{1}{\varphi}\right)$ times the optimum cost of the graph.

Lemma 14. Let T^*_G be a tree with the optimum Dasgupta cost on G. Then, there exists a cluster respecting tree T with respect to $\mathcal{C}$ on G (Definition 15) such that

$$\mathrm{COST}_G(T) \leq O\left(\frac{1}{\varphi}\right) \cdot \mathrm{COST}_G(T^*_G).$$

Moreover the contracted tree of G with respect to $\mathcal{C} = (C_1, C_2, \ldots, C_k)$ (Definition 15) is a binary tree.

Proof. Let $H = G/\mathcal{C}$ denote the contraction of G with respect to $\mathcal{C}$. Let $\mathcal{T}_H$ denote the set of trees supported on k leaves (which correspond to the k clusters). We will construct a cluster respecting tree T for G with the claimed properties. This is done in two phases. In the first phase, we show it is possible to obtain a tree $T' \in \mathcal{T}_H$ by repeatedly applying Corollary 3. In phase two, we extend this tree further by refining each of the k leaves in T' further to obtain a tree T for G. Assuming the T' after phase one indeed belongs to $\mathcal{T}_H$, we obtain T in the following way: For $i \in [k]$, take the leaf C_i (in T') and extend it T^*_i, where T^*_i is the tree with optimum Dasgupta cost on induced graph $G\{C_i\}$.

Phase 1: Producing T'. We show how to generate T' level by level. It will be convenient to attach to each node in T' a set $I \subseteq [k]$ of indices. We begin by attaching $[k]$ to the root. An application of Corollary 3 produces a cut $(B, \overline{B})$ with sparsity (as per Definition 8) within a factor $O(1/\varphi)$ of the sparsest cut in G which additionally is cluster respecting (see Definition 14). Make B and $\overline{B}$ the two children of the root and attach to B the index set $I_B \subseteq [k]$ which denotes the clusters contained in B (similarly define $I_{\overline{B}} \subseteq [k]$). Recurse and apply Corollary 3 to both $G\{B\}$ and $G\{\overline{B}\}$. Note that we can do this because both $G\{B\}$ (resp. $G\{\overline{B}\}$) are (k', φ, ϵ)-clusterable (resp $(k-k', \varphi, \epsilon)$-clusterable) for some $1 \leq k' < k$ (Observation 1). Thus, in all, phase 1 returns a tree $T' \in \mathcal{T}_H$.

Phase 2: Extending T' On termination, as seen earlier, phase 1 produces a tree T' whose leaves correspond to the clusters $C_1, C_2, \ldots C_k$ in some order. Note that

$$\mathrm{COST}_G(T) = \mathrm{WCOST}_H(T') + \sum_{i=1}^{k} \mathrm{COST}_{G\{C_i\}}(T^*_i). \tag{26}$$

We will show this cost is at most $O(1/\varphi) \cdot \mathrm{COST}_G(T^*_G)$. To this end, let us consider another tree T^{RSC} that is obtained by taking the tree T' and extending it by repeated applications of *exact* recursive sparses cut procedure (with approximation factor being 1). Thus, the tree T^{RSC} obtained this way is within a factor $O(1/\varphi)$ for the sparsest cut at each step. By Theorem 2.7, we get

$$\mathrm{COST}_G(T^{\mathrm{RSC}}) \leq O\left(\frac{1}{\varphi}\right) \cdot \mathrm{COST}_G(T^*_G) \tag{27}$$

Let us rewrite this approximation in more detail. For $i \in [k]$, letting $T^{\mathrm{RSC}}(i)$ denote the tree obtained by applying recursive sparsest cut on each C_i. Thus, by (27) we get

$$O\left(\frac{1}{\varphi}\right) \cdot \mathrm{COST}_G(T_G^*) \geq \mathrm{COST}_G(T^{\mathrm{RSC}}) = \mathrm{WCOST}_H(T') + \sum_{i \in [k]} \mathrm{COST}_{G\{C_i\}}(T^{\mathrm{RSC}}(i)) \tag{28}$$

Now two cases arise. First, consider the case where $\mathrm{WCOST}_H(T') \leq \sum_{i \in [k]} \mathrm{COST}_{G\{C_i\}}(T_i^*)$. In this case, by (26) we have $\mathrm{COST}_G(T_G^*) \leq \mathrm{COST}_G(T) \leq 2 \sum_{i \in [k]} \mathrm{COST}_{G\{C_i\}}(T_i^*)$. Also, we have $\mathrm{COST}_G(T_G^*) \geq \sum_{i \in [k]} \mathrm{COST}_{G\{C_i\}}(T_i^*)$. Thus, we have

$$\mathrm{COST}_G(T) \leq 2 \cdot \mathrm{COST}_G(T_G^*).$$

Now consider the other case where $\mathrm{WCOST}_H(T') > \sum_{i \in [k]} \mathrm{COST}_{G\{C_i\}}(T_i^*)$. Recall from (26) that

$$\mathrm{COST}_G(T) = \mathrm{WCOST}_H(T') + \sum_{i \in [k]} \mathrm{COST}_{G\{C_i\}}(T_i^*) \leq 2 \cdot \mathrm{WCOST}_H(T') \leq O\left(\frac{1}{\varphi}\right) \cdot \mathrm{COST}_G(T_G^*),$$

where, the last inequality holds by (28). Moreover, by construction, the tree T' is the a binary tree (and is the contracted tree of T) as desired.

□

Finally, we prove Lemma 12.

Lemma 12. Let $H = G/\mathcal{C}$ be the contraction of G with respect to the partition $\mathcal{C}$ (Definition 4). Let T_H^* and T_G^* be optimum weighted Dasgupta trees for H and G respectively. Then we have

$$\mathrm{COST}_G(T_G^*) \leq \mathrm{WCOST}_H(T_H^*) + \textsc{TotalClustersCost}(G) \leq O\left(\frac{1}{\varphi^7}\right) \cdot \mathrm{COST}_G(T_G^*),$$

where $\textsc{TotalClustersCost}(G)$ is an output of Algorithm 6 which satisfies the guarantees of Theorem A.4

Proof of Lemma 12. By Lemma 14, we know there is a cluster respecting tree T (Definition 15) on n leaves such that

$$\mathrm{COST}_G(T_G^*) \geq \Omega(\varphi) \cdot \mathrm{COST}_G(T) \tag{29}$$

Let T' be the corresponding contracted tree obtained from T (Definition 15). Note that by construction T in Lemma 14 we have T' has k leaves such that for every $i \in [k]$, we extend the leaf corresponding to C_i, by tree T_i^*, where, T_i^* is the tree with the optimum Dasgupta cost on the induced subgraph $G\{C_i\}$.

Thus, we have

$$\mathrm{COST}_G(T) = \mathrm{WCOST}_H(T') + \sum_{i \in [k]} \mathrm{COST}_{G\{C_i\}}(T_i^*) \tag{30}$$

Therefore, we have

$$
\begin{aligned}
&\mathrm{COST}_G(T_G^*) \\
&\geq \Omega(\varphi) \cdot \mathrm{COST}_G(T) && \text{By (29)} \\
&= \Omega(\varphi) \cdot \left(\mathrm{WCOST}_H(T') + \sum_{i \in [k]} \mathrm{COST}_{G\{C_i\}}(T_i^*) \right) && \text{By (30)} \\
&\geq \Omega(\varphi) \cdot \left(\mathrm{WCOST}_H(T_H^*) + \sum_{i \in [k]} \mathrm{COST}_{G\{C_i\}}(T_i^*) \right) && \text{By optimality of } T_H^* \text{ on } H \\
&\geq \Omega(\varphi) \cdot \mathrm{WCOST}_H(T_H^*) + \Omega(\varphi^7) \cdot \textsc{TotalClustersCost}(G) && \text{By Theorem A.4} \\
&\geq \Omega(\varphi^7) \cdot \left(\mathrm{WCOST}_H(T_H^*) + \textsc{TotalClustersCost}(G) \right). && (31)
\end{aligned}
$$

Also by Theorem A.4 we have

$$
\begin{aligned}
\mathrm{COST}_G(T_G^*) &\leq \mathrm{WCOST}_H(T_H^*) + \sum_{i=1}^{k} \mathrm{COST}_{G\{C_i\}}(T_i^*) \\
&\leq \mathrm{WCOST}_H(T_H^*) + \textsc{TotalClustersCost}(G).
\end{aligned}
\tag{32}
$$

□

A.3.3 Proof of Theorem 2.1

Finally, to prove Theorem 2.1 we prove an intermediate step, which is Lemma 15.

Lemma 15. Let $H = G/\mathcal{C}$ be the contraction of G with respect to the partition $\mathcal{C}$ (Definition 4) and let T_H^* denote an optimum weighted Dasgupta tree for H. Let $0 < a < 1 < a'$ and $b \leq \frac{a}{kd\sqrt{\log k}}$. Let $\widetilde{H} = \left([k], \binom{[k]}{2}, \widetilde{W}, \widetilde{w}\right)$ be an approximation of H such that the following hold:

- For all $i \in [k]$, $\frac{w(i)}{2} \leq \widetilde{w}(i) \leq 2 \cdot w(i)$, and
- $a \cdot \mathrm{WCOST}_H(T_H^*) - b \cdot mn \leq \mathrm{WCOST}_{\widetilde{H}}(\widetilde{T}) \leq O\left(a'\sqrt{\log k} \cdot \mathrm{WCOST}_H(T_H^*) + b \cdot mn \cdot \sqrt{\log k}\right)$

where, $\widetilde{T} = \mathrm{WRSC}(\widetilde{H})$ denote a hierarchical clustering tree constructed on the graph $\widetilde{H}$ using the recursive sparsest cut algorithm. We set $\mathrm{EST} = \frac{1}{a} \cdot \mathrm{WCOST}_{\widetilde{H}}(\widetilde{T}) + \frac{b}{a} mn + \textsc{TotalClustersCost}(G)$. Then we have

$$
\mathrm{COST}_G(T_G^*) \leq \mathrm{EST} \leq O\left(\frac{a'\sqrt{\log k}}{a \cdot \varphi^7}\right) \cdot \mathrm{COST}_G(T_G^*).
$$

Proof. Let $\widetilde{H} = \left([k], \binom{[k]}{2}, \widetilde{W}, \widetilde{w}\right)$ denote the graph defined in the premise. Let $\widetilde{T} = \mathrm{RSC}(\widetilde{H})$ denote a hierarchical clustering tree constructed on the graph $\widetilde{H}$ using the recursive sparsest cut algorithm. We have that

$$
a \cdot \mathrm{WCOST}_H(T_H^*) - b \cdot mn \leq \mathrm{WCOST}_{\widetilde{H}}(\widetilde{T}) \leq O\left(a' \cdot \sqrt{\log k} \cdot \mathrm{WCOST}_H(T_H^*) + b \cdot mn\sqrt{\log k}\right) \tag{33}
$$

Note that
$$\mathrm{EST} = \frac{1}{a} \cdot \mathrm{WCOST}_{\widetilde{H}}(\widetilde{T}) + \frac{b}{a} mn + \textsc{TotalClustersCost}(G). \tag{34}$$
Therefore, by (33) and (34) we have
$$\begin{aligned}
&\mathrm{WCOST}_H(T_H^*) + \textsc{TotalClustersCost}(G) \le \mathrm{EST} \\
&\le O\left(\frac{a' \cdot \sqrt{\log k}}{a} \cdot \mathrm{WCOST}_H(T_H^*) + \frac{b \cdot mn}{a} \cdot \sqrt{\log k}\right) + \textsc{TotalClustersCost}(G).
\end{aligned} \tag{35}$$
Let T_G^* denote a Dasgupta tree with optimum cost for G. Then, by Lemma 12 we have
$$\mathrm{COST}_G(T_G^*) \le \mathrm{WCOST}_H(T_H^*) + \textsc{TotalClustersCost}(G) \le \left(\frac{1}{\varphi^7}\right) \cdot \mathrm{COST}_G(T_G^*). \tag{36}$$
By the first part of both (35) and (36) we have
$$\mathrm{COST}_G(T_G^*) \le \mathrm{EST}. \tag{37}$$
We also have
$$\begin{aligned}
&\mathrm{EST} \\
&\le O\left(\frac{a' \cdot \sqrt{\log k}}{a} \cdot \mathrm{WCOST}_H(T_H^*) + \frac{b \cdot mn \cdot \sqrt{\log k}}{a}\right) + \textsc{TotalClustersCost}(G) && \text{By (35)} \\
&\le O\left(\frac{a'\sqrt{\log k}}{a \cdot \varphi^7}\right) \mathrm{COST}_G(T_G^*) + O\left(\frac{b \cdot d \cdot n^2 \cdot \sqrt{\log k}}{a}\right) + \textsc{TotalClustersCost}(G) && \text{By (36)} \\
&\le O\left(\frac{a'\sqrt{\log k}}{a \cdot \varphi^7}\right) \mathrm{COST}_G(T_G^*) + O\left(\frac{n^2}{k}\right) + \textsc{TotalClustersCost}(G) && \text{As } b \le \frac{a}{d \cdot k\sqrt{\log k}} \\
&\le O\left(\frac{a'\sqrt{\log k}}{a\varphi^7}\right) \mathrm{COST}_G(T_G^*) + O(1) \cdot \textsc{TotalClustersCost}(G) && \text{By Observation 2} \\
&\le O\left(\frac{a'\sqrt{\log k}}{a\varphi^7}\right) \mathrm{COST}_G(T_G^*) + O\left(\frac{1}{\varphi^6}\right) \cdot \mathrm{COST}_G(T_G^*) && \text{By Theorem A.4} \\
&\le O\left(\frac{a'\sqrt{\log k}}{a \cdot \varphi^7}\right) \mathrm{COST}_G(T_G^*).
\end{aligned} \tag{38}$$
Therefore, by (37) and (38) we have
$$\mathrm{COST}_G(T_G^*) \le \mathrm{EST} \le O\left(\frac{a'\sqrt{\log k}}{a \cdot \varphi^7}\right) \cdot \mathrm{COST}_G(T_G^*).$$
□

Theorem 2.1. *Let $k \ge 2$, $\varphi \in (0,1)$ and $\frac{\epsilon}{\varphi^2}$ be a sufficiently small constant. Let $G = (V, E)$ be a bounded degree graph that admits a (k, φ, ϵ)-clustering $C_1, \ldots, C_k$. Let $|V| = n$.*

There exists an algorithm ($\textsc{EstimatedCost}(G)$*; Algorithm 1) that w.h.p. estimates the optimum Dasgupta cost of G within an $O\left(\frac{\sqrt{\log k}}{\varphi^{O(1)}}\right)$ factor in time $O^*\left(n^{1/2+O(\epsilon/\varphi^2)} \cdot (d_{\max})^{O(1)}\right)$ using $O^*\left(n^{O(\epsilon/\varphi^2)} \cdot (d_{\max})^{O(1)}\right)$ seed queries.*

Proof. Let $H = G/\mathcal{C}$ be the contraction of G with respect to the partition $\mathcal{C}$ (Definition 4) and let T_H^* denote an optimum weighted Dasgupta tree for H. Let $\widetilde{H} = \left([k], \binom{[k]}{2}, \widetilde{W}, \widetilde{w}\right)$ be the graph obtained by ApproxContractedGraph$(G, \xi, \mathcal{D})$ (Algorithm 4). Let $\widetilde{T} = \text{RSC}(\widetilde{H})$ denote a hierarchical clustering tree constructed on the graph $\widetilde{H}$ using the recursive sparsest cut algorithm. Therefore, by Lemma 8 with probability at least $1 - 2 \cdot n^{-100}$ we have

$$\Omega(\varphi^2) \cdot \text{WCOST}_H(T_H^*) - \xi mnk^2 \leq \text{WCOST}_{\widetilde{H}}(\widetilde{T}) \leq O\left(\frac{\sqrt{\log k}}{\varphi^2} \cdot \text{WCOST}_H(T_H^*) + \xi mnk^2 \sqrt{\log k}\right). \tag{39}$$

Note that as per line (5) of Algorithm 5 we estimate the Dasgupta cost of G by

$$\text{EST} = O\left(\frac{1}{\varphi^2}\right) \cdot \text{WCOST}_{\widetilde{H}}(\widetilde{T}) + \text{TotalClustersCost}(G) + O\left(\frac{\xi mnk^2}{\varphi^2}\right). \tag{40}$$

Set $a = c \cdot \varphi^2$, where c is the hidden constant in $\Omega(\varphi^2)$. Set $a' = 1/\varphi^2, b = \xi \cdot k^2$, where $\xi = \frac{c \cdot \varphi^2}{d \cdot k^3 \sqrt{\log k}}$ as per line 1 of Algorithm 5. Thus, $b = \frac{c \cdot \varphi^2}{d \cdot k \sqrt{\log k}} \leq \frac{a}{d \cdot k \sqrt{\log k}}$. So, we can apply Lemma 15, which gives

$$\text{COST}_G(T_G^*) \leq \text{EST} \leq O\left(\frac{a' \sqrt{\log k}}{a \cdot \varphi^7}\right) \cdot \text{COST}_G(T_G^*) = \frac{\sqrt{\log k}}{\varphi^{11}} \cdot \text{COST}_G(T_G^*).$$

Running Time: Now, we prove the running time bound. First, the EstimatedCost procedure calls WeightedDotProductOracle, which by Theorem A.3 has running time $O^*\left(n^{1/2+O(\epsilon/\varphi^2)} \cdot \left(\frac{1}{\xi}\right)^{O(1)}\right)$.

Then, the EstimatedCost procedure calls the ApproxContractedGraph procedure.

By Theorem A.2, this has running time $O^*\left(n^{1/2+O(\epsilon/\varphi^2)} \cdot \left(\frac{1}{\xi}\right)^{O(1)}\right)$. Finally, recall the procedure WRSC runs in time poly(k). Therefore, the overall running time of EstimatedCost procedure is seen to be $n^{1/2+O(\epsilon/\varphi^2)} \cdot \left(\frac{d \cdot k \cdot \log n}{\varphi \cdot \xi}\right)^{O(1)}$. Substituting in

$$\xi = \frac{\varphi^2}{d \cdot k^3 \cdot \sqrt{\log k}},$$

we get the required running time.

Finally, we bound the number of seed queries issued. First, let us consider the number of seed queries required by the ApproxContractedGraph procedure to estimate the cluster sizes $\widetilde{w}(i)$. These quantities are estimated to within a multiplicative $(1 \pm \delta)$ factor with $\delta = \frac{\xi}{512 \cdot k^2 \cdot n^{40 \cdot \epsilon/\varphi^2}}$ By simple Chernoff bounds, this can be done using

$$O\left(\frac{\log n \cdot k^2}{\delta^2}\right) = O\left(\frac{n^{80\epsilon/\varphi^2} \cdot k^6 \cdot \log n}{\xi^2}\right)$$

seeds. Let us now bound the number of samples taken by this procedure by plugging in the value of $\xi = \frac{\varphi^2}{d \cdot k^3 \cdot \sqrt{\log k}}$. This gives

$$s = \frac{10^9 \cdot \log n \cdot k^{12} \cdot n^{80 \cdot \epsilon/\varphi^2} \cdot \log k \cdot d_{\text{avg}}^2}{\varphi^4}.$$

Then, by Theorem A.4 the number of seeds taken to compute TotalClustersCost is $n^{1/3} \cdot \left(\frac{k \cdot \log n}{\varphi}\right)^{O(1)}$.

Combining, we obtain that the total number of seed queries issued is $O^*\left(n^{1/3} + n^{O(\epsilon/\varphi^2)} \cdot (d\cdot)^{O(1)}\right)$ as claimed. $\square$

A.4 Proof of Lemma 5, Lemma 7 and Claim 1

Lemma 5. Let $A, B \subseteq V$. Let $S_A \subseteq A$ and $S_B \subseteq B$ denote (multi)sets of vertices sampled independently and uniformly at random from A and B respectively, where $|S_A|, |S_B| \geq \frac{1600\cdot k^3 \cdot n^{40\epsilon/\varphi^2} \cdot \log n}{\xi^2}$. Let M denote the lazy random walk matrix of G, and $M = U\Sigma U^T$ be the eigendecomposition of M. Then, with probability at least $1 - n^{-100\cdot k}$ we have

$$\left| \mathbb{1}_A^T \cdot (U_{[k]}\Sigma_{[k]}U_{[k]}^T)\ \mathbb{1}_B\ - \frac{|A|\cdot|B|}{|S_A|\cdot|S_B|} \cdot \mathbb{1}_{S_A}^T (U_{[k]}\Sigma_{[k]}U_{[k]}^T)\mathbb{1}_{S_B} \right| \leq \xi \cdot n \tag{12}$$

Proof. Let $a = \mathbb{1}_A^T \cdot (U_{[k]}\Sigma_{[k]}^{1/2})$ and $b = (\Sigma_{[k]}^{1/2}U_{[k]}^T)\mathbb{1}_B$. This allows us to write the first term on LHS of Equation (12) as $\langle a, b\rangle$. Analogously, define $a' = \frac{|A|}{|S_A|} \cdot \mathbb{1}_{S_A}^T \cdot (U_{[k]}\Sigma_{[k]}^{1/2})$ and $b' = \frac{|B|}{|S_B|} \cdot (\Sigma_{[k]}^{1/2}U_{[k]}^T)\mathbb{1}_{S_B}$ so that the second term on LHS of Equation (12) can be written as $\langle a', b'\rangle$. With this setup, we have the following:

$$\left|\langle a, b\rangle - \langle a', b'\rangle\right| = \left|\langle a, b - b'\rangle + \langle a - a', b'\rangle\right| \leq \|a\|_2\|b - b'\|_2 + \|a - a'\|_2\|b'\|_2 \tag{41}$$

where the last inequality follows by triangle inequality and Cauchy-Schwarz. Now by expanding out a we get

$$a = \sum_{x\in A} \mathbb{1}_x^T \left(U_{[k]}\Sigma_{[k]}^{1/2}\right).$$

We show this quantity is estimated coordinate wise very well by the vector $a' \in \mathbb{R}^k$ defined as follows:

$$a' = \frac{|A|}{|S_A|} \cdot \sum_{x\in S_A} \mathbb{1}_x^T \left(U_{[k]}\Sigma_{[k]}^{1/2}\right).$$

Note that for all $i \in [k]$ we have $\mathbb{E}[a'(i)] = a(i)$. For any $i \in [k]$ we first show that with high probability $|a(i) - a'(i)|$ is small, then by union bound we prove that $||a' - a||$ is small. Note that for every $i \in [k]$ we have $a(i) = \sum_{x\in A} \sqrt{\sigma_i} \cdot u_i(x)$, where u_i is the i-th eigenvector of M and σ_i is the i-th eigenvalue of M. For every $x \in S_A$, let Z_x be a random variable defined as $Z_x = |A| \cdot \sqrt{\sigma_i} \cdot u_i(x)$. Thus, we have $a'(i) = \frac{1}{S_A} \cdot \sum_{x\in S_A} Z_x$, and $a(i) = \frac{1}{S_A} \cdot \sum_{x\in S_A} \mathbb{E}[Z_x]$. Therefore, by Hoeffding Bound (Fact 1), we have

$$\Pr\left[|a'(i) - a(i)| \geq t\right] \leq \exp\left(\frac{-2|S_A|\cdot t^2}{(2\cdot \max_{x\in S_A}|Z_x|)^2}\right). \tag{42}$$

Next we need to bound $\max_{x\in S_A}|Z_x|$. Note that for every $i \in [k]$ and every $x \in V$ we have

$$\begin{aligned}
|Z_x| &= |A| \cdot \sqrt{\sigma_i} \cdot |u_i(x)| \\
&\leq |A| \cdot ||u_i||_\infty && \text{As } \sigma_i = 1 - \frac{\lambda_i}{2} \leq 1 \\
&= |A| \cdot \frac{n^{20\epsilon/\varphi^2}}{\sqrt{\min_{i\in k}|C_i|}} && \text{By Lemma 6} \\
&\leq \frac{|A|\cdot n^{20\epsilon/\varphi^2}}{\sqrt{\frac{n}{k}}} && \text{As } \forall i \in k, |C_i| \approx \frac{n}{k}.
\end{aligned} \tag{43}$$

Let $w_A = \frac{|A|\cdot n^{20\epsilon/\varphi^2}}{\sqrt{\frac{n}{k}}}$ and $\beta = \frac{\xi}{2\cdot k\cdot n^{20\epsilon/\varphi^2}}$. By (42) and (43) we have

$$\Pr\left[|a'(i) - a(i)| \geq \beta \cdot w_A\right] \leq \exp\left(\frac{-2|S_A|\cdot(\beta\cdot w_A)^2}{4\cdot w_A^2}\right) = \exp\left(-|S_A|\beta^2/2\right) \leq n^{-200k},$$

where the last inequality holds by choice of $|S_A| \geq \frac{1600\cdot k^3\cdot n^{40\epsilon/\varphi^2}\cdot \log n}{\xi^2} \geq 400\cdot k\cdot \log n \cdot \frac{1}{\beta^2}$. Thus, by a union bound over all $i \in [k]$, with probability at least $1 - k\cdot n^{-200k}$ we have

$$||a - a'||_2 = \sqrt{\sum_{i=1}^{k}(a'(i) - a(i))^2} \leq \sqrt{k}\cdot\beta\cdot w_A. \tag{44}$$

A similar analysis shows that with probability at least $1 - k\cdot n^{-200k}$ we have

$$||b - b'||_2 \leq \sqrt{k}\cdot\beta\cdot w_B, \tag{45}$$

where, $w_B = \frac{|B|\cdot n^{20\epsilon/\varphi^2}}{\sqrt{\frac{n}{k}}}$. Also note that

$$\begin{aligned}
||a||_2 &= \left|\left|\mathbb{1}_A^T\left(U_{[k]}\Sigma_{[k]}^{1/2}\right)\right|\right|_2 \\
&\leq \left|\left|\mathbb{1}_A^T\right|\right|_2 \cdot \left|\left|U_{[k]}\right|\right|_2 \cdot \left|\left|\Sigma_{[k]}^{1/2}\right|\right|_2 \\
&\leq \sqrt{|A|}\cdot \max_{i\in[k]}\sqrt{\sigma_i} && \text{As } ||U_{[k]}||_2 = 1, \text{ and } \Sigma \text{ is diagonal} \\
&\leq \sqrt{n} && \text{As } \sigma_i = 1 - \frac{\lambda_i}{2} \leq 1
\end{aligned} \tag{46}$$

Similarly we have

$$||b||_2 \leq \sqrt{n}. \tag{47}$$

Thus by (41) we get

$$\begin{aligned}
\|a\|_2\|b - b'\|_2 + \|a - a'\|_2\|b'\|_2 &\leq \sqrt{k\cdot n}\cdot\beta\cdot(w_A + w_B) && \text{By (44), (45), (46), (47)} \\
&\leq \xi\cdot n && \text{As } w_A + w_B \leq \frac{2\cdot n\cdot n^{20\epsilon/\varphi^2}}{\sqrt{\frac{n}{k}}} \text{ and } \beta = \frac{\xi}{2\cdot k\cdot n^{20\epsilon/\varphi^2}}
\end{aligned}$$

Therefore, with probability at least $1 - 2\cdot k\cdot n^{-200k} \geq 1 - n^{-100k}$ we have

$$\left|\mathbb{1}_A^T\cdot(U_{[k]}\Sigma_{[k]}U_{[k]}^T)\ \mathbb{1}_B\ - \frac{|A|\cdot|B|}{|S_A|\cdot|S_B|}\cdot\mathbb{1}_{S_A}^T(U_{[k]}\Sigma_{[k]}U_{[k]}^T)\mathbb{1}_{S_B}\right| \leq \xi\cdot n.$$

□

Fact 1. (**Hoeffding Bounds**) Let $Z_1, Z_2, \cdots Z_n$ be iid random variables with $Z_i \in [a, b]$ for all $i \in [n]$ where $-\infty \leq a \leq b \leq \infty$. Then

$$\Pr\left[\frac{1}{n}\cdot\sum|(Z_i - \mathbb{E}[Z_i])| \geq t\right] \leq \exp(-2nt^2/(b-a)^2), \text{ and}$$
$$\Pr\left[\frac{1}{n}\cdot\sum|(Z_i - \mathbb{E}[Z_i])| \leq t\right] \leq \exp(-2nt^2/(b-a)^2).$$

Next, we prove Lemma 7.

Lemma 7. Let $S \subseteq V$ denote a set of random vertices returned by the RANDOMSAMPLEMODEL (Definition 11) in a regular graph. For every $i \in [k]$ let $S_i = S \cap C_i$. If $|S| \geq \frac{400 \cdot \log n \cdot k^2}{\delta^2}$, then with probability at least $1 - n^{-100 \cdot k}$ for every $i \in [k]$ we have $|S_i| \in (1 \pm \delta) \cdot |S| \cdot \frac{|C_i|}{n}$.

Proof. Let $s = |S|$. For $x \in V$, and $r \in [s]$, let Y_x^r be a random variable which is 1 if the r-th sampled vertex is v, and 0 otherwise. Thus $\mathbb{E}[Y_x^r] = \frac{1}{n}$. Observe that $|S_i| = |S \cap C_i|$ is a random variable defined as $\sum_{r=1}^{s} \sum_{x \in C_i} Y_x^r$ where its expectation is given by

$$\mathbb{E}[|S \cap C_i|] = \sum_{r=1}^{s} \sum_{x \in C_i} Y_x^r \geq s \cdot \frac{|C_i|}{n} \geq s \cdot \frac{\Omega(1)}{k},$$

where, the last inequality holds since all clusters have size $\Omega(n/k)$, since we assume $\mathrm{vol}(C_i)/\mathrm{vol}(C_j) = O(1)$ for all i, j and the graph is d-regular.

Notice that the random variables Y_x^r are negatively associated, since for each r, $\sum_{x \in V} Y_x^r = 1$. Therefore, by Chernoff bound,

$$\Pr\left[\left||S \cap C_i| - \frac{|C_i|}{n}\right| > \delta \cdot s \cdot \frac{|C_i|}{n}\right] \leq 2 \cdot \exp\left(-\frac{\delta^2}{3} \cdot \frac{s}{k}\right) \leq n^{-120 \cdot k},$$

where, the last inequality holds by choice of $s \geq \frac{400 \cdot \log n \cdot k^2}{\delta^2}$. Therefore, by union bound,

$$\Pr\left[\exists i : \left||S \cap C_i| - \frac{|C_i|}{n}\right| > \delta \cdot s \cdot \frac{|C_i|}{n}\right] \leq 2 \cdot k \cdot n^{-120 \cdot k} \leq n^{-100 \cdot k}.$$

□

Finally, we prove Claim 1.

Claim 1 (Tightness of Lemma 13). Let $d > 3$ be a constant. Then, there exist a $(2, \varphi, \epsilon)$ clusterable, d-regular graph G such that

$$\min\left(\phi_{\text{out}}(B), \phi_{\text{out}}(V \setminus B)\right) \geq \phi_{\text{in}}(G),$$

where $(B, \overline{B})$ is the unique cluster respecting cut of G.

Proof. We present such an instance G explicitly. We pick a large enough integer m. Let $X_1 = \{1, 2, \cdots, m - 2\epsilon m\}$ denote the set of first $m - 2\epsilon m$ integers and use standard constructions to obtain an $d-1$ regular $\varphi \cdot d$-expander on vertices in X_1. Also, let $Y_1 = \{m - 2\epsilon m + 1, m - 2\epsilon m + 2, \cdots m\}$ denote another set of $2\epsilon m$ integers and obtain another $d-1$ regular $\varphi \cdot d$ expander on Y_1. Put a matching of size $\epsilon\varphi dm$ between the sets X_1 and Y_1. Also, put a matching on remaining degree $d-1$ vertices in X_1. Notice that $|E(X_1, Y_1)| = \epsilon\varphi dm$. Let $C_1 = X_1 \cup Y_1$. Now we describe another set of vertices. This time we consider three sets: $X_2 = \{1, 2, \cdots, m - 4\epsilon m\}$, $Y_2 = \{m - 4\epsilon m + 1, \cdots m - 2\epsilon m\}$ and $Z_2 = \{m - 2\epsilon m + 1, \cdots, m\}$. We again obtain a $d-1$ regular $\varphi \cdot d$ expander on all of these sets. Next, add a matching of size $\epsilon\varphi dm$ between X_2 and Y_2 and between X_2 and Z_2. We add a matching between remaining degree $d-1$ vertices in X_2 and another matching between remanining degree $d-1$ vertices in Z_2. Notice that $|E(X_2, Y_2)| = \epsilon\varphi dm = |E(X_2, Z_2)|$. Next, let $C_2 = X_2 \cup Y_2 \cup Z_2$.

Finally, we add a matching between the remaining degree $d-1$ vertices in Y_1 and Y_2. Overall this gives a d-regular graph on $2m$ vertices. We let $B = C_1$ and thus $\overline{B} = C_2$. Notice that $\varphi_{\text{out}}(C_1) = \varphi_{\text{out}}(C_2) = \epsilon$. Also, by construction, note that $\varphi_{\text{in}}(C_1) = \varphi_{\text{in}}(C_2) \geq \varphi$. Now consider the following set $S = X_1 \cup Z_2$. We see that

$$|E(S, \overline{S})| = |E(X_1, Y_1)| + |E(X_2, Z_2)| = 2\epsilon\varphi dm.$$

Also $|S| = |X_1| + |Z_2| = m$. And therefore, it holds that $\phi(G) \leq \phi(S) = 2\epsilon\varphi$. □

B Sublinear estimator for cost of expanders

In this section, we formally prove Theorem 2.2 from Section 2.1, which demonstrates an algorithm for estimating the Dasgupta cost of a φ-expander up to a $\text{poly}(1/\varphi)$ factor using $\approx n^{1/3}$ seed queries.

Then, we prove Theorem A.4 which demonstrates an algorithm for estimating the total contribution of the clusters to the Dasgupta cost of a d-regular graph that admits (k, φ, ϵ)-clustering.

Theorem 2.2. *Let $G = (V, E)$ be a φ-expander (possibly with self-loops). Let T^* denote the tree with optimum Dasgupta cost for G. Then procedure* ClusterCost *(Algorithm 3), uses $O^*\left(n^{1/3}\right)$ seed queries and with probability $1 - n^{-101}$ returns a value such that:*

$$COST(T^*) \leq \textsc{ClusterCost}(G) \leq O\left(\frac{1}{\varphi^5}\right) \cdot COST(T^*).$$

Theorem A.4. *Let $G = (V, E)$ be a d-regular (k, φ, ϵ)-clusterable graph. For every $i \in [k]$, let $G\{C_i\}$ denote the induced subgraph on C_i with added self loops so that the degrees in $G\{C_i\}$ and G are the same, and let T_i^* denote the tree with optimum Dasgupta cost for $G\{C_i\}$. Then procedure* TotalClustersCost *(Algorithm 6) returns a value such that:*

$$\sum_{i\in[k]} COST_{G\{C_i\}}(T_i^*) \leq \textsc{TotalClustersCost}(G) \leq O\left(\frac{1}{\varphi^5}\right) \cdot \sum_{i\in[k]} COST_{G\{C_i\}}(T_i^*).$$

For completeness, we restate Theorem 2.3 from [MS21] and the algorithm for computing $\mathcal{T}_{\text{deg}}$ from [MS21]. However, we don't explicitly construct $\mathcal{T}_{\text{deg}}$.

Theorem 2.3 (Theorem 3 in [MS21])**.** *Given any graph $G = (V, E, w)$ with inner-conductance φ as input, Algorithm 2 runs in $O(m + n \log n)$ time, and returns an HC tree $\mathcal{T}_{\text{deg}}(G)$ that satisfies $COST_G(\mathcal{T}_{\text{deg}}(G)) = O(1/\varphi^4) \cdot OPT_G$.*

Algorithm 2 HCwithDegrees($G\{V\}$) [MS21]

Input: $G = (V, E, w)$ with the ordered vertices such that $d_{v_1} \geq \ldots \geq d_{v_{|V|}}$
Output: An HC tree $\mathcal{T}_{\text{deg}}(G)$
if $|V| = 1$ **then**
  return the single vertex V as the tree
else
  $i_{\max} := \lfloor \log_2 |V| - 1 \rfloor$; $r := 2^{i_{\max}}$; $A := \{v_1, \ldots, v_r\}$; $B := V \backslash A$
  Let $\mathcal{T}_1 :=$ HCwithDegrees($G\{A\}$); $\mathcal{T}_2 :=$ HCwithDegrees($G\{B\}$)
  return $\mathcal{T}_{\text{deg}}$ with $\mathcal{T}_1$ and $\mathcal{T}_2$ as the two children
end if

The rest of the section is structured as follows. In Section B.1 we first prove that for every φ-expander G, the quantity $\sum_{x\in V} \text{rank}(x) \deg(x)$ approximates the cost of $\mathcal{T}_{\text{deg}}$ up to $O(1/\varphi)$ factor. In Section B.2, we show how to estimate the quantity $\sum_{x\in V} \text{rank}(x) \deg(x)$. Then, in Section B.3, we put everything together and complete the proofs of Theorem 2.2 and Theorem A.4. Finally, in Section B.4 we prove the optimality of our sampling complexity for a single expander.

B.1 Bound cost of an expander by $\sum \operatorname{rank}(x) \cdot \deg(x)$

Let $G = (V, E)$ be an arbitrary expander with vertices $x_1, x_2, \ldots x_n$ ordered such that $d_1 \geq d_2 \geq \ldots \geq d_n$, where $d_i = \deg(x_i)$. We denote by $\mathcal{T}_{\deg}$ the Dasgupta Tree returned by Algorithm 1 of [MS21]. Recall that this is a binary tree which is obtained by recursive applications of a merge procedure. The call at the root level to merge aggregates a left subtree with leaves $v_1, v_2, \cdots v_{n/2}$ and a right subtree which has remaining vertices as leaves. We would like to show the following two lemmas.

Lemma B.1. *Let $G = (V, E)$ be a graph with degree sequence $d_1 \geq d_2 \geq \ldots \geq d_n$ and expansion φ. We have*

$$COST_G(\mathcal{T}_{\deg}) \geq \Omega(\varphi) \sum_{i=1}^{n} i \cdot d_i.$$

Lemma B.2. *Let $G = (V, E)$ be a graph with degree sequence $d_1 \geq d_2 \geq \ldots \geq d_n$. We have*

$$COST_G(\mathcal{T}_{\deg}) \leq 2 \sum_{i=1}^{n} i \cdot d_i.$$

Note that Lemma B.2 does not require the graph to be an expander. Both Lemma B.1 and Lemma B.2 hold even if the graph has self-loops.

B.1.1 Lower bound on Dasgupta cost of an expander (Proof of Lemma B.1)

In this section, we prove Lemma B.1. We will need some notation. Order the vertices in decreasing order of degrees and let $H = \lfloor \log_2 n \rfloor$. For each $i \in \{0, 1, \ldots H\}$, define the i-th bucket as

$$L_i = \{j \in V : 2^i \leq j \leq 2^{i+1} - 1\}.$$

We also need another notation. Define $L_{\leq i} = \bigcup_{j \leq i} L_j$. We will prove the following two claims.

Claim 2. $\mathrm{COST}_G(\mathcal{T}_{\deg}) \geq \frac{1}{2} \cdot \sum_{i=0}^{H} |E(L_i, \overline{L_i})| \cdot |L_{\leq i}|$.

Claim 3. $\sum_{i=0}^{H} |E(L_i, \overline{L_i})| \cdot |L_{\leq i}| \geq \varphi \cdot \sum i \cdot d_i$.

Note that once these two claims are shown, Lemma B.1 follows as a corollary.

Proof. (Of Claim 2) Recall that the vertices of $\mathcal{T}_{\deg}$ are arranged in decreasing order. Also, recall $\mathrm{COST}_G(T) = \sum_{\{x,y\} \in E} |\textsc{leaves}(\mathcal{T}_{\deg}[\mathrm{LCA}(x, y)])|$.

We will lower bound $\mathrm{COST}_G(\mathcal{T}_{\deg})$ by considering contributions to Dasupta Objective from a subset of the edges. In particular, we sum only over edges between "prefix sets" in $\mathcal{T}_{\deg}$ to get

$$\mathrm{COST}_G(\mathcal{T}_{\deg}) \geq \sum_{i=1}^{H} |E(L_i, L_{\leq i-1})| \cdot |L_{\leq i}|. \tag{48}$$

The above expression peels off sets L_i one at a time and considers the contribution of edges in the set $E(L_i, L_{\leq i-1})$ which is at least $|L_{\leq i-1}| + |L_i| = |L_{\leq i}|$. We will show that this is at least half the target expression (i.e., half the right hand side in the claim above) $\sum_{i=0}^{H} |E(L_i, \overline{L_i})| \cdot |L_{\leq i}|$ which will finish the proof.

The two expressions differ in the contribution they charge to an edge. Fix some i and take an edge $e \in E(L_i, \overline{L_i})$. Denote the contribution of edge e in Equation (48) as $\textsc{contrib}(e)$ and denote the contribution of e to the target expression as $\textsc{target}(e)$. It suffices to show that for every $e \in \bigcup_{i=0}^{H} E(L_i, \overline{L_i})$, $\textsc{target}(e) \leq 2\textsc{contrib}(e)$.

Fix $0 \le i \le H$ and take an edge $e \in E(L_i, \overline{L_i})$. Let $j \neq i$ be the index such that $e \in E(L_i, L_j)$ We have $\textsc{contrib}(e) \ge \max\{|L_{\le i}|, |L_{\le j}|\}$ and $\textsc{target}(e) = |L_{\le j}| + |L_{\le i}| \le 2\max\{|L_{\le i}|, |L_{\le j}|\}$. This holds for every edge $e \in \bigcup_{i=0}^{H} E(L_i, \overline{L_i})$ and this finishes the proof. $\square$

Next, we prove Claim 3.

Proof. (Of Claim 3) Since the expansion of G is at least φ, we obtain

$$\begin{aligned}\sum_{i=0}^{H} |E(L_i, \overline{L_i})| \cdot |L_{\le i}| &\ge \sum_{i=0}^{H} \varphi \cdot \mathrm{vol}(L_i) \cdot |L_{\le i}| \\ &= \varphi \sum_{i=0}^{H} \left(\sum_{j \in L_i} d_j \right) |L_{\le i}|.\end{aligned}$$

Furthermore, note that for all $j \in L_i$, it holds that $j \le |L_{\le i}|$. Therefore, from the above, we get that

$$\begin{aligned}\sum_{i=0}^{H} |E(L_i, \overline{L_i})| \cdot |L_{\le i}| &\ge \varphi \sum_{i=0}^{H} \left(\sum_{j \in L_i} d_j \right) |L_{\le i}| \\ &\ge \varphi \sum_{i=0}^{H} \sum_{j \in L_i} d_j \cdot j \\ &= \varphi \sum_{j \in [n]} d_j \cdot j,\end{aligned}$$

which finishes the proof. $\square$

B.1.2 Upper bound on Dasgupta cost of an expander (Proof of Lemma B.2)

In this section, we prove Lemma B.2. Like the previous section, we do this by proving the following two claims.

Claim 4. $\mathrm{COST}_G(\mathcal{T}_{\deg}) \le \sum_{i=0}^{H} \mathrm{vol}(L_i) \cdot |L_{\le i}|$.

Claim 5. $\sum_{i=0}^{H} \mathrm{vol}(L_i) \cdot |L_{\le i}| \le 2 \sum i \cdot d_i$.

Lemma B.2 follows as a corollary.

Proof. (Of Lemma B.2) Immediate from Claim 4 and Claim 5. $\square$

Now, we will prove Claim 4 and Claim 5 in the rest of this section. We begin with the first claim.

Proof. (Of Claim 4) We want to show $\mathrm{COST}_G(\mathcal{T}_{\deg}) \le \cdot \sum_{i=0}^{H} \mathrm{vol}(L_i) \cdot |L_{\le i}|$. Write $Obj(e)$ to denote the contribution to Dasgupta Cost of the edge e in tree $\mathcal{T}_{\deg}$. Recall, for an edge e which is not a self-loop, $Obj(e)$ equals the number of leaves in the subtree rooted at the LCA of the endpoints of the edge. Denote by $\textsc{target}(e)$ the contribution of edge e to the objective in the right hand side of this expression. If $e = (u, u)$ is a self-loop, then e does not contribute to the Dasgupta cost, and therefore $Obj(e) = 0 \le \textsc{target}(e)$. So consider an edge $e = (u, v)$ such that $u \neq v$ with $u \in L_i$, $v \in L_j$ where $i \le j$. The edge e is considered in the above sum at indices i and j. The contribution of e to the target objective is given as

$$\textsc{target}(e) = |L_{\le i}| + |L_{\le j}|.$$

On the other hand, from Algrorithm 2 and by definition of L_i's, we have $Obj(e) \le |L_{\le i}| + |L_{\le j}| = \textsc{target}(e)$. This holds for every edge and therefore

$$\mathrm{COST}_G(\mathcal{T}_{\mathrm{deg}}) = \sum_{e \in E(G)} Obj(e) \le \sum_{e \in E(G)} \textsc{target}(e) = \sum_{i=0}^{H} \mathrm{vol}(L_i) \cdot |L_{\le i}|.$$

□

Finally, we prove Claim 5 to wrap up.

Proof. (Of Claim 5) Note that $|L_{\le i}| \le 2^{i+1}$. We have,

$$\begin{aligned} \sum_{i=0}^{H} \mathrm{vol}(L_i) \cdot |L_{\le i}| &\le \sum_{i=0}^{H} \left(\sum_{j \in L_i} d_j \right) \cdot |L_{\le i}| \\ &\le \sum_{i=0}^{H} \left(\sum_{j \in L_i} d_j \right) \cdot 2^{i+1}. \end{aligned}$$

We want to upper-bound the last expression above. Note that this expression is of the form $\sum_{j \in [n]} \alpha_j \cdot d_j$ where $\alpha_j = 2^{i+1}$ if $j \in L_i$. However, for any $j \in L_i$, note that $j \ge 2^i = \frac{1}{2}\alpha_j$. This means that

$$\sum_{j \in [n]} \alpha_j \cdot d_j \le 2 \sum_{j \in [n]} j \cdot d_j$$

as desired. □

B.2 Estimating $\sum \mathrm{rank}(x) \cdot \deg(x)$

In this section, we prove Lemma B.3, which asserts that we can estimate $\sum \mathrm{rank}(x) \cdot \deg(x)$ using $O^*\left(n^{1/3}\right)$ samples.

Lemma B.3. *Let $G = (V, E)$ be a φ-expander (possibly with self-loops). There exists an estimator v using $O^*\left(n^{1/3}\right)$ samples, such that with probability at least $1 - n^{-100}$,*

$$\sum_{x \in V} \mathrm{rank}(x) \deg(x) \le v \le O(1) \cdot \sum_{x \in V} \mathrm{rank}(x) \deg(x).$$

Partition the vertices into buckets as follows: For $d = 2^0, 2^1, \cdots, 2^{\log(n/\varphi)}$, let $B_d := \{x \in V : d \le \deg(x) < 2d\}$. We will refer to B_d as the *degree class* of d. Let $n_d := |B_d|$ denote the size of the degree class, and let r_d denote the highest rank in B_d. Note that r_d is the number of vertices in G that have degree at least d, so we have $r_d = \sum_{t \ge d} n_t$.

Sometimes we will write $B_{\ge d}$ and $B_{<d}$ to denote $\cup_{t \ge d} B_d$ and $\cup_{t<d} B_d$, respectively.

Note that there are at most $\log(n/\varphi)$ different degree classes, since each vertex can have at most $n(1/\varphi - 1)$ self-loops.

The vertices in B_d have ranks $r_d, r_d - 1, \ldots, r_d - n_d + 1$ and degrees in $[d, 2d]$, which gives the bounds

$$\frac{d}{2} \cdot n_d \cdot r_d \le \sum_{i=r_d-n_d+1}^{r_d} i \cdot d \le \sum_{x \in B_d} \mathrm{rank}(x) \cdot \deg(x) \le \sum_{i=r_d-n_d+1}^{r_d} i \cdot 2d \le 2d \cdot n_d \cdot r_d. \tag{49}$$

Thus, our goal will be to efficiently approximate the quantities $r_d \cdot n_d$.

We start by proving the following technical lemma, which shows that there exists a degree class B_d that contains a large fraction of the degree mass, and that satisfies $d \le O\left(\frac{\log(n/\varphi)}{\varphi^2}\right) \cdot n_d$.

Lemma B.4. *There exists a degree class d such that $n_d \cdot d \ge \frac{m\cdot\varphi}{4\log(n/\varphi)}$ and $d \le \frac{16}{\varphi^2}\log(n/\varphi) \cdot n_d$.*

Proof. Let $m' \ge \varphi \cdot m$ denote the number of non-self-loop edges, and $\deg'(\cdot)$ denote the degrees discounting self-loops. Orient the edges from high degree to low degree (break ties arbitrarily within any degree class). That way, we have $m' = \sum_{x\in V} \deg'_{in}(x)$. Say that a degree class B_d is *heavy* if $n_d \cdot d \ge \frac{m'}{4\log(n/\varphi)}$, and say that it is *light* otherwise. Moreover, call a degree class B_d *good* if $\sum_{x\in B_d} \deg'_{in}(x) \ge \frac{n_d\cdot d\cdot\varphi}{2}$, and *bad* otherwise. There must exist a good heavy class, since otherwise

$$\begin{aligned}
m' &= \sum_{B_d : B_d \text{ is light}} \sum_{x\in B_d} \deg'_{in}(x) + \sum_{B_d : B_d \text{ is heavy}} \sum_{x\in B_d} \deg'_{in}(x) \\
&\le \sum_{B_d : B_d \text{ is light}} 2d\cdot n_d + \sum_{B_d : B_d \text{ is heavy}} \sum_{x\in B_d} \deg'_{in}(x) \\
&< \log(n/\varphi)\frac{m'}{2\log(n/\varphi)} + \sum_{B_d : B_d \text{ is heavy}} \sum_{x\in B_d} \deg'_{in}(x), && \text{by definition of the light classes} \\
&< \frac{m'}{2} + \sum_{B_d : B_d \text{ is heavy}} \frac{n_d\cdot d\cdot\varphi}{2}, && \text{assuming that all heavy classes are bad} \\
&< \frac{m'}{2} + \frac{m\cdot\varphi}{2}, \\
&\le m', && \text{since } m\cdot\varphi \le m' \text{ in a } \varphi\text{-expander}
\end{aligned}$$

which is a contradiction. Thus, there exists a degree class d that is both heavy and good, i.e.

$$n_d \cdot d \ge \frac{m'}{4\log(n/\varphi)} \ge \frac{m\cdot\varphi}{4\log(n/\varphi)} \qquad \text{(heavy)}$$

and

$$\sum_{x\in B_d} \deg'_{in}(x) \ge \frac{n_d\cdot d\cdot\varphi}{2} \qquad \text{(good)}.$$

Let d be such a degree class. To establish the lemma, it remains to show that $d \le \frac{16}{\varphi^2}\log(n/\varphi)\cdot n_d$.

First, observe that $r_d \cdot d \le \sum_{x:\deg(x)\ge d} \deg(x) \le 2m \le \frac{8\log(n/\varphi)\cdot n_d\cdot d}{\varphi}$, where the last inequality follows from the assumption that d is heavy. Therefore, we have

$$r_d \le \frac{8\log(n/\varphi)\cdot n_d}{\varphi}.$$

Now, consider the number of non-self-loop edges between B_d and $B_{\ge d}$. Recall that we orient the edges from high degree to low degree, so that the number of non-self-loop edges between B_d and $B_{\ge d}$ is equal to $\sum_{x\in B_d} \deg'_{in}(x) \ge \frac{n_d\cdot d\cdot\varphi}{2}$. On the other hand, the number of non-self-loop edges between B_d and $B_{\ge d}$ can be at most $|B_{\ge d}|\cdot|B_d| = \left(\sum_{t\ge d} n_t\right)\cdot n_d = r_d\cdot n_d \le \frac{8\log(n/\varphi)\cdot n_d^2}{\varphi}$. Combining, we obtain

$$\frac{n_d\cdot d\cdot\varphi}{2} \le \frac{8\log(n/\varphi)\cdot n_d^2}{\varphi},$$

which gives $d \le \frac{16}{\varphi^2}\log(n/\varphi)\cdot n_d$, as required. ∎

We now introduce the definition of a Dasgupta cost heavy degree class, i.e. a class that contributes a significant fraction of the Dasgupta cost.

Definition B.5. Say that a degree class d is *α-Dasgupta Cost Heavy*, or just *α-DC-heavy*, if

$$\sum_{x \in B_d} \operatorname{rank}(x) \deg(x) \geq 2\alpha \sum_{x \in V} \operatorname{rank}(x) \deg(x).$$

The following claim shows that for α-DC-heavy classes, we can use n_d as a proxy for r_d.

Claim 6. If d is α-DC-heavy, then $n_d \geq \frac{1}{2}\alpha \cdot r_d$.

Proof. By Equation (49), we have

$$2d \cdot n_d \cdot r_d \geq \sum_{x \in B_d} \operatorname{rank}(x) \deg(x) \geq 2\alpha \sum_{x \in V} \operatorname{rank}(x) \deg(x) \geq 2\alpha \sum_{i=1}^{r_d} i \cdot d \geq 2\alpha \cdot \frac{r_d^2}{2} \cdot d,$$

which rearranges to

$$n_d \geq \frac{1}{2}\alpha \cdot r_d.$$

□

The next lemma is the key result underlying our bound on the number of samples required. It shows that a degree class that contributes a nontrivial amount to Dasgupta cost of the graph must contain at least a $\approx n^{-1/3}$ fraction of edges of the graph:

Lemma B.6. *If a degree class t is α-DC-heavy, then $n_t \cdot t \cdot n^{1/3} \cdot \frac{\log^2 n}{\alpha^2 \varphi^2} \geq \Omega(m)$.*

Proof. We will apply Lemma B.4, which asserts that there exists a degree class B_d that contains a large fraction of the degree mass, and that satisfies $d \leq O\left(\frac{\log^2(n/\varphi)}{\varphi}\right) \cdot n_d$. We will then use the degree class B_d as a reference, and show that the degree mass of B_t cannot be much smaller.

More formally,we have the following optimization problem over the variables t, d, n_t, n_d:

$$\begin{aligned}
\max_{t,d,n_t,n_d} \ & \frac{n_d \cdot d}{n_t \cdot t} \\
\text{such that} \\
& n_t^2 \cdot t \geq \alpha^2 \cdot n_d^2 \cdot d \\
& d \leq \frac{16 \log(n/\varphi)}{\varphi^2} \cdot n_d \\
& n_t, n_d \leq n \\
& n_t, d, t \geq 1 \\
& n_d \geq 0.
\end{aligned}$$

First, we will show that the optimal value is $\approx n^{1/3}$.

Claim 7. The above optimization problem has a finite optimal value. Furthermore, if t, d, n_d, n_t is an optimal solution, then the first two constraints are tight and $t = 1$.

Proof. First, we show that the optimization problem has a finite optimal value. Observe that adding the constraint $t \leq n^4$ does not change the optimal objective value, since increasing the value of t can only harm the objective, and any feasible choice of n_d, n_t, d remains feasible after adding the constraint. Now, with the additional constraint, we have that the feasible region is bounded (since $1 \leq n_t \leq n, 1 \leq t \leq n^4$, $0 \leq n_d \leq n$, and $1 \leq d \leq \frac{16\log(n/\varphi)}{\varphi^2} \cdot n_d$), closed, and non-empty (since taking for instance $d = 1, n_d = 1$, $t = n_t = n$ is a feasible solution). So the optimization problem has a finite value and attains its maximum value.

Next, we show that if t, d, n_d, n_t is an optimal solution, then the first two constraints are tight and $t = 1$. Suppose that the first constraint is loose, i.e. that $n_t^2 \cdot t > \alpha^2 \cdot n_d^2 \cdot d$. Clearly, we can't have $n_t = t = 1$ and $n_d = n$ (otherwise the first constraint would not be satisfied), so it is possible to either decrease t, decrease n_t or increase n_d. Either of these options gives a higher objective value, which contradicts the optimality of the given solution.

Suppose instead that the second constraint is loose. Let $d' = \gamma^2 \cdot d, n_d' = \frac{n_d}{\gamma}$ for some sufficiently small $\gamma > 1$. Then t, n_t, d', n_d' is a feasible solution, but $n_d' \cdot d' = \gamma \cdot n_d \cdot d > n_d \cdot d$, so this gives a higher objective value, which is a contradiction.

Finally, suppose that $t > 1$, and that the first two constraints are tight. If $n_t < n$, then let $t' = 1$, $n_t' = n_t\sqrt{t}$. Then t', n_t', d, n_d is a feasible solution, but $n_t' \cdot t' = n_t\sqrt{t} < n_t \cdot t$, so this gives a higher objective value, contradiction. If instead $n_t = n$, then we have $n_t^2 \cdot t > n^2$ (by the assumption that $t > 1$). On the other hand, we have $n_t^2 \cdot t = \alpha^2 \cdot n_d^2 \cdot d = \frac{16\log(n/\varphi^2)}{\varphi^2} n_d^3$ (by the assumption that the first two constraints are tight), from which we deduce $n_d, d > 1$. In particular, there exists $\gamma > 1$ such that $t' = t \cdot \gamma^{-1}$, $n_d' = n_d \cdot \gamma^{-1/3}$, $d' = d \cdot \gamma^{-1/3}$ is a feasible solution. But this solution has objective value $\frac{n_d \cdot d}{n_t \cdot t} \cdot \gamma^{1/3} > \frac{n_d \cdot d}{n_t \cdot t}$, contradiction. □

Now let t, d, n_d, n_t be an optimal solution. It follows by Claim 7 that

$$\begin{aligned}
n^2 &\geq n_t^2 \cdot t && \text{since } n_t \leq n \text{ and } t = 1\\
&= \alpha^2 \cdot n_d^2 \cdot d && \text{since the first constraint is tight}\\
&= \frac{\alpha^2 \cdot \varphi^4}{16^2 \log^2(n/\varphi)} \cdot d^3 && \text{since the second constraint is tight,}
\end{aligned}$$

which rearranges to

$$d \leq \left(\frac{O(n\log(n/\varphi))}{\varphi^2 \cdot \alpha}\right)^{2/3}$$

Furthermore, since $t = 1$, we have

$$n_t \cdot t = n_t = \sqrt{n_t^2 \cdot t} = \alpha \cdot \sqrt{n_d^2 \cdot d}.$$

Thus,

$$\frac{n_d \cdot d}{n_t \cdot t} = \frac{n_d \cdot d}{\alpha \cdot n_d \cdot d^{1/2}} = \frac{d^{1/2}}{\alpha} \leq \frac{1}{\alpha}\left(\frac{O(n\log(n/\varphi))}{\varphi^2 \cdot \alpha}\right)^{1/3}$$

This shows that the optimal solution to the optimization problem has value $O(n^{1/3}\varphi^{-2/3}\alpha^{-4/3}\log^{1/3}(n/\varphi))$.

Now let B_d be a degree class such that $n_d \cdot d \geq \frac{m \cdot \varphi}{4\log n}$ and $d \leq \frac{16}{\varphi^2}\log(n/\varphi) \cdot n_d$ (exists by Lemma

B.4), and let t be any α-DC-heavy degree class. By Claim 6, we have

$$\begin{aligned}
n_t^2 \cdot t &\geq \frac{1}{2} \cdot \alpha \cdot r_t \cdot n_t \cdot t && \text{by Claim 6,} \\
&\geq \alpha^2 \sum_{x \in V} \deg(x)\mathrm{rank}(x) && \text{since } t \text{ is } \alpha\text{-DC heavy} \\
&\geq \alpha^2 \sum_{x \in B_d} \deg(x)\mathrm{rank}(x) \\
&\geq \alpha^2 \cdot n_d \cdot r_d \cdot d \\
&\geq \alpha^2 \cdot n_d^2 \cdot d
\end{aligned}$$

Thus, t, n_t, d, n_d is a feasible solution to the optimization problem, and in particular

$$n_t \cdot t \cdot O(n^{1/3}\varphi^{-2/3}\alpha^{-4/3}\log^{1/3}(n/\varphi)) \geq n_d \cdot d \geq \frac{m \cdot \varphi}{4\log(n/\varphi)},$$

which gives the result. $\square$

We can now obtain a good estimator for the size of each bucket.

Lemma B.7. *Given α, there exists an estimator $\hat{n}_t$ using $O\left(n^{1/3} \cdot \frac{\log^3(n/\varphi)}{\alpha^2\varphi^2}\right)$ samples, such that with probability at least $1 - \frac{1}{2}n^{-101}$, the following holds:*

1. *For every degree class t, it holds that $\hat{n}_t \leq 6n_t$*
2. *If t is an α-DC-heavy class, then $\hat{n}_t \geq \frac{1}{2}n_t$.*

Proof. Let c be the constant in front of m in Lemma B.6, and let $s = 16c \cdot n^{1/3} \cdot \frac{\log^3(n/\varphi)}{\alpha^2\varphi^2}$. Let S be a set of s vertices sampled independently at random with probability proportional to their degree. For each degree class t, let $X_t = \frac{2m}{s \cdot t}|\{x \in S : x \in B_t\}|$. Then $\mathbb{E}[X_t] = \frac{1}{t}\sum_{x \in B_t}\deg(x) \leq \frac{2t \cdot n_t}{t} = 2n_t$. By Markov's inequality, $\Pr[X_t > 6n_t] \leq \frac{1}{3}$. Repeat $O(\log n)$ times and let $\hat{n}_t$ be the median, so that $\Pr[\hat{n}_t \geq 6n_t] \leq \frac{1}{4}n^{-102}$.

Now, suppose that t is α-DC heavy. We have that $\frac{t \cdot s}{2m} \cdot X_t$ is a sum of independent $\{0,1\}$ random variables, with $\mathbb{E}[\frac{t \cdot s}{2m}X_t] = \frac{s}{2m}\sum_{x \in B_t}\deg(x) \geq \frac{s}{2m}t \cdot n_t \geq 16$. Here the last inequality holds by Lemma B.6 and the choice of s. By Chernoff bounds, we obtain that $\Pr[\frac{t \cdot s}{2m}X_t \leq \frac{1}{2} \cdot \frac{t \cdot s}{2m} \cdot n_t] \leq \exp(-2) \leq \frac{1}{3}$. Since $\hat{n}_t$ is obtained from X_t by repeating $O(\log n)$ times and taking the median, we get that $\Pr[\hat{n}_t < \frac{1}{2}n_t] \leq \frac{1}{4}n^{-102}$.

Taking the union bound over all t gives the result. $\square$

Similarly, we can obtain an estimator for the highest rank in each bucket.

Lemma B.8. *Given α, there exists an estimator $\hat{r}_t$ using $O\left(n^{1/3} \cdot \frac{\log^5(n/\varphi)}{\alpha^2\varphi^2}\right)$ samples, such that with probability at least $1 - \frac{1}{2}n^{-101}$, the following holds:*

1. *For every degree class t, it holds that $\hat{r}_t \leq 6r_t$*
2. *If t is an α-DC-heavy class, then $\hat{r}_t \geq \frac{1}{4}r_t$.*

Proof. Let $\alpha' = \frac{\alpha}{8\log^2(n/\varphi)}$, and let $\hat{n}$ be the estimator from Lemma B.7 with parameter α'. For each degree class d, let $\hat{r}_d = \sum_{t \geq d} \hat{n}_t$. Condition on the success of $\hat{n}$ (which happens with probability at least $1 - \frac{1}{2}n^{-101}$). Then Property 1 follow immediately from Lemma B.7. It remains to prove that Property 2 holds. Fix an α-DC-heavy class t. Say that a degree class $t' > t$ is *heavy* if

$$n_{t'} \geq \frac{r_t}{2\log(n/\varphi)},$$

and otherwise say that it is *light*. First, we show that if t' is heavy, then t' is $\frac{\alpha}{8\log^2(n/\varphi)}$-DC-heavy. Indeed, if t' is heavy, then

$$\begin{aligned}
2\sum_{x \in B_{t'}} \operatorname{rank}(x)\deg(x) &\geq t' \cdot n_{t'} \cdot r_{t'} && \text{by Equation (49)} \\
&\geq \frac{2t}{4\log^2(n/\varphi)} r_t^2 && \text{because } r_t' \geq n_t' \geq \frac{r_t}{2\log(n/\varphi)} \text{ and } t' \geq 2t \\
&\geq \frac{1}{4\log^2(n/\varphi)} \sum_{x \in B_t} \operatorname{rank}(x)\deg(x) && \text{by Equation (49), since } r_t \geq n_t \\
&\geq \frac{2\alpha}{4\log^2(n/\varphi)} \sum_{x \in V} \operatorname{rank}(x)\deg(x) && \text{by the assumption that } t \text{ is } \alpha\text{-DC heavy.}
\end{aligned}$$

So if t' is heavy, then it is $\frac{\alpha}{8\log^2(n/\varphi)}$-DC heavy, and in particular, by Lemma B.7, $\hat{n}_{t'} \geq \frac{1}{2}n_t$. We now have

$$\begin{aligned}
\hat{r}_t &= \sum_{t' \geq t : t' \text{ is light}} \hat{n}_{t'} + \sum_{t' \geq t : t' \text{ is heavy}} \hat{n}_{t'} \\
&\geq \frac{1}{2} \sum_{t' \geq t : t' \text{ is heavy}} n_{t'} + \sum_{t' \geq t : t' \text{ is light}} \hat{n}_{t'} && \text{by Lemma B.7} \\
&= \frac{1}{2} \sum_{t' \geq t} n_t + \sum_{t' \geq t : t' \text{ is light}} \left(\hat{n}_{t'} - \frac{1}{2} n_{t'} \right) \\
&\geq \frac{1}{2} \sum_{t' \geq t} n_t - \frac{1}{2} \sum_{t' \geq t : t' \text{ is light}} n_{t'} \\
&\geq \frac{1}{2} \sum_{t' \geq t} n_t - \frac{\log(n/\varphi)}{4\log(n/\varphi)} r_t && \text{by definition of light classes} \\
&= \frac{1}{4} r_t.
\end{aligned}$$

□

We are now ready to prove Lemma B.3.

Proof of Lemma B.3. Let $\alpha = \frac{1}{4\log(n/\varphi)}$. Let $\hat{n}$ be the estimator from Lemma B.7 with parameter α and let $\hat{r}$ be the rank estimator from Lemma B.8 with parameter α. Let

$$v = 32 \sum_d \hat{r}_d \cdot \hat{n}_d \cdot d.$$

Condition on the success of the estimators $\hat{n}$ and $\hat{r}$ (which happens with probability at least $1-n^{-101}$). Then, for each degree class d, we have

$$\begin{aligned} d \cdot \hat{r}_d \cdot \hat{n}_d &\le 36 d \cdot r_d \cdot n_d && \text{by Lemma B.7 and Lemma B.8} \\ &\le 72 \sum_{x \in B_d} \deg(x) \mathrm{rank}(x), && \text{by Equation (49).} \end{aligned}$$

Summing over all degree classes d, we have

$$v = 32 \sum_d \hat{r}_d \cdot \hat{n}_d \cdot d \le 32 \cdot 72 \sum_{x \in V} \mathrm{rank}(x) \deg(x),$$

which gives the upper bound. It remains to prove the lower-bound. Say that a degree class is *heavy* if it is $\frac{1}{4\log(n/\varphi)}$-DC heavy, and say that it is *light* otherwise. We have

$$\begin{aligned} \sum_{x \in V} \mathrm{rank}(x) \deg(x) &= \sum_d \sum_{x \in B_d} \mathrm{rank}(x) \deg(x) \\ &= \sum_{d: B_d \text{ is light}} \sum_{x \in B_d} \mathrm{rank}(x) \deg(x) + \sum_{d: B_d \text{ is heavy}} \sum_{x \in B_d} \mathrm{rank}(x) \deg(x) \\ &\le \frac{\log(n/\varphi)}{2\log(n/\varphi)} \sum_{x \in V} \mathrm{rank}(x) \deg(x) + \sum_{d: B_d \text{ is heavy}} \sum_{x \in B_d} \mathrm{rank}(x) \deg(x) \\ &\le \frac{1}{2} \sum_{x \in V} \mathrm{rank}(x) \deg(x) + \sum_{d: B_d \text{ is heavy}} 2d \cdot n_d \cdot r_d && \text{by Equation (49)} \\ &\le \frac{1}{2} \sum_{x \in V} \mathrm{rank}(x) \deg(x) + 16 \sum_{d: B_d \text{ is heavy}} d \cdot \hat{n}_d \cdot \hat{r}_d && \text{by Lemmas B.7 and B.8} \\ &\le \frac{1}{2} \sum_{x \in V} \mathrm{rank}(x) \deg(x) + \frac{1}{2} v. \end{aligned}$$

Rearranging, we obtain the lower-bound. □

B.3 Correctness of ClusterCost and TotalClustersCost (Proof of Theorem 2.2 and Theorem A.4)

In this section we put everything together to prove Theorem 2.2. Furthermore, we also present the procedure TotalClustersCost for approximating the contribution of the clusters to the Dasgupta cost to a d-regular (k, φ, ϵ)-clusterable graph, and we prove its guarantee (Theorem A.4).

Theorem 2.2. *Let $G = (V, E)$ be a φ-expander (possibly with self-loops). Let T^* denote the tree with optimum Dasgupta cost for G. Then procedure ClusterCost (Algorithm 3), uses $O^*\left(n^{1/3}\right)$ seed queries and with probability $1 - n^{-101}$ returns a value such that:*

$$COST(T^*) \le \textsc{ClusterCost}(G) \le O\left(\frac{1}{\varphi^5}\right) \cdot COST(T^*).$$

Proof of Theorem 2.2. Follows immediately from Lemma B.3, Lemma B.1, Lemma B.2 and Theorem 2.3. □

Next, we present the procedure TotalClustersCost for approximating the contribution of the clusters to the Dasgupta cost to a d-regular (k, φ, ϵ)-clusterable graph. A natural approach is to run the ClusterCost procedure on each cluster and sum the output. This would use $\approx n^{1/3}$ seeds. However, since we assume that the graph is d-regular, we can do something much simpler.

By virtue of Lemma B.1 and Lemma B.2, we want to approximate the quantity $\sum \text{rank}(v) \cdot \deg(v)$ for each cluster C_i. However, since the graph is d-regular, we get

$$\sum_{v \in C_i} \text{rank}_{G\{C_i\}}(v) \deg_{G\{C_i\}}(v) = \sum_{v \in C_i} \text{rank}_{G\{C_i\}}(v) \cdot d = \sum_{j=1}^{|C_i|} j \cdot d \approx |C_i|^2 \cdot d.$$

Furthermore, by the assumption that $\max_{i,j} |C_i|/|C_j| = O(1)$, we have $|C_i| = O(n/k)$ for all i, so the total contribution from the cluster simplifies further as

$$\sum_{i \in [k]} \sum_{v \in C_i} \text{rank}_{G\{C_i\}}(v) \deg_{G\{C_i\}}(v) \approx \sum_{i \in [k]} |C_i|^2 \cdot d \approx k \cdot \frac{n^2}{k^2} \cdot d = \frac{d \cdot n^2}{k}.$$

Motivated by this, our procedure TotalClustersCost below simply outputs the number $\frac{d \cdot n^2}{k}$.

Algorithm 6 TotalClustersCost(G)

1: **return** $O\left(\frac{d \cdot n^2}{k}\right)$

Remark B.9. Algorithm 6 uses the approximation $|C_i| \approx n/k$ for all i, which introduces a dependence on the parameter $\eta := \max_{i,j} \frac{|C_i|}{|C_j|}$ in the approximation guarantee. Since we assume $\eta = O(1)$, this is enough for our purposes. However, if desired, one can easily remove this dependence by approximating each $|C_i|$ more accurately via sampling vertices with their cluster labels.

Theorem A.4. *Let $G = (V, E)$ be a d-regular (k, φ, ϵ)-clusterable graph. For every $i \in [k]$, let $G\{C_i\}$ denote the induced subgraph on C_i with added self loops so that the degrees in $G\{C_i\}$ and G are the same, and let T_i^* denote the tree with optimum Dasgupta cost for $G\{C_i\}$. Then procedure TotalClustersCost (Algorithm 6) returns a value such that:*

$$\sum_{i \in [k]} COST_{G\{C_i\}}(T_i^*) \leq \textit{TotalClustersCost}(G) \leq O\left(\frac{1}{\varphi^5}\right) \cdot \sum_{i \in [k]} COST_{G\{C_i\}}(T_i^*).$$

Proof. Let $v = 3 \cdot \eta^2 \cdot \frac{d \cdot n^2}{k}$ be the value output by Algorithm 6, where $\eta = \max_{i,j} |C_i|/|C_j| = O(1)$.

For every $i \in [k]$, let $\mathcal{T}_i$ denote the output of Algorithm 2 on input $G\{C_i\}$. We have

$$\begin{aligned}
\text{COST}_{G\{C_i\}}(T_i^*) &\leq \text{COST}_{G\{C_i\}}(\mathcal{T}_i) && \text{by optimality of } T_i^* \\
&\leq 2 \sum_{v \in C_i} \text{rank}_{G\{C_i\}}(v) \cdot \deg_{G\{C_i\}}(v) && \text{by Lemma B.2} \\
&= 2 \sum_{j=1}^{|C_i|} j \cdot d && \text{since } \deg_{G\{C_i\}}(v) = \deg_G(v) = d \text{ for all } v \in C_i \\
&= 2 \cdot \frac{|C_i|(|C_i| + 1)}{2} \cdot d \\
&\leq 3\eta^2 \frac{n^2}{k^2} \cdot d && \text{since } |C_i| \leq \eta \cdot \frac{n}{k}.
\end{aligned}$$

Summing over $i \in [k]$ and recalling that $\textsc{TotalClustersCost}(G) = 3\eta^2 \frac{n^2}{k} \cdot d$, yields the lower bound. Next, we prove the upper bound. We have

$$\begin{aligned}
\mathrm{COST}_{G\{C_i\}}(T_i^*) &\geq \Omega(\varphi^4)\ \mathrm{COST}_{G\{C_i\}}(\mathcal{T}_i) && \text{by Theorem 2.3} \\
&\geq \Omega(\varphi^5) \sum_{v \in C_i} \mathrm{rank}_{G\{C_i\}}(v) \cdot \deg_{G\{C_i\}}(v) && \text{by Lemma B.2} \\
&= \Omega(\varphi^5) \sum_{j=1}^{|C_i|} j \cdot d && \text{since } \deg_{G\{C_i\}}(v) = \deg_G(v) = d \text{ for all } v \in C_i \\
&= \Omega(\varphi^5) \cdot \frac{|C_i|(|C_i|+1)}{2} \cdot d \\
&= \Omega(\varphi^5) \cdot \frac{n^2}{k^2} \cdot d && \text{since } |C_i| \geq \frac{n}{k \cdot \eta} = \Omega\left(\frac{n}{k}\right)
\end{aligned}$$

Summing over $i \in [k]$ and recalling that $\textsc{TotalClustersCost}(G) = 3\eta^2 \frac{n^2}{k} \cdot d = O\left(\frac{n^2}{k} \cdot d\right)$, yields the upper bound. $\square$

We need the following Observation in the proof of Theorem 2.1.

Observation 2. It holds that $\textsc{TotalClustersCost}(G) \geq \Omega\left(\frac{n^2}{k}\right)$. This is because the value output by Algorithm 6 is given by $v = 3 \cdot \eta^2 \cdot \frac{d \cdot n^2}{k} = \Omega\left(\frac{n^2}{k}\right)$, since $d, \eta \geq 1$.

B.4 Lower bound on the necessary number of seeds (Proof of Theorem 2.4)

In this subsection, we prove Theorem 2.4 from Section 2.1, which shows that the query complexity of $\textsc{ClusterCost}$ is tight.

Theorem 2.4. *For every positive constant $\alpha > 1$ and n sufficiently large, there exists a pair of expanders G and G' such that $\sum_{i=1}^{n} i \cdot d_i \leq n^2$, $\sum_{i=1}^{n} i \cdot d_i' \geq \alpha n^2$ and at least $\Omega(n^{1/3})$ vertices need to be queried in order to have probability above $2/3$ of distinguishing between them (where $d_1 \geq ... \geq d_n \geq 1$ is the degree sequence in G and $d_1' \geq ... \geq d_n' \geq 1$ is the degree sequence in G').*

Proof. We will construct the two graphs on the same vertex set V. Pick a set $C \subseteq V$ of size $\frac{n^{2/3}}{2}$. Let $b = \frac{|V \setminus C|}{|C|} = 2n^{1/3} - 1$. Construct the graph G as follows:

- C forms a clique
- Add a perfect b-matching between C and $V \setminus C$.

Then every vertex in C has degree $\frac{n^{2/3}}{2} - 1 + b$, and every vertex in $V \setminus C$ has degree 1. Therefore,

$$\sum_{i=1}^{n} i \cdot d_i \leq \sum_{i=1}^{\frac{n^{2/3}}{2}} i \cdot (n^{2/3} + b) + \sum_{i=\frac{n^{2/3}}{2}+1}^{n} i \leq \frac{1}{2} n^{2/3} \cdot (n^{2/3})^2 + n^2/2 \leq n^2.$$

Construct G' as follows:

- C forms a clique
- Add a $(2\alpha \cdot b, 2\alpha)$-regular bipartite graph between C and $V \setminus C$.

- For each vertex in C, delete $b(2\alpha - 1)$ of its edges internal edges in C.

Now every vertex in C has degree $\frac{n^{2/3}}{2} - 1 + b$, but every vertex in $V \setminus C$ has degree 2α. Therefore,

$$\sum_{i=1}^{n} i \cdot d_i' \geq \sum_{i=1}^{\frac{n^{2/3}}{2}} i \cdot \frac{n^{2/3}}{2} + \sum_{i=\frac{n^{2/3}}{2}+1}^{n} i \cdot 2\alpha \geq \frac{n^{2/3}}{2} \cdot \left(\frac{n^{2/3}}{2}\right)^2 + 2\alpha \cdot \frac{n^2}{2} \geq \alpha n^2.$$

Every vertex in C has the same degree in G and G', so to distinguish between the two graphs, we need to query a vertex in $V \setminus C$.

We have $|E_G|, |E_{G'}| \geq \frac{|C|^2}{2} = \frac{n^{4/3}}{8}$. In G, the probability that a given query returns a vertex in $V \setminus C$, is $\frac{|V \setminus C|}{2|E_G|} \leq \frac{4n}{n^{4/3}} = 4n^{-1/3}$. Similarly, in G', the probability that a given query returns a vertex in $V \setminus C$, is $\frac{|V \setminus C| \cdot 2\alpha}{2|E_G'|} \leq \frac{8\alpha n}{n^{4/3}} = 8\alpha n^{-1/3}$.

Now suppose that the number of queries is at most $\frac{n^{1/3}}{8\alpha}$. If the true input graph is G, then with probability at least $(1 - n^{-1/3})^{\frac{n^{1/3}}{8\alpha}} \geq \frac{1}{3}$, we fail to query any vertices in $V \setminus C$. Similarly, if the true input graph graph is G', then with probability at least $(1 - 8\alpha n^{-1/3})^{\frac{n^{1/3}}{8\alpha}} \geq \frac{1}{3}$, we fail to query any vertices in $V \setminus C$. So with probability at least $\frac{1}{3}$, we fail to distinguish the graphs. □

C Correctness of WeightedDotProductOracle (Proof of Theorem A.3)

Obtaining the weighted dot product oracle: Recall that we aim to spectrally approximate $\mathcal{L}_H$ by $\widetilde{\mathcal{L}}$ with probability at least $1 - n^{-100}$ (as in Equation (4)) and this amounts to approximating all quadratic forms on $\mathcal{L}_H$ with quadratic forms on $\widetilde{\mathcal{L}}$. We achieve this by getting estimates for $\left\langle f_x, \Sigma_{[k]} f_y \right\rangle$ to be accurate with probability at least $1 - n^{-100k}$. The details are presented below.

Theorem A.3. *Let M denote the random walk matrix of G, and let $M = U\Sigma U^T$ denote the eigendecomposition of M. With probability at least $1 - n^{-100}$ over the initialization procedure (Algorithm 7) the following holds:*

With probability at least $1 - 3 \cdot n^{-100 \cdot k}$ for all $x, y \in V$ we have

$$|\left\langle f_x, \Sigma_{[k]} f_y \right\rangle_{apx} - \left\langle f_x, \Sigma_{[k]} f_y \right\rangle| \leq \frac{\xi}{nk^2},$$

where $\left\langle f_x, \Sigma_{[k]} f_y \right\rangle_{apx} =$ WeightedDotProductOracle$(G, x, y, \xi, \mathcal{D})$.

Moreover, the running time of the procedures InitializeWeightedDotProductOracle (Algorithm 7) and WeightedDotProductOracle (Algorithm 8) is $O^*\left(n^{1/2+O(\epsilon/\varphi^2)} \cdot \left(\frac{1}{\xi}\right)^{O(1)}\right)$.

Remark C.1. This result is similar to Theorem 2 in [GKL+21a]. The difference being Theorem 2 in [GKL+21a] approximates dot product between the embedding vectors $\langle f_x, f_y \rangle$. Here, we instead want to approximate the weighted dot product $\left\langle f_x, \Sigma_{[k]} f_y \right\rangle$.

We also need to set up some notation which is used in this section. Let $m \leq n$ be integers. For any matrix $A \in \mathbb{R}^{n \times m}$ with singular value decomposition (SVD) $A = Y\Gamma Z^T$ we assume $Y \in \mathbb{R}^{n \times n}$ and $Z \in \mathbb{R}^{m \times n}$ are orthogonal matrices and $\Gamma \in \mathbb{R}^{n \times n}$ is a diagonal matrix of singular values. Since Y and Z are orthogonal matrices, their columns form an orthonormal basis. For any integer $q \in [m]$

we denote $Y_{[q]} \in \mathbb{R}^{n\times q}$ as the first q columns of Y and $Y_{-[q]}$ to denote the matrix of the remaining columns of Y. We also denote $Z_{[q]}^T \in \mathbb{R}^{q\times n}$ as the first q rows of Z^T and $Z_{-[q]}^T$ to denote the matrix of the remaining rows of Z. Finally we denote $\Gamma_{[q]}^T \in \mathbb{R}^{q\times q}$ as the first q rows and columns of Γ and we use $\Gamma_{-[q]}$ as the last $n-q$ rows and columns of Γ. So for any $q \in [m]$ the span of $Y_{-[q]}$ is the orthogonal complement of the span of $Y_{[q]}$, also the span of $Z_{-[q]}$ is the orthogonal complement of the span of $Z_{[q]}$. Thus we can write $A = Y_{[q]}\Gamma_{[q]}Z_{[q]}^T + Y_{-[q]}\Gamma_{-[q]}Z_{-[q]}^T$.

Algorithm 7 InitializeWeightedDotProductOracle(G, ξ)

1: $t \leftarrow \frac{20\cdot\log n}{\varphi^2}$
2: $R_{\text{init}} \leftarrow n^{1/2+O(\epsilon/\varphi^2)} \cdot \left(\frac{k\cdot\log n}{\xi}\right)^{O(1)}$
3: $s \leftarrow n^{O(\epsilon/\varphi^2)} \cdot \left(\frac{k\cdot\log n}{\xi}\right)^{O(1)}$
4: $I_S \leftarrow s$ indices chosen independently and uniformly at random with replacement from $\{1,\dots,n\}$
5: $\widehat{Q} \leftarrow$ EstimateTransitionMatrix($G, I_S, R_{\text{init}}, t$) # $\widehat{Q}$ *has at most* $R_{init}\cdot s$ *non-zeros*
6: $\mathcal{G} \leftarrow$EstimateCollisionProbabilities($G, I_S, R_{\text{init}}, t$)
7: Compute eigendecomposition $\frac{n}{s}\cdot\mathcal{G} := \widehat{W}\widehat{\Sigma}\widehat{W}^T$ of $\frac{n}{s}\cdot\mathcal{G}$ # $\mathcal{G} \in \mathbb{R}^{s\times s}$
8: $\Psi \leftarrow \frac{n}{s}\cdot\widehat{W}_{[k]}\widehat{\Sigma}_{[k]}^{-2}\widehat{W}_{[k]}^T$ # $\Psi \in \mathbb{R}^{s\times s}$
9: **return** $\mathcal{D}_w := \{\Psi, \widehat{Q}\}$

Algorithm 8 WeightedDotProductOracle($G, x, y, \xi, \mathcal{D}$) # $\mathcal{D}_w := \{\Psi, \widehat{Q}\}$

1: $R_{\text{query}} \leftarrow n^{1/2+O(\epsilon/\varphi^2)} \cdot \left(\frac{k\cdot\log n}{\xi}\right)^{O(1)}$
2: $\widehat{m}_x \leftarrow$ RunRandomWalks($G, R_{\text{query}}, t+1, x$) # *unlike in [GKL*[+]*21a], walk length* $= t+1$
3: $\widehat{m}_y \leftarrow$ RunRandomWalks($G, R_{\text{query}}, t, y$)
4: **return** $\left\langle f_x, \Sigma_{[k]} f_y\right\rangle_{apx} := (\widehat{m}_x^T\widehat{Q})\Psi(\widehat{Q}^T\widehat{m}_y)$

We build up on a collection of tools from [GKL[+]21a]. First, we use Lemma 16 which shows that (k,φ,ϵ)-clusterable graphs, the outer products of the columns of the t-step random walk transition matrix has small spectral norm. This holds because the matrix power dominates by the first k eigenvectors and each of them has bounded infinity norm.

Lemma 16 (A higher success probability version of Lemma 23 from [GKL[+]21a] with improved estimation error). Let $k \geq 2$ be an integer, $\varphi \in (0,1)$ and $\epsilon \in (0,1)$. Let $G = (V,E)$ be a d-regular graph that admits a (k,φ,ϵ)-clustering $C_1,\dots C_k$. Let M be the random walk transition matrix of G. Let $1 > \xi > 1/n^5$, $t \geq \frac{20\log n}{\varphi^2}$. Let $c > 1$ be a large enough constant and let $s \geq c\cdot k^8\cdot n^{(400\epsilon/\varphi^2)}\cdot\log n/\xi^2$. Let $I_S = \{i_1,\dots,i_s\}$ be a multiset of s indices chosen independently and uniformly at random from $\{1,\dots,n\}$. Let S be the $n\times s$ matrix whose j-th column equals $\mathbb{1}_{i_j}$. Suppose that $M^t = U\Sigma^tU^T$ is the eigendecomposition of M^t and $\sqrt{\frac{n}{s}}\cdot M^tS = \widetilde{U}\widetilde{\Sigma}\widetilde{W}^T$ is the SVD of $\sqrt{\frac{n}{s}}\cdot M^tS$. If $\frac{\epsilon}{\varphi^2} \leq \frac{1}{10^5}$ then with probability at least $1-n^{-100\cdot k}$ we have

$$\left|\left|U_{[k]}\Sigma_{[k]}^{-2t}U_{[k]}^T - \widetilde{U}_{[k]}\widetilde{\Sigma}_{[k]}^{-2}\widetilde{U}_{[k]}^T\right|\right|_2 < \xi$$

The following lemma from [GKL[+]21a] is instrumental in analyzing collision probabilities of random walks from every vertex $x \in V$ in a (k,φ,ϵ)-clusterable graph.

Lemma 17 ([GKL+21a]). Let $k \geq 2$ be an integer, $\varphi \in (0,1)$ and $\epsilon \in (0,1)$. Let $G = (V, E)$ be a d-regular and that admits a (k, φ, ϵ)-clustering $C_1, \ldots, C_k$. Let M be the random walk transition matrix of G. For any $t \geq \frac{20 \log n}{\varphi^2}$ and any $x \in V$ we have

$$\|M^t \mathbb{1}_x\|_2 \leq O\left(k \cdot n^{-1/2 + 20\epsilon/\varphi^2}\right).$$

To prove the correctness of weighted dot product of spectral embedding of vertices, we use a similar proof strategy to [GKL+21a], which, albeit, develops an estimator for the *unweighted* dot product between spectral embeddings i.e., $\langle f_x, f_y \rangle$. In Lemma 18, we show that the weighted dot product of spectral embeddings i.e., $\left\langle f_x, \Sigma_{[k]} f_y \right\rangle$ can be estimated by the appropriate linear transformation of the random walk transition matrix. Since we seek weighted dot products unlike [GKL+21a], we run a t-step random walk from x, and a $t+1$-step walk from y. The one-step longer walk helps us to inject the matrix of eigenvalues in between the dot product of spectral embedding of vertex x and y.

Lemma 18. Let $k \geq 2$ be an integer, $\varphi \in (0,1)$ and $\epsilon \in (0,1)$. Let $G = (V, E)$ be a d-regular graph that admits a (k, φ, ϵ)-clustering $C_1, \ldots C_k$. Let M be the random walk transition matrix of G. Let $1/n^5 < \xi < 1$, $t \geq \frac{20 \log n}{\varphi^2}$. Let $c > 1$ be a large enough constant and let $s \geq c \cdot n^{480\epsilon/\varphi^2} \cdot \log n \cdot k^{13}/(\xi^2)$. Let $I_S = \{i_1, \ldots, i_s\}$ be a multiset of s indices chosen independently and uniformly at random from $\{1, \ldots, n\}$. Let S be the $n \times s$ matrix whose j-th column equals $\mathbb{1}_{i_j}$. Let $M^t = U \Sigma^t U^T$ be the eigendecomposition of M^t and $\sqrt{\frac{n}{s}} \cdot M^t S = \widetilde{U} \widetilde{\Sigma} \widetilde{W}^T$ be the SVD of $\sqrt{\frac{n}{s}} \cdot M^t S$. If $\frac{\epsilon}{\varphi^2} \leq \frac{1}{10^5}$ then with probability at least $1 - n^{-100 \cdot k}$ we have

$$\left| \mathbb{1}_x^T U_{[k]} \Sigma_{[k]} U_{[k]}^T \mathbb{1}_y - (M^{t+1} \mathbb{1}_x)^T (M^t S) \left(\frac{n}{s} \cdot \widetilde{W}_{[k]} \widetilde{\Sigma}_{[k]}^{-4} \widetilde{W}_{[k]}^T \right) (M^t S)^T (M^t \mathbb{1}_y) \right| \leq \frac{\xi}{nk^2}.$$

Proof. Let $m_x = M^{t+1} \mathbb{1}_x$ and $m_y = M^t \mathbb{1}_y$. Let $c' > 1$ be a large enough constant we will set later. Let $\xi' = \frac{\xi}{c' \cdot k^4 \cdot n^{40\epsilon/\varphi^2}}$. Let c_1 be the constant in front of s in Lemma 16. Thus for large enough c we have $s \geq c \cdot n^{480\epsilon/\varphi^2} \cdot \log n \cdot k^{13}/(\xi^2) \geq c_1 \cdot n^{400\epsilon/\varphi^2} \cdot \log n \cdot k^8/(\xi'^2)$, and therefore by Lemma 16 applied with ξ', with probability at least $1 - n^{-100 \cdot k}$ we have

$$\left|\left| U_{[k]} \Sigma_{[k]}^{-2t} U_{[k]}^T - \widetilde{U}_{[k]} \widetilde{\Sigma}_{[k]}^{-2} \widetilde{U}_{[k]}^T \right|\right|_2 \leq \xi'$$

By Cauchy-Schwarz and submultiplicativity of the spectral norm we have

$$\begin{aligned} \left| m_x^T U_{[k]} \Sigma_{[k]}^{-2t} U_{[k]}^T m_y - m_x^T \widetilde{U}_{[k]} \widetilde{\Sigma}_{[k]}^{-2} \widetilde{U}_{[k]}^T m_y \right| &\leq \left|\left| U_{[k]} \Sigma_{[k]}^{-2t} U_{[k]}^T - \widetilde{U}_{[k]} \widetilde{\Sigma}_{[k]}^{-2} \widetilde{U}_{[k]}^T \right|\right|_2 \|m_x\|_2 \|m_y\|_2 \\ &\leq \xi' \|m_x\|_2 \|m_y\|_2 \end{aligned} \tag{50}$$

In the rest of the proof we will show $m_x^T (U_{[k]} \Sigma_{[k]}^{-2t} U_{[k]}^T) m_y = \mathbb{1}_x^T U_{[k]} \Sigma_{[k]} U_{[k]}^T \mathbb{1}_y$ (**Step 1**) and $m_x^T \widetilde{U}_{[k]} \widetilde{\Sigma}_{[k]}^{-2} \widetilde{U}_{[k]}^T m_y = m_x^T (M^t S)(\widetilde{W}_{[k]} \widetilde{\Sigma}_{[k]}^{-4} \widetilde{W}_{[k]}^T)(M^t S)^T m_y$ (**Step 2**), and finally obtain the result by combining these facts with (50) and the upper bound on $\|m_x\|_2$ provided by Lemma 17.

Step 1: Note that $M^t = U \Sigma^t U^T$. Therefore we get $M^{t+1} \mathbb{1}_x = U \Sigma^{t+1} U^T \mathbb{1}_x$, and $M^t \mathbb{1}_y = U \Sigma^t U^T \mathbb{1}_y$. Thus we have

$$m_x^T U_{[k]} \Sigma_{[k]}^{-2t} U_{[k]}^T m_y = \mathbb{1}_x^T \left(\left(U \Sigma^{t+1} U^T \right) \left(U_{[k]} \Sigma_{[k]}^{-2t} U_{[k]}^T \right) \left(U \Sigma^t U^T \right) \right) \mathbb{1}_y \tag{51}$$

Note that $U^T U_{[k]}$ is a $n \times k$ matrix such that the top $k \times k$ matrix is $I_{k\times k}$ and the rest is zero. Also $U_{[k]}^T U$ is a $k \times n$ matrix such that the left $k \times k$ matrix is $I_{k\times k}$ and the rest is zero. Therefore we have

$$U\Sigma^{t+1}\left(U^T U_{[k]}\right)\Sigma_{[k]}^{-2t}\left(U_{[k]}^T U\right)\Sigma^t U^T = UHU^T,$$

where H is a $n \times n$ matrix such that the top left $k \times k$ matrix is $\Sigma_{k\times k}$ and the rest is zero. Hence, we have

$$UHU^T = U_{[k]}\Sigma_{[k]}U_{[k]}^T.$$

Thus we have

$$m_x^T(U_{[k]}\Sigma_{[k]}^{-2t}U_{[k]}^T)m_y = \mathbb{1}_x^T U_{[k]}\Sigma_{[k]}U_{[k]}^T\mathbb{1}_y \tag{52}$$

Step 2: We have $\sqrt{\frac{n}{s}}\cdot M^t S = \widetilde{U}\widetilde{\Sigma}\widetilde{W}^T$ where $\widetilde{U}\in\mathbb{R}^{n\times n}$, $\widetilde{\Sigma}\in\mathbb{R}^{n\times n}$ and $\widetilde{W}\in\mathbb{R}^{s\times n}$. Therefore,

$$\begin{aligned}
&(m_x)^T(M^t S)\left(\frac{n}{s}\cdot\widetilde{W}_{[k]}\widetilde{\Sigma}_{[k]}^{-4}\widetilde{W}_{[k]}^T\right)(M^t S)^T(m_y)\\
&= m_x^T\left(\sqrt{\frac{s}{n}}\cdot\widetilde{U}\widetilde{\Sigma}\widetilde{W}^T\right)\left(\frac{n}{s}\cdot\widetilde{W}_{[k]}\widetilde{\Sigma}_{[k]}^{-4}\widetilde{W}_{[k]}^T\right)\left(\sqrt{\frac{s}{n}}\cdot\widetilde{W}\widetilde{\Sigma}\widetilde{U}^T\right)m_y\\
&= m_x^T\left(\widetilde{U}\widetilde{\Sigma}\widetilde{W}^T\right)\left(\widetilde{W}_{[k]}\widetilde{\Sigma}_{[k]}^{-4}\widetilde{W}_{[k]}^T\right)\left(\widetilde{W}\widetilde{\Sigma}\widetilde{U}^T\right)m_y
\end{aligned} \tag{53}$$

Note that $\widetilde{W}^T\widetilde{W}_{[k]}$ is a $n \times k$ matrix such that the top $k \times k$ matrix is $I_{k\times k}$ and the rest is zero. Also $\widetilde{W}_{[k]}^T\widetilde{W}$ is a $k \times n$ matrix such that the left $k \times k$ matrix is $I_{k\times k}$ and the rest is zero. Therefore we have

$$\widetilde{\Sigma}\left(\widetilde{W}^T\widetilde{W}_{[k]}\right)\widetilde{\Sigma}_{[k]}^{-4}\left(\widetilde{W}_{[k]}^T\widetilde{W}\right)\widetilde{\Sigma} = \widetilde{H},$$

where $\widetilde{H}$ is a $n \times n$ matrix such that the top left $k \times k$ matrix is $\widetilde{\Sigma}_{[k]}^{-2}$ and the rest is zero. Hence, we have

$$(\widetilde{U}\widetilde{\Sigma}\widetilde{W}^T)\left(\frac{n}{s}\cdot\widetilde{W}_{[k]}\widetilde{\Sigma}_{[k]}^{-4}\widetilde{W}_{[k]}^T\right)(\widetilde{W}\widetilde{\Sigma}\widetilde{U}^T) = \widetilde{U}\widetilde{H}\widetilde{U}^T = \widetilde{U}_{[k]}\widetilde{\Sigma}_{[k]}^{-2}\widetilde{U}_{[k]}^T \tag{54}$$

Putting (54) and (53) together we get

$$m_x^T(M^t S)(\widetilde{W}_{[k]}\widetilde{\Sigma}_{[k]}^{-4}\widetilde{W}_{[k]}^T)(M^t S)^T m_y = m_x^T\widetilde{U}_{[k]}\widetilde{\Sigma}_{[k]}^{-2}\widetilde{U}_{[k]}^T m_y \tag{55}$$

Putting it together. By (50), (52) and (55) we have

$$\begin{aligned}
&\left|m_x^T(M^t S)\left(\frac{n}{s}\cdot\widetilde{W}_{[k]}\widetilde{\Sigma}_{[k]}^{-4}\widetilde{W}_{[k]}^T\right)(M^t S)^T m_y - \mathbb{1}_x^T U_{[k]}U_{[k]}^T\mathbb{1}_y\right|\\
&= \left|m_x^T\widetilde{U}_{[k]}\widetilde{\Sigma}_{[k]}^{-2}\widetilde{U}_{[k]}^T m_y - m_x^T U_{[k]}\Sigma_{[k]}^{-2t}U_{[k]}^T m_y\right| && \text{By (52) and (55)}\\
&\le \xi'\cdot\|m_x\|_2\|m_y\|_2 && \text{By (50)}
\end{aligned} \tag{56}$$

By Lemma 17 for any vertex $x \in V$ we have

$$\|m_x\|_2^2 = \|M^t\mathbb{1}_x\|_2^2 \le O\left(k^2\cdot n^{-1+40\epsilon/\varphi^2}\right). \tag{57}$$

Therefore by choice of c' as a large enough constant and choosing $\xi' = \frac{\xi}{c'\cdot k^4\cdot n^{40\epsilon/\varphi^2}}$ we have

$$\left|m_x^T(M^t S)\left(\frac{n}{s}\cdot\widetilde{W}_{[k]}\widetilde{\Sigma}_{[k]}^{-4}\widetilde{W}_{[k]}^T\right)(M^t S)^T m_y - \mathbb{1}_x^T U_{[k]}\Sigma_{[k]}U_{[k]}^T\mathbb{1}_y\right| \le O\left(\xi'\cdot k^2\cdot n^{-1+40\epsilon/\varphi^2}\right) \le \frac{\xi}{nk^2}. \tag{58}$$

□

Finally, Lemma 19 bounds the absolute deviation between $\left\langle f_x, \Sigma_{[k]} f_y \right\rangle_{apx}$ and our estimator. We put the two together using triangle inequality to prove Theorem A.3

Lemma 19 (A higher success probability version of Lemma 29 from [GKL$^+$21b] with improved estimation error.). Let $G = (V, E)$ be a d-regular that admits a (k, φ, ϵ)-clustering $C_1, \dots C_k$. Let $1/n^5 < \xi < 1$. Let $\mathcal{D}$ denote the data structure constructed by the procedure $\textsc{InitializeOracle}(G, \delta, \xi)$ (Algorithm 7). Let $x, y \in V$. Let $\left\langle f_x, \Sigma_{[k]} f_y \right\rangle_{apx} \in \mathbb{R}$ denote the value returned by the procedure $\textsc{WeightedDotProductOracle}(G, x, y, \xi, \mathcal{D})$ (Algorithm 8). Let $t \geq \frac{20 \log n}{\varphi^2}$. Let $c > 1$ be a large enough constant and let $s \geq c \cdot n^{240 \cdot \epsilon/\varphi^2} \cdot \log n \cdot k^4$. Let $I_S = \{i_1, \dots, i_s\}$ be a multiset of s indices chosen independently and uniformly at random from $\{1, \dots, n\}$. Let S be the $n \times s$ matrix whose j-th column equals $\mathbb{1}_{i_j}$. Let M be the random walk transition matrix of G. Let $\sqrt{\frac{n}{s}} \cdot M^t S = \widetilde{U} \widetilde{\Sigma} \widetilde{W}^T$ be the SVD of $\sqrt{\frac{n}{s}} \cdot M^t S$. If $\frac{\epsilon}{\varphi^2} \leq \frac{1}{10^5}$, and Algorithm 7 succeeds, then with probability at least $1 - n^{-100k}$ we have

$$\left| \left\langle f_x, \Sigma_{[k]} f_y \right\rangle_{apx} - (M^{t+1} \mathbb{1}_x)^T (M^t S) \left(\frac{n}{s} \cdot \widetilde{W}_{[k]} \widetilde{\Sigma}_{[k]}^{-4} \widetilde{W}_{[k]}^T \right) (M^t S)^T (M^t \mathbb{1}_y) \right| < \frac{\xi}{nk^2}.$$

Remark C.2. The result in Gluch et al. above, obtains a success probability of at least $1 - n^{-100}$. It can be improved to $1 - n^{-100k}$ with an overhead of *poly*(k) times as many samples.

We now prove Theorem A.3

Proof of Theorem A.3. **Correctness:** Let $s = \Theta(n^{480\epsilon/\varphi^2} \cdot \log n \cdot k^{13}/(\xi^2))$. Recall that $I_S = \{i_1, \dots, i_s\}$ is the multiset of s vertices each sampled uniformly at random (see line 3 of Algorithm 7). Let S be the $n \times s$ matrix whose j-th column equals $\mathbb{1}_{i_j}$. Recall that M is the random walk transition matrix of G. Let $\sqrt{\frac{n}{s}} \cdot M^t S = \widetilde{U} \widetilde{\Sigma} \widetilde{W}^T$ be the eigendecomposition of $\sqrt{\frac{n}{s}} \cdot M^t S$. We define

$$e_1 = \left| (M^{t+1} \mathbb{1}_x)^T (M^t S) \left(\frac{n}{s} \cdot \widetilde{W}_{[k]} \widetilde{\Sigma}_{[k]}^{-4} \widetilde{W}_{[k]}^T \right) (M^t S)^T (M^t \mathbb{1}_y) - \mathbb{1}_x^T U_{[k]} \Sigma_{[k]} U_{[k]}^T \mathbb{1}_y \right|$$

and

$$e_2 = \left| \left\langle f_x, \Sigma_{[k]} f_y \right\rangle_{apx} - (M^{t+1} \mathbb{1}_x)^T (M^t S) \left(\frac{n}{s} \cdot \widetilde{W}_{[k]} \widetilde{\Sigma}_{[k]}^{-4} \widetilde{W}_{[k]}^T \right) (M^t S)^T (M^t \mathbb{1}_y) \right|$$

By triangle inequality we have

$$\left| \left\langle f_x, \Sigma_{[k]} f_y \right\rangle_{apx} - \left\langle f_x, \Sigma_{[k]} f_y \right\rangle \right| = \left| \left\langle f_x, \Sigma_{[k]} f_y \right\rangle_{apx} - \mathbb{1}_x^T U_{[k]} \Sigma_{[k]} U_{[k]}^T \mathbb{1}_y \right| \leq e_1 + e_2.$$

Let $\xi' = \xi/2$. Let c be a constant in front of s in Lemma 18 and c' be a constant in front of s in Lemma 19. Recall, line 3 of Algorithm 7 sets $s = \Theta(n^{480\epsilon/\varphi^2} \cdot \log n \cdot k^{13}/(\xi^2))$.

Since $\frac{\epsilon}{\varphi^2} \leq \frac{1}{10^5}$ and $s \geq c \cdot n^{480\epsilon/\varphi^2} \cdot \log n \cdot k^{13}/(\xi'^2)$, by Lemma 18 with probability at least $1 - n^{-100 \cdot k}$ we have $e_1 \leq \frac{\xi'}{nk^2} = \frac{\xi}{2 \cdot nk^2}$. Since $s \geq c' \cdot n^{240\epsilon/\varphi^2} \cdot \log n \cdot k^4$, by Lemma 19, with probability at least $1 - 2 \cdot n^{-100 \cdot k}$ we have $e_2 \leq \frac{\xi}{2 \cdot nk^2}$. Thus with probability at least $1 - 3 \cdot n^{-100 \cdot k}$ we have

$$\left| \left\langle f_x, \Sigma_{[k]} f_y \right\rangle_{apx} - \left\langle f_x, \Sigma_{[k]} f_y \right\rangle \right| \leq e_1 + e_2 \leq \frac{\xi}{2 \cdot nk^2} + \frac{\xi}{2 \cdot nk^2} \leq \frac{\xi}{nk^2}.$$

Running time of InitializeOracle: The algorithm first samples a set I_S. Then, as per line 7 of Algorithm 7, it estimates the empirical probability distribution of t-step random walks starting from any vertex $x \in I_S$. The $\textsc{EstimateTransitionMatrix}$ procedure runs R_{init} random walks of length t from each vertex $x \in I_S$. So it takes $O(\log n \cdot s \cdot R_{\text{init}} \cdot t)$ time and requires $O(\log n \cdot s \cdot R_{\text{init}})$ space to store endpoints of random walks. Then as per line 6 of Algorithm

7 it estimates matrix $\mathcal{G}$ such that the entry corresponding to the x^{th} row and y^{th} column of $\mathcal{G}$ is an estimation of pairwise collision probability of random walks starting from $x, y \in I_S$. To compute $\mathcal{G}$ we call Algorithm EstimateCollisionProbabilities$(G, I_S, R_{\text{init}}, t)$ (from [GKL+21a]) for $O(\log n)$ times. This procedure takes $O(s \cdot R_{\text{init}} \cdot t \cdot \log n)$ time and it requires $O(s^2 \cdot \log n)$ space to store matrix $\mathcal{G}$. Computing the SVD of $\mathcal{G}$ (done in line 7 of Algorithm 7) takes time $O(s^3)$. Thus overall Algorithm 7 runs in time $O\left(\log n \cdot s \cdot R_{\text{init}} \cdot t + s^3\right)$. Thus, by choice of $t = \Theta\left(\frac{\log n}{\varphi^2}\right)$, $R_{\text{init}} = \Theta(n^{1/2+O(\epsilon/\varphi^2)} \cdot \log^{O(1)} n \cdot k^{O(1)}/(\xi)^{O(1)})$ and $s = \Theta(n^{O(\epsilon/\varphi^2)} \cdot (\log n)^{O(1)} \cdot k^{O(1)}/\xi^{O(1)})$ as in Algorithm 7 we get that Algorithm 7 runs in time $\log n \cdot s \cdot R_{\text{init}} \cdot t + s^3 = (\frac{k \cdot \log n}{\xi \cdot \varphi})^{O(1)} \cdot n^{1/2+O(\epsilon/\varphi^2)}$ and returns a data structure of size $O\left(s^2 + \log n \cdot s \cdot R_{\text{init}}\right) = (\frac{k \cdot \log n}{\xi})^{O(1)} \cdot n^{1/2+O(\epsilon/\varphi^2)}$.

Running time of WeightedDotProductOracle: Algorithm WeightedDotProductOracle runs R_{query} random walks of length $t, t+1$ from vertex x and vertex y, then it computes $(\widehat{m}_x^T \widehat{Q})$ and $(\widehat{Q}^T \widehat{m}_y)$. Since $\widehat{Q} \in \mathbb{R}^{n \times s}$ has s columns and since $\widehat{m}_x$ has at most R_{query} non-zero entries, thus one can compute $\widehat{m}_x^T \cdot \widehat{Q}$ in time $R_{\text{query}} \cdot s$. Finally Algorithm 8 returns value $(\widehat{m}_x^T \widehat{Q})\Psi(\widehat{Q}^T \widehat{m}_y)$. Since $(\widehat{m}_x^T \widehat{Q}), (\widehat{Q}^T \widehat{m}_y) \in \mathbb{R}^s$ and $\Psi \in \mathbb{R}^{s \times s}$ one can compute $(\widehat{m}_x^T \widehat{Q})\Psi(\widehat{Q}^T \widehat{m}_y)$ in time $O(s^2)$. Thus overall Algorithm 8 takes $O\left(t \cdot R_{\text{query}} + s \cdot R_{\text{query}} + s^2\right)$ time. Thus, by choice of $t = O\left(\frac{\log n}{\varphi^2}\right)$, $R_{\text{query}} = n^{1/2+O(\epsilon/\varphi^2)} \cdot \left(\frac{k}{\xi}\right)^{O(1)}$ and $s = n^{O(\epsilon/\varphi^2)} \cdot (\frac{k \cdot \log n}{\xi})^{O(1)}$ we get that the Algorithm 8 runs in time $(\frac{k \cdot \log n}{\xi \cdot \varphi})^{O(1)} \cdot n^{1/2+O(\epsilon/\varphi^2)}$. □