Sublinear Algorithms for Estimating Single-Linkage Clustering Costs

Pan Peng* Christian Sohler† Yi Xu‡

Abstract

Single-linkage clustering is a fundamental method for data analysis. It proceeds iteratively by merging the two clusters with the smallest inter-cluster distance, starting from singleton clusters, until all points are combined into a single cluster. The distance between two clusters is defined as the minimum distance between any pair of points across the clusters. Algorithmically, one can compute a single-linkage k-clustering (a partition into k clusters) by computing a minimum spanning tree and dropping the $k-1$ most costly edges. This clustering minimizes the sum of spanning tree weights of the clusters. This motivates us to define the cost of a single-linkage k-clustering as the weight of the corresponding spanning forest, denoted by cost_k. Besides, if we consider single-linkage clustering as computing a hierarchy of clusterings, the total cost of the hierarchy is defined as the sum of the individual clusterings, denoted by $\text{cost}(G) = \sum_{k=1}^{n} \text{cost}_k$.

In this paper, we assume that the distances between data points are given as a graph G with average degree d and edge weights from $\{1,\dots,W\}$. If there is no edge, we assume the distance to be infinite. Given query access to the adjacency list of G, we present a sampling-based algorithm that computes a succinct representation of estimates $\widehat{\text{cost}}_k$ for all k. The running time is $\tilde{O}(d\sqrt{W}/\varepsilon^3)$, and the estimates satisfy $\sum_{k=1}^{n} |\widehat{\text{cost}}_k - \text{cost}_k| \le \varepsilon \cdot \text{cost}(G)$, for any $0 < \varepsilon < 1$. Thus we can approximate the cost of every k-clustering upto $(1+\varepsilon)$ factor *on average*. In particular, our result ensures that we can estimate $\text{cost}(G)$ upto a factor of $1 \pm \varepsilon$ in the same running time. We further establish a lower bound showing that our algorithm is nearly optimal, by proving that any algorithm achieving such accuracy must take $\Omega(d\sqrt{W}/\varepsilon^2)$ queries.

We also extend our results to the similarity setting, where edges represent similarities rather than distances. In this case, the clusterings are defined by a maximum spanning tree, and our algorithms run in $\tilde{O}(dW/\varepsilon^3)$ time. We also prove a nearly matching lower bound of $\Omega(Wd/\varepsilon^2)$ queries for estimating the total similarity-based cost. These bounds reveal an interesting – and perhaps surprising – separation between the distance and similarity settings. Finally, we extend our algorithms to metric space settings and validate our theoretical findings through extensive experiments.

*School of Computer Science and Technology, University of Science and Technology of China. Email: ppeng@ustc.edu.cn. Supported in part by NSFC grant 62272431.

†Department of Mathematics and Computer Science, University of Cologne. Email: csohler@uni-koeln.de. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 459420781.

‡School of Computer Science and Technology, University of Science and Technology of China. Email: yi_xu@mail.ustc.edu.cn. Supported in part by NSFC grant 62272431.

Contents

1 Introduction

Hierarchical clustering is a fundamental and widely used technique in data analysis and machine learning. In agglomerative (bottom-up) clustering, we start with a set of n objects and a defined distance or similarity relationship between these objects, as well as a method to evaluate the distance or similarity between clusters, i.e. groups of objects. The clustering process begins by treating each object as its own individual cluster. It then iteratively merges pairs of clusters based on a specified criterion related to their distance or similarity, continuing this process until only a single cluster remains. This merging process implicitly defines a hierarchy of clusters, and for every choice of the number of clusters k, the corresponding clustering is the one obtained after the first $n-k$ merges. Agglomerative clustering—typically implemented through methods such as single-linkage, complete-linkage, and average linkage—has been extensively studied and widely used in various fields, including data analysis and machine learning (see e.g. [HTF09, For10]). These methods are well-established in standard data analysis toolkits.

In this work, we focus on one of the most widely used agglomerative clustering methods: single-linkage clustering (SLC). When the underlying relationship between objects is defined in terms of *distance*, SLC operates by iteratively identifying the *closest* pair of objects that belong to different clusters and connecting them with an edge. This approach emphasizes the merging of clusters based on proximity. To mathematically represent this process, we utilize a graph G where each vertex corresponds to an object, and the weight of an edge indicates the distance between these objects. In this framework (see e.g. [GR69]), SLC can be formalized by first constructing a minimum spanning tree (MST) T of G. The edges of T are then considered in *non-decreasing* order of their weights to facilitate cluster merging. This iterative process continues until the desired number of clusters, k, is reached, resulting in a k-clustering. In particular, when $k = 1$, it reveals the complete hierarchical structure of the graph. In cases where the relationship between objects is based on *similarity* rather than distance, SLC operates by iteratively identifying the *most similar* pair of objects that belong to different clusters and can be defined using a *maximum* spanning tree. Specifically, we first construct a maximum spanning tree (MaxST) T of graph G and then order the edges in T in *non-increasing* order of their weights, subsequently merging clusters iteratively.

SLC typically requires at least linear time for implementation. In many applications, the underlying graph can be massive, making computational analysis challenging – simply reading the input may become impractical or even impossible. This motivates to design *sublinear-time algorithms* that only read a small portion of the input to extract the SLC information. Since producing the full clustering hierarchy necessarily takes linear time, we instead ask:

> *Can we define **cost measures** that effectively capture the behavior and structure of SLC, and can we approximate these measures – as well as key summary statistics, such as a **succinct representation** of the hierarchy – in sublinear time?*

1.1 Our Contributions

We answer the above question in the affirmative by introducing several cost measures and presenting nearly optimal sublinear-time algorithms that approximate these costs and the corresponding key summary statistics. We begin by introducing the cost measures.

Depending on the relationship between objects is distance or similarity, we consider two settings.

1.1.1 Cost Functions from SLC

SLC in the Distance Graph We are given a connected weighted graph $G = (V, E)$ with edge weights $w(e) \in [1, W]$, for some parameter $W \geq 1$. The edge weights are the distances between the objects; for any pair i, j that does not form an edge, i.e., $(i, j) \notin E$, we define $w((i, j)) = \infty$.

Recall that in each step of single-linkage clustering (SLC), we merge the two clusters that are connected by the closest pair of objects – that is, by the shortest edge between them. Since this decision is made by evaluating pairs of clusters and the cost of merging them, it is natural to define a cost function at the cluster level. In particular, the cost of each cluster should be independent of the structure of the other clusters. This leads us to define the overall clustering cost as the sum of individual cluster costs, analogous to classical objectives in k-median and k-means clustering.

A key question, then, is: what cost function is implicitly being optimized by this greedy merging process? A natural candidate is the cost of the minimum spanning tree, which represents the minimum total weight needed to connect all elements of a cluster. Indeed, under this cost function, the optimal k-clustering is exactly the one produced by SLC when the algorithm is halted with k clusters remaining. Now we formally define the k-clustering cost for SLC.

***k*-clustering Cost:** In SLC, the k-clustering corresponds to a partition into k connected components obtained by removing the $k-1$ most expensive edges from the MST. This observation motivates the following definition: the cost of a k-clustering is defined as the sum of the weights of the minimum spanning trees of each of the k clusters. We denote this cost as cost_k. Formally, let $w_1, \ldots, w_{n-1}$ be the edge weights of the MST sorted in non-decreasing order. Then, the *cost of the k-single-linkage clustering (k-SLC)* is given by

$$\text{cost}_k = \sum_{i=1}^{n-k} w_i.$$

This corresponds to the total weight of the $n-k$ smallest edges in the MST, that is, the edges used to form the k clusters in SLC. Equivalently, it is the total weight of the resulting spanning forest with k connected components.

Total Cost of the Hierarchy: In a graph with n vertices, the cost of the n-clustering is 0, since every vertex forms its own cluster. At the other extreme, the 1-clustering cost equals the total weight of the MST, marking the end of the hierarchy. We define the *total cost of the clustering hierarchy* computed by the SLC algorithm as

$$\text{cost}(G) = \sum_{k=1}^{n} \text{cost}_k = \sum_{k=1}^{n-1} \text{cost}_k = \sum_{i=1}^{n-1} w_i + \sum_{i=1}^{n-2} w_i + \cdots + \sum_{i=1}^{2} w_i + w_1 = \sum_{i=1}^{n-1} (n-i) w_i \tag{1}$$

This can be interpreted as the sum over all edges in the MST with each edge weighted according to the point of time when they are used to merge two clusters by the single-linkage algorithm—earlier edges contribute more to the total cost.

Note that each merge step in the SLC algorithm minimizes the corresponding flat clustering cost under our definition. This follows by induction: For the base case $k = n-1$, SLC selects the minimum-weight edge in the graph, minimizing cost_{n-1}. Assuming cost_k is minimized, SLC merges the closest pair of clusters to form $k-1$ clusters, adding the next lightest MST edge and thus minimizing cost_{k-1}. Therefore, SLC minimizes each cost_k, and the entire hierarchy it constructs is optimal under this cost function.

Profile Vector: Note that cost_k provides finer-grained information about the state of the hierarchy at each level. Specifically, we define the length-n vector $(\text{cost}_1, \text{cost}_2, \ldots, \text{cost}_n)$ as the *SLC profile vector of*

the distance graph G. This vector captures nuanced information about the cluster structure in the graph and offers insight into the hierarchical relationships among clusters.

For example, if the curve of cost_k versus k is relatively smooth, this suggests that as k decreases, cost_k increases more gradually, which may indicate a denser underlying graph with generally shorter pairwise distances. Furthermore, if the total clustering cost, $\text{cost}(G)$, is relatively small, this can reflect short distances between vertices in G. These characteristics are further evidenced in our experiments (see Fig. 1b and Fig. 1c).

SLC in the Similarity Graph Similarly to the distance based formulation we are given a connected weighted graph $G = (V,E)$ with edge weights $w(e) \in [1,W]$ for some parameter $W \geq 1$. Here the edge weights represent similarities and for any pair $(i,j) \notin E$ we define $w((i,j)) = 0$.

If the relationship between objects is based on *similarity* rather than distance, SLC iteratively finds the *most similar* pair of objects that belong to different clusters to merge, and can be defined using a *maximum* spanning tree instead. In this case, edges are processed in *non-increasing* order of their weights.

***k*-clustering Cost:** We extend our formulation in the distance-based clustering to the case of similarity-based clustering, and define the cost of a k-clustering as the sum of the costs of the maximum spanning trees of the clusters, denoted as $\text{cost}_k^{(s)}$. Formally, let $w_1^{(s)}, w_2^{(s)}, ..., w_{n-1}^{(s)}$ be the weights of the maximum spanning tree (MaxST) in non-increasing order (we use the superscript $^{(s)}$ to indicate the case of a *similarity relationship*). The cost of a k-clustering in the similarity graph can then be defined as

$$\text{cost}_k^{(s)} = \sum_{i=1}^{n-k} w_i^{(s)}.$$

This corresponds to the total weight of the $n-k$ largest edges in the MaxST, that is, the edges used to form the k clusters in SLC.

Total Cost of the Hierarchy: The total cost of the single-linkage clustering in the similarity graph is then defined as

$$\text{cost}^{(s)}(G) = \sum_{k=1}^{n} \text{cost}_k^{(s)} = \sum_{k=1}^{n-1} \text{cost}_k^{(s)} = \sum_{i=1}^{n-1} (n-i) \cdot w_i^{(s)} \tag{2}$$

Analogous to the induction analysis in the distance setting, we observe that each merge step in the SLC algorithm precisely maximizes the similarity of the corresponding flat clustering under our definition. Consequently, the entire SLC algorithm maximizes the total similarity of the resulting hierarchy according to this new cost function.

Profile Vector: Analogously, we define the length-n vector $(\text{cost}_1^{(s)}, \text{cost}_2^{(s)}, \ldots, \text{cost}_n^{(s)})$ as the *SLC profile vector of the similarity graph G*.

1.1.2 Sublinear Time Algorithms

We give sublinear algorithms for estimating the costs of the hierarchy of single-linkage clustering for both distance-based and similarity-based clustering. Our algorithms assume query access to the graph in the *adjacency list model* (except in the metric space setting discussed below), where one can access the weight of the i-th neighbor of a specified vertex v in $\Theta(i)$ time[1] (see Section 2). The query complexity of an algorithm refers to the maximum number of such queries made on any input.

[1] Our hardness result also holds for the case of $O(1)$ time access.

Distance-based Clustering We start with the problem of estimating $\mathrm{cost}(G)$ given in Equation (1) in sublinear time. We have the following result. (Throughout the paper, $\tilde{O}(f)$ refers to $O(f \cdot \mathrm{poly}\log f)$.)

Theorem 1.1. *Let G be a weighted graph with edge weights in $\{1,\ldots,W\}$ with average (unweighted) degree d. Assume that $\sqrt{W} \le n$ and let $0 < \varepsilon < 1$ be a parameter. Algorithm 4 outputs an estimate $\widehat{\mathrm{cost}}(G)$ of the single-linkage clustering cost $\mathrm{cost}(G)$ in the distance graph such that with probability at least $3/4$,*

$$(1-\varepsilon)\mathrm{cost}(G) \le \widehat{\mathrm{cost}}(G) \le (1+\varepsilon)\mathrm{cost}(G).$$

The query complexity and running time of the algorithm are $O(\frac{\sqrt{W}d}{\varepsilon^3}\log^4(\frac{Wd}{\varepsilon}))$ in expectation.

Note that the running time of our algorithm only depends on W, d, ε and is independent of the size n of the graph. Furthermore, in Theorem 1.1, we assumed that $\sqrt{W} \le n$, as otherwise (i.e., $n < \sqrt{W}$), we can directly find the minimum spanning tree algorithm in $\tilde{O}(n \cdot d) = \tilde{O}(\sqrt{W}d)$ time and obtain the clustering cost exactly from it.

We then strengthen our result and give an algorithm that efficiently derives a succinct representation of the SLC profile vector $(\mathrm{cost}_1, \mathrm{cost}_2, \ldots, \mathrm{cost}_n)$ in sublinear time, which allows us to approximate the total cost of clustering. We obtain the following result.

Theorem 1.2. *Assume that $\sqrt{W} \le n$ and let $0 < \varepsilon < 1$ be a parameter. Algorithm 5 generates a succinct representation of an approximation $(\widehat{\mathrm{cost}}_1, \ldots, \widehat{\mathrm{cost}}_n)$ of the SLC profile vector $(\mathrm{cost}_1, \ldots, \mathrm{cost}_n)$ in the distance graph such that with probability at least $3/4$, it holds that*

$$\sum_{k=1}^{n} \left|\widehat{\mathrm{cost}}_k - \mathrm{cost}_k\right| \le \varepsilon \cdot \mathrm{cost}(G).$$

The query complexity and running time of the algorithm are $O(\frac{\sqrt{W}d}{\varepsilon^3}\log^4(\frac{Wd}{\varepsilon}))$ in expectation.

Note that the above theorem implies that we can approximate the cost cost_k of every k-clustering upto an absolute error that *on average* is a $(1+\varepsilon)$-approximation of the true cost. Given the succinct representation and any specified integer $k \in \{1,\ldots,n\}$, we can recover an estimate $\widehat{\mathrm{cost}}_k$ for cost_k in $O(\log(\frac{\log W}{\varepsilon}))$ time. The estimate satisfies that $|\widehat{\mathrm{cost}}_k - \mathrm{cost}_k| = O(\varepsilon \cdot (\mathrm{cost}_k + \max\{k, \frac{n}{\sqrt{W}}\}W))$ (see Lemma 6.8).

We remark that by applying the median trick, the success probability of the two aforementioned algorithms can be enhanced to $1-\delta$ for any δ, while incurring an $O(\log(1/\delta))$ factor in the running time (see Section 2).

To complement our algorithmic result, we show that the query complexity of the algorithm from Theorem 1.1 for estimating the SLC cost, $\mathrm{cost}(G)$, is nearly optimal by giving the following lower bound.

Theorem 1.3. *Let $\frac{W^{1/4}}{\sqrt{40n}} < \varepsilon < \frac{1}{2}$ and $W > 1$. Any algorithm that $(1+\varepsilon)$-approximates the cost of SLC cost $\mathrm{cost}(G)$ in the distance graph with success probability at least $3/4$ needs to make $\Omega(d\sqrt{W}/\varepsilon^2)$ queries.*

Similarity-based Clustering We extend our results to the case of similarity-based clustering. Our algorithmic result for estimating the cost defined in Equation (2) is given in the following theorem.

Theorem 1.4. *Let G be a weighted graph with edge weights in $\{1,\ldots,W\}$ with average (unweighted) degree d. Assume that $W \le n$ and let $0 < \varepsilon < 1$ be a parameter. Algorithm 8 outputs an estimate $\widehat{\mathrm{cost}^{(s)}}(G)$ of the single-linkage clustering cost $\mathrm{cost}^{(s)}(G)$ in the similarity graph such that with probability at least $3/4$,*

$$(1-\varepsilon)\mathrm{cost}^{(s)}(G) \le \widehat{\mathrm{cost}^{(s)}}(G) \le (1+\varepsilon)\mathrm{cost}^{(s)}(G).$$

The query complexity and running time of the algorithm are $O(\frac{Wd}{\varepsilon^3}\log^4(\frac{Wd}{\varepsilon}))$ in expectation.

We also give a sublinear time algorithm for approximating the SLC profile vector in the similarity graph G and establish the following theorem.

Theorem 1.5. *Assume that $W \leq n$ and let $\varepsilon < 1$. Algorithm 9 generates a succinct representation of an approximation $(\widehat{\text{cost}^{(s)}}_1, \ldots, \widehat{\text{cost}^{(s)}}_n)$ of the SLC profile vector $(\text{cost}_1^{(s)}, \ldots, \text{cost}_n^{(s)})$ in the similarity graph such that with probability at least $3/4$, it holds that*

$$\sum_{k=1}^{n} |\widehat{\text{cost}^{(s)}}_k - \text{cost}_k^{(s)}| \leq \varepsilon \cdot \text{cost}^{(s)}(G).$$

The query complexity and running time of the algorithm are $O(\frac{Wd}{\varepsilon^3}\log^4(\frac{Wd}{\varepsilon}))$ in expectation.

Similarly, we note that the above theorem implies that we can approximate the cost $\text{cost}_k^{(s)}$ of every k-clustering upto an absolute error that *on average* is a $(1+\varepsilon)$-approximation of the true cost. We further remark that given the succinct representation and any specified integer $k \in \{1, \ldots, n\}$, we can recover an estimate $\widehat{\text{cost}^{(s)}}_k$ for $\text{cost}_k^{(s)}$ in $O(\log(\frac{\log W}{\varepsilon}))$ time. The estimate satisfies that $|\widehat{\text{cost}^{(s)}}_k - \text{cost}_k^{(s)}| = O(\varepsilon \cdot \max\{\text{cost}_k^{(s)}, n\})$.

We can also use the median trick to boost the success probability of the above two algorithms to $1-\delta$ while incurring a $O(\log(1/\delta))$ factor in the running time.

We also show that our algorithm for estimating $\text{cost}^{(s)}(G)$ achieves nearly optimal query complexity by providing the following lower bound.

Theorem 1.6. *Let $\sqrt{\frac{W}{40n}} < \varepsilon < \frac{1}{2}$ and $W > 10$. Any algorithm that $(1+\varepsilon)$-approximates the SLC cost $\text{cost}^{(s)}(G)$ in the similarity graph with success probability at least $3/4$ needs to make $\Omega(dW/\varepsilon^2)$ queries.*

Comparison between the Distance and Similarity Settings Recall that for distance-based clustering, we achieved a running time of $\tilde{O}(\sqrt{W}d/\varepsilon^3)$ with a lower bound of $\Omega(\sqrt{W}d/\varepsilon^2)$, whereas for similarity-based clustering, our algorithm runs in $\tilde{O}(Wd/\varepsilon^3)$ with a matching lower bound of $\Omega(Wd/\varepsilon^2)$. This reveals an interesting – and perhaps surprising – separation in the complexities between the two settings.

Metric Space Clustering We further extend our results to the metric space, where the metric can either represent distance or similarity between vertices, and the metric must satisfy the triangle inequality. In this case, we assume that the algorithm can query the weight of any specified vertex pair in constant time.

When the metric represents distance, we have the following result.

Theorem 1.7. *Let G be an n-point graph in metric space, where each edge weight represents* distance *between two vertices, and $0 < \varepsilon < 1$ be a parameter. Algorithm 11 outputs an estimate $\widehat{\text{cost}}(G)$ of single-linkage clustering cost $\text{cost}(G)$ in metric space, such that with probability at least $3/4$,*

$$(1-\varepsilon)\text{cost}(G) \leq \widehat{\text{cost}}(G) \leq (1+\varepsilon)\text{cost}(G).$$

The query complexity and running time of the algorithm are $\tilde{O}(n/\varepsilon^7)$ in expectation.

When the metric represents similarity, we design an algorithm with the same performance guarantees as above, though the algorithm and analysis differ slightly. See Theorem C.1 for the formal statement.

To complement our theoretical findings, we conduct experiments on various real networks, including those where edges represent distance relationships and those where edges represent similarity relationships. Our experiments show that our algorithms achieve both good accuracy and better running time.

1.2 Technical Overview

The CRT Approach for Estimating the Weight of MST Our algorithms and lower bounds are inspired by the work of Chazelle, Rubinfeld, and Trevisan [CRT05] on estimating MST weight, which we refer to as the CRT approach. Their key insight is that the MST cost can be reduced to estimating the number of connected components (#CCs) in a sequence of thresholded subgraphs. Let G_j be the subgraph containing edges with weight at most j, and let c_j be its number of connected components. Then they show that the MST cost is $\text{cost}(\text{MST}) = n - W + \sum_{j=1}^{W-1} c_j$.

To estimate each c_j, their algorithm samples vertices and performs BFS-based local exploration, with the exploration size determined by a stochastic process. This yields an estimate of c_j within additive error εn in expected time $\tilde{O}(d/\varepsilon^2)$, where d is the average degree. Combining the estimates across all j gives a $(1+\varepsilon)$-approximation to cost(MST) in total time $\tilde{O}(dW/\varepsilon^2)$.

For the lower bound, they reduce from a distributional problem of distinguishing between two biased coin distributions, parameterized by $q \in (0, 1/2]$ and $\varepsilon \in (0,1)$. Any algorithm with success probability at least $3/4$ must make $\Omega(1/(q\varepsilon^2))$ queries. This is then used to construct two distributions over weighted graphs whose MST costs differ by at least a factor of $(1+\varepsilon)$, yet cannot be distinguished using $o(Wd/\varepsilon^2)$ queries.

We now present a high-level overview of the main techniques underlying our algorithms for estimating SLC costs.

1.2.1 On Estimating cost(G)

Our approach to estimating cost(G) proceeds in three main steps. (1) We begin by reducing the problem to estimating the number of connected components in a sequence of subgraphs, following the framework of [CRT05]. (2) We then adapt and refine their component-counting technique to suit our setting, providing a more tailored analysis. (3) Finally, to improve efficiency, we exploit the monotonicity of connected component counts across the subgraph sequence and apply a binary search strategy to accelerate computation. This final step constitutes the main technical contribution of our paper.

Reduction to Estimating #CCs By exploiting the relation between the number of edges on the MST with weight j and the number of connected components in certain subgraphs of G ([CRT05]), we derive an equivalent formula of cost(G) in Theorem 4.1. We show that

$$\text{cost}(G) = \frac{n(n-1)}{2} + \frac{1}{2} \cdot \sum_{j=1}^{W-1} (c_j^2 - c_j),$$

where c_j is the number of connected components in the subgraph induced by the edges of weight at most j. Note that our formula for cost(G) is a quadratic function of the number of the c_j. This stands in contrast to [CRT05] where the MST weight is a linear function of the c_j.

Invoking Sublinear Time Algorithm for #CCs To approximate cost(G) within a factor of $(1+\varepsilon)$, it suffices to estimate each c_j up to additive error $\varepsilon n/\sqrt{W}$. Recall that the CRT approach approximates the number of connected components in a subgraph within additive error εn in expected $\tilde{O}(d/\varepsilon^2)$ time, and the success probability can be boosted to $1-\delta$ with a multiplicative $\log(1/\delta)$ overhead. By setting the error tolerance to $\varepsilon n/\sqrt{W}$ and $\delta = O(1/W)$ for each c_j, we obtain a total running time of $\tilde{O}(d/(\varepsilon/\sqrt{W})^2 \cdot W) = \tilde{O}(W^2 d/\varepsilon^2)$ to estimate cost(G).

To improve this, we show in Lemma 3.1 that each c_j can instead be estimated within additive error $\max\{\varepsilon n/\sqrt{W}, \varepsilon c_j\}$ in time $\tilde{O}(\sqrt{W}d/\varepsilon^2)$ by adapting a variant of CRT algorithm with a refined analysis. Estimating all $c_1, \ldots, c_{W-1}$ to this precision yields a $(1+\varepsilon)$-approximation to $\text{cost}(G)$ in total time $\tilde{O}(\sqrt{W}d/\varepsilon^2 \cdot W) = \tilde{O}(W^{3/2}d/\varepsilon^2)$.

Applying a Binary Search Strategy We now ask whether the running time for estimating $\text{cost}(G)$ can be further improved, possibly to sublinear in W. Recall that $\text{cost}(G) = \frac{n(n-1)}{2} + \frac{1}{2}\sum_{j=1}^{W-1}(c_j^2 - c_j)$. Therefore, it suffices to approximate the sum $\sum_{j=1}^{W-1}(c_j^2 - c_j)$. Computing this exactly would require evaluating all $W-1$ terms, which is too costly. To design an $o(W)$-time algorithm, we must avoid estimating every c_j individually.

A crucial observation is that the sequence $(c_1, \ldots, c_W)$ is non-increasing, since G_j includes more edges as j increases, and adding edges cannot increase the number of connected components. This implies that the sequence $(c_j^2 - c_j)$ is also non-increasing. We exploit this monotonicity to design a faster algorithm.

A High Level Idea We partition the range $[1, n]$ into buckets $(B_{i+1}, B_i]$ using a geometric sequence of breakpoints: $B_1 \geq B_2 \geq \cdots \geq B_t$, with $B_{i+1} = B_i/(1+\varepsilon)$. Within each bucket, the c_j values are close, so we approximate all c_j in the bucket by a representative value, e.g., B_i. Then we estimate the sum $\sum_{j=1}^{W-1}(c_j^2 - c_j)$ as:

$$\sum_{i=1}^{t} [\#\text{ of } c_j \text{ in } (B_{i+1}, B_i]] \cdot (B_i^2 - B_i).$$

To estimate the number of c_j values in a bucket, we perform binary search over the $\{c_1, \ldots, c_W\}$ sequence to locate the indices where the values cross B_i and B_{i+1}. Since each such search uses $O(\log W)$ steps and the number of buckets is $O(\log_{1+\varepsilon} W) = O((\log W)/\varepsilon)$, we access at most $O((\log^2 W)/\varepsilon)$ values of c_j's.

Each c_j can be estimated within additive error $\max \varepsilon n/\sqrt{W}, \varepsilon c_j$ in time $\tilde{O}(\sqrt{W}d/\varepsilon^2)$, so the total running time becomes: $\tilde{O}\left(\frac{\sqrt{W}d}{\varepsilon^2} \cdot \frac{\log^2 W}{\varepsilon}\right) = \tilde{O}\left(\frac{\sqrt{W}d}{\varepsilon^3}\right)$. This yields the desired sublinear-in-W algorithm for estimating $\text{cost}(G)$.

The Challenge However, the main challenge is that we do not have direct access to the true sequence $(c_1, \ldots, c_W)$ – only approximate estimates $\hat{c}_j$. Due to estimation errors, the estimated sequence $(\hat{c}_1, \ldots, \hat{c}_W)$ may no longer be monotonic, making the direct application of the above algorithm infeasible.

Handling the Challenge In Section 5, we abstract a core technical problem: given a noisy version of the *non-increasing* sequence $X = (x_1, x_2, \ldots, x_W)$ with each $x_j \in [L, R]$, the goal is to obtain a succinct approximation of X without estimating every entry individually.

To build intuition, consider the simpler setting where all x_j are known. We divide the range $[L, R]$ into t intervals:

$$[B_t, B_{t-1}],\ (B_{t-1}, B_{t-2}],\ \ldots,\ (B_2, B_1],$$

with $L = B_t < B_{t-1} < \cdots < B_1 = R$. Using the fact that X is non-increasing, we can binary search for each endpoint B_i to find the smallest index j_i such that $x_{j_i} \leq B_i$, thereby computing how many entries fall into each interval. We then approximate all values in each interval using a representative (e.g., B_i).

In the general setting, we do not have direct access to x_j but only to estimates $\hat{x}_j$ satisfying error bounds $|\hat{x}_j - x_j| \leq T_j$. These estimates may break monotonicity, so we design the interval endpoints to ensure robustness. Specifically, we guarantee that if $x_j \in (B_{i+1}, B_i]$, then $\hat{x}_j$ lies in the same or a neighboring interval, provided that $T_j < \min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\}$. Moreover, we show (in Lemma 5.1) that binary search over the noisy sequence $\hat{X}$ still preserves a weak monotonicity: for any two keys $B < B'$, their corresponding indices j_B and $j_{B'}$ satisfy $j_B \geq j_{B'}$, maintaining consistency of bucket counts. This then ensures that using B_i to approximate x_j introduces only a small error (see Lemma 5.4).

Finally, we must carefully choose the bucket endpoints. If we use too few, the approximation within each bucket may be inaccurate; if we use too many, we risk excessive running time and reduced robustness due to estimation noise. By striking the right balance, we can ensure both accuracy and efficiency in approximating the entire sequence.

Back to Approximating $\mathrm{cost}(G)$ We now apply the above binary search strategy to estimate the clustering cost $\mathrm{cost}(G)$, using the estimated sequence $(\hat{c}_1,\ldots,\hat{c}_W)$.

As discussed, choosing appropriate bucket endpoints is non-trivial in this setting. We design the interval endpoints to align with the error bounds for $\hat{c}_j$, which satisfy $|\hat{c}_j - c_j| \le T_j := O\left(\varepsilon \cdot \max\left\{\frac{n}{\sqrt{W}}, c_j\right\}\right)$. To handle this, we partition the range $[1,n]$ into three subranges – $[1,\varepsilon n/\sqrt{W})$, $[\varepsilon n/\sqrt{W}, n/\sqrt{W})$, and $[n/\sqrt{W},n]$ – and construct arithmetic and geometric sequences separately of bucket endpoints B_i within each. This allows us to use only $O(\log W/\varepsilon)$ buckets while still maintaining the desired approximation guarantees.

For each bucket, we use binary search to determine the number of $\hat{c}_j$ values it contains, requiring $O(\log W)$ accesses per bucket. Thus, the total number of $\hat{c}_j$ queries is $O\left(\frac{\log^2 W}{\varepsilon}\right)$, and the total running time for estimating $\mathrm{cost}(G)$ is this quantity multiplied by the cost of estimating a single $\hat{c}_j$, resulting in $\tilde{O}(\sqrt{W}d/\varepsilon^3)$ time.

Remark It is tempting to apply our binary search strategy to approximate the weight of MST, which is $\mathrm{cost}(\mathrm{MST}) = n - W + \sum_{j=1}^{W-1} c_j$. Indeed, we can reuse the binary search framework to ensure that we only need to access $O\left(\frac{\log^2 W}{\varepsilon}\right)$ estimates $\hat{c}_j$ of the true values c_j. However, to ensure a $(1+\varepsilon)$-approximation of the MST weight, each $\hat{c}_j$ must approximate c_j within an additive error of $\varepsilon \cdot \max\left\{\frac{n}{W}, c_j\right\}$, which requires $\tilde{O}\left(\frac{Wd}{\varepsilon^2}\right)$ query time per estimate (Lemma 3.1). As a result, we do not achieve an asymptotic improvement over the original algorithm of [CRT05]. This limitation is not unexpected, since [CRT05] established a lower bound of $\Omega\left(\frac{Wd}{\varepsilon^2}\right)$ queries for approximating the MST weight.

1.2.2 Succinct Representation of the Profile Vector

We now present our approach for efficiently constructing a succinct representation of the estimated profile vector $(\widehat{\mathrm{cost}}_1,\ldots,\widehat{\mathrm{cost}}_n)$, which approximates the true clustering cost profile $(\mathrm{cost}_1,\ldots,\mathrm{cost}_n)$.

To do so, we first leverage the relationship between the number of edges in an MST and the number of connected components, deriving an equivalent formula for the k-clustering cost, in Eq. (3):

$$\mathrm{cost}_k = \sum_{i=1}^{n-k} w_i = n + \sum_{j=1}^{w_{n-k}-1} c_j - k \cdot w_{n-k}.$$

The central challenge is that, for an arbitrary k, w_{n-k} – the $(n-k)$-th smallest weight in the MST – is difficult to determine. Then we observe that for any weight $j \in \{1,\ldots,W\}$, the number of edges in the MST of weight at most j can be computed as $n - c_j$. Thus, if $w_{n-k} = j$, we can compute cost_k accordingly. In particular, for any $j \in \{1,\ldots,W\}$, if $k = c_j$, then we have $\mathrm{cost}_k = n + \sum_{i=1}^{j-1} c_i - c_j \cdot j$, as stated in Lemma 6.6.

In our binary search strategy, we partition the c_j values into intervals and use an endpoint of each interval to represent all c_j's within it. This suggests a natural way to estimate cost_{c_j}: for each c_j, we use a representative value $\overline{\mathrm{cost}}_{B_i}$ to estimate it, as stated in Eq. (4), where B_i is the interval endpoint corresponding to c_j.

Furthermore, we note that the profile vector $(\mathrm{cost}_1,\ldots,\mathrm{cost}_n)$ is monotonic in k, with k ranging from 1 to n. Therefore, for $k \in [B_{i+1}, B_i)$, the value cost_k lies within the range $(\overline{\mathrm{cost}}_{B_i}, \overline{\mathrm{cost}}_{B_{i+1}}]$. This allows us

to use $\overline{\text{cost}}_{B_{i+1}}$ as a representative for all cost_k values in that interval. Namely, we partition the index range $[1,n]$ into $O(\log W/\varepsilon)$ intervals. For each interval, we estimate the clustering cost at a chosen endpoint B_{i+1} as $\overline{\text{cost}}_{B_{i+1}}$ and use $\widehat{\text{cost}}_k = \overline{\text{cost}}_{B_{i+1}}$ to approximate all cost_k for k within the interval.

We provide an accuracy guarantee for the estimate of cost_k for any fixed k (see Lemma 6.6). The idea is as follows. Observe that $|\widehat{\text{cost}}_k - \text{cost}_k| = |\overline{\text{cost}}_{B_{i+1}} - \text{cost}_k| \le |\overline{\text{cost}}_{B_{i+1}} - \text{cost}_{c_{j_{i+1}}}| + |\text{cost}_{c_{j_{i+1}}} - \text{cost}_k|$, where j_{i+1} is the smallest index such that $\hat{c}_{j_{i+1}} \le B_{i+1}$. We bound these two terms separately. First, for each interval, B_{i+1} is close to $c_{j_{i+1}}$, so the gap $|\overline{\text{cost}}_{B_{i+1}} - \text{cost}_{c_{j_{i+1}}}|$ is small. Second, since both k and $c_{j_{i+1}}$ are constrained by nearby interval endpoints, the gap $|\text{cost}_{c_{j_{i+1}}} - \text{cost}_k|$ is also small. Consequently, the error $|\widehat{\text{cost}}_k - \text{cost}_k|$ is bounded.

We show in Theorem 1.2 that the sum of these errors over all $k \in [n]$ accounts to only an additive error of $\varepsilon \cdot \text{cost}(G) = \varepsilon \sum_{k=1}^{n} \text{cost}_k$, so that, *on average*, each estimator of cost_k achieves a $(1+\varepsilon)$-approximation.

1.2.3 More Accurate Estimation in the Similarity Setting

To extend our approach to the similarity setting, which in turn is more closely related to the maximum spanning tree, we first derive an equivalent formula for the clustering cost in Theorem 7.1:

$$\text{cost}^{(s)}(G) = \sum_{j=1}^{W} \frac{(c_j^{(s)} + n - 1)(n - c_j^{(s)})}{2},$$

where $c_j^{(s)}$ denotes the number of connected components in the subgraph induced by edges with weight *at least* j.

To $(1+\varepsilon)$-approximate $\text{cost}^{(s)}(G)$, it suffices to approximate each $c_j^{(s)} + n - 1$ within an additive error εn and approximate each $n - c_j^{(s)}$ within an additive error $\max\{\varepsilon n/W, \varepsilon(n - c_j^{(s)})\}$. The former can be done in expected $\tilde{O}(d/\varepsilon^2)$ time. However, estimating $n - c_j^{(s)}$ is technically more subtle.

Our earlier algorithm for estimating the number of connected components (Algorithm 1) achieves only an additive error of $\max\{\varepsilon n/W, \varepsilon c_j^{(s)}\}$ in expected $\tilde{O}(Wd/\varepsilon^2)$ time. But when $c_j^{(s)} > n/2$, the error of Algorithm 1 is greater than $\varepsilon(n - c_j^{(s)})$, which is insufficient for our purposes. To resolve this, we observe that when $c_j^{(s)}$ is large, most vertices are isolated. Therefore, we separately estimate the number of isolated vertices, along with the number of connected components among the non-isolated portion of $G_j^{(s)}$.

This refinement allows us to estimate $n - c_j^{(s)}$ within an additive error of

$$\max\{\varepsilon n/W, \varepsilon \min\{c_j^{(s)}, n - c_j^{(s)}\}\},$$

running in $\tilde{O}(Wd/\varepsilon^2)$ time, as stated in Lemma 7.4. Finally, we apply binary search to the sequence of $n - c_j^{(s)}$ estimates, and design the interval endpoints to align with the error bounds for $n - c_j^{(s)}$. Since these error bounds are more complex than those in the distance case, careful consideration is required when choosing the endpoints. In particular, we partition the range $[0, n-1]$ into five subranges – $[0, \varepsilon n/W)$, $[\varepsilon n/W, n/W)$, $[n/W, n/2)$, $[n/2, n - n/W)$, and $[n - n/W, n-1]$ – and construct bucket endpoints B_i according to either arithmetic or geometric progressions within each subrange. This allows us to use only $O(\log W/\varepsilon)$ buckets while still maintaining the desired approximation guarantees. We show in Theorem 1.4 that the sum of the corresponding products of estimators for $c_j^{(s)} + n - 1$ and $n - c_j^{(s)}$ yields a good approximation to $\text{cost}^{(s)}(G)$.

1.2.4 On the Lower Bounds and Metric Space Setting

The Lower Bounds Our lower bounds build on the distributional problem of distinguishing between two biased coin distributions, as studied in [CRT05]. Leveraging the hardness of this problem, we construct two graph distributions such that, with high probability, their total clustering costs $\text{cost}(G)$ differ by at least a factor of $1+\varepsilon$. We then show that any algorithm making only $o(\sqrt{W}d/\varepsilon^2)$ queries cannot distinguish between graphs drawn from these two distributions. The construction closely follows the MST lower bound from [CRT05], but uses different edge weights. The lower bound for approximating the similarity-based clustering cost $\text{cost}^{(s)}(G)$ is established using a similar argument.

The Metric Space Setting In this setting, our approach closely follows the work of Czumaj and Sohler on estimating the MST weight in metric spaces [CS09]. We begin by expressing the clustering cost as a quadratic function of the number of connected components across $\tilde{O}(\log n/\varepsilon)$ subgraphs. To estimate the number of connected components, we leverage the algorithm of [CS09], which runs in $\tilde{O}(n/\varepsilon^6)$ time and identifies a set of *representative vertices* – vertices whose neighborhoods are shared by many others – allowing the algorithm to inspect only the neighborhoods of these representatives.

Our contribution includes a refined error analysis for estimating the number of connected components, as well as new lower bounds on $\text{cost}(G)$ in terms of the number of representative vertices and connected components. Using these insights, we show that directly applying the cost formula to the component estimates yields a $(1+\varepsilon)$-approximation of the clustering cost, for both distance- and similarity-based metrics.

1.3 Related Work

Hierarchical clustering has been extensively studied in the context of approximation algorithms with polynomial running times. A significant body of work has explored hierarchical clustering in metric and Euclidean spaces, including studies on hierarchical k-median [LNRW10] and the use of the largest cluster radius as a cost measure [DL05]. Recently, [Das16] introduced a cost function for similarity-based hierarchical clustering and proposed an approximation algorithm for this cost. Several improvements and generalizations have since emerged [CC17, MW23, CAKMTM19, CCN19]. Other recent work on hierarchical correlation clustering and fitting distances in ultrametrics, such as [AKLL25], has further improved approximation algorithms in the hierarchical setting.

In terms of sublinear algorithms for hierarchical clustering, [AKLP22] studied such algorithms concerning Dasgupta cost [Das16] within dynamic streaming, query, and massively parallel computation models. [BBD$^+$17] also considered affinity and single-linkage clustering in the massively parallel computation setting. [ACM$^+$22] focused on streaming algorithms for identifying a hierarchical clustering with low Dasgupta cost and estimating the value of the optimal hierarchica l tree. Additionally, [KKLM23] provided a sublinear-time hierarchical clustering oracle for graphs exhibiting significant flat clustering, and [KKL$^+$25] estimated Dasgupta cost for well-clusterable graphs in sublinear-time. Other sublinear algorithms addressing graph clustering with conductance-based measures include local graph clustering [ST13, ACL06] and spectral clustering oracles for well-clusterable graphs [Pen20, GKL$^+$21, SP23].

Our work is closely related to a series of studies on estimating the number of connected components and the weight of minimum spanning trees. [CRT05] pioneered a sublinear-time algorithm for these problems using sampling and truncated BFS. Subsequent investigations have explored these problems in various contexts, including Euclidean space [CEF$^+$05], metric space [CS09], and graph streaming [HP19, PS18].

Organization The rest of the paper is organized as follows. We present preliminaries in Section 2, and describe our connected component estimation algorithm and give its analysis in Section 3. In Section 4, we express the total clustering cost $\mathrm{cost}(G)$ via connected components. We then give our binary search strategy for succinctly approximating noisy monotone sequence in Section 5. In Section 6, we prove Theorem 1.1 for distance-based clustering, and Theorem 1.2. We extend our approach to the similarity setting in Section 7, introducing a new algorithm and proving Theorem 1.4 and Theorem 1.5. Lower bounds for both settings are given in Section 8, and experimental validation appears in Section 9. Additional proofs, generalizations, metric space results, and further experiments are deferred to the appendix for clarity.

2 Preliminaries

Model of Computation We will assume that the algorithm is given access to an adjacency list representation of the (undirected and connected) input graph $G = (V, E)$, except when G is a metric graph. That is, we assume w.l.o.g. that $V = \{1, \ldots, n\}$ and that the parameter n is given to the algorithm. The edges of the graph are stored in an array of size n whose i-th entry is a pointer to the list of neighbors of vertex i. With each neighbor j that appears in the adjacency list of vertex i, we store the weight $w((i, j))$. In particular, computing the degree $\deg(v)$ of a vertex v requires scanning through all of its neighbors and so this can be done in $O(\deg(v))$ time. We will use d to denote the average vertex degree of G. We emphasize that the algorithm has the ability to query the i-th neighbor of a specified vertex v in $\Theta(i)$ time, for any given $i \leq \deg(v)$.

Probability Amplification Consider an algorithm A which outputs out_A, with the assumption that out_A is correct with constant probability greater than $2/3$. We can amplify the success probability by constructing a new algorithm A^* that runs $C \cdot \log(1/\delta)$ independent instances of A, for some sufficiently large constant $C > 0$. The output out_{A^*} is then set to be the median of the output values from these instances. By Chernoff bound it follows that out_{A^*} is correct with probability at least $1 - \delta$, where δ can be made arbitrarily small (see e.g. [MR95]).

Chernoff–Hoeffding bound We will make use of the following Chernoff–Hoeffding bound (see Theorem 4.4 and Theorem 4.5 in [MU17]).

Theorem 2.1 (The Chernoff–Hoeffding bound)**.** *Let $t \geq 1$. Let $X = \sum_{1 \leq i \leq t} X_i$, where $X_i, 1 \leq i \leq t$, are independently distributed in $[0, 1]$. Then for all $0 < \varepsilon \leq 1$,*

$$\Pr[X \geq (1+\varepsilon)\,\mathbf{E}[X]] \leq e^{-\mathbf{E}[X]\varepsilon^2/3}, \ \Pr[X \leq (1-\varepsilon)\,\mathbf{E}[X]] \leq e^{-\mathbf{E}[X]\varepsilon^2/2}.$$

3 Estimating the Number of Connected Components

To estimate the cost of single-linkage clustering, we first need an algorithm that efficiently approximates the number of connected components in a subgraph H of a graph G. This approximation serves as a crucial step in our overall methodology and builds upon an algorithm to approximate the number of connected components and the cost of a minimum spanning tree from [CRT05]. We give the formal description in Algorithm 1. We start by considering a slight modification of an algorithm from [CRT05] to approximate the number of connected components and we give an improved analysis for the case that their number is relatively small. We remark that the edges of subgraph H are implicitly defined (as a function of the weight

of the edge in G and its vertices; for example, it could be the set of edges above or below a certain weight), so we must inspect *all* edges incident to a vertex v in G to find its neighbors in H.

Algorithm 1: APPROXIMATECONNECTEDCOMPONENTS(G, H, ε, k, d)

input : graph G, implicit subgraph H, approx. parameter ε, threshold parameter k, avg. degree d
output: an estimate $\hat{c}$ of the number of connected components in H

choose $r = \lceil 64k/\varepsilon^2 \rceil$ vertices $u_1, \ldots, u_r$ uniformly at random
set threshold $\Gamma = \lceil 4k/\varepsilon \rceil$ and $d^{(G)} = d \cdot \Gamma$
for *each sampled vertex* u_i **do**
    set $\beta_i = 0$
    take the first step of BFS: identify neighbors of u_i in H by examining all incident edges in G
    let $d_{u_i}^{(G)}$ be the degree of u_i in G
    if u_i *is isolated in* H **then**
        set $\beta_i = 1$
    else
        (*) flip a coin
        if *(heads) & (# vertices visited* $\in H < \Gamma$ *during the BFS) & (no visited vertex* $\in H$ *has degree in* $G > d^{(G)}$ *during the BFS)* **then**
            resume BFS on H, doubling the number of visited edges in G
            if *this allows BFS on* H *to explore the whole connected component of* u_i **then**
                set $\beta_i = d_{u_i}^{(G)} \cdot 2^{\#\text{ coin flips}} / \#$ edges visited in G
            else
                go to (*)
return $\hat{c} = \frac{n}{r} \sum_{i=1}^{r} \beta_i$

The underlying intuition is as follows. For a vertex u in a connected component C_u of subgraph H, let $\mathrm{vol}(C_u) = \sum_{v \in C_u} d_v^{(G)}$, where $d_v^{(G)}$ is degree of v in graph G. The sum $\sum_{u \in C_u} \frac{d_u^{(G)}}{\mathrm{vol}(C_u)} = 1$ for each component C_u. Therefore, across all vertices in V, we have: $\sum_{u \in V} \frac{d_u^{(G)}}{\mathrm{vol}(C_u)} = c$, where c is the total number of connected components in subgraph H. By sampling and averaging estimates β_i for sampled vertices, we approximate the value of c.

At this point, we provide a lemma to establish a theoretical guarantee for Algorithm 1.

Lemma 3.1. *Let* $1 > \varepsilon > 0$ *and* $k \geq 1$. *Suppose (an upper bound on) the average degree* d *of graph* G *is known. Given access to the graph* G *and an implicit subgraph* $H \subseteq G$ *(where* H *shares the same vertex set as* G*, and the edges of* H *are determined by evaluating conditions on the edges of* G*), Algorithm 1 outputs* $\hat{c}$ *such that*

$$|\hat{c} - c| \leq \varepsilon \cdot \max\left\{\frac{n}{k}, c\right\},$$

with probability at least $7/8$*, where* c *is the number of connected components in* H*. The query complexity and running time of the algorithm are* $O(\frac{k}{\varepsilon^2} d \log(\frac{k}{\varepsilon} d))$ *in expectation.*

Before proving Lemma 3.1, consider the following lemma, adapted from [CRT05].

Lemma 3.2 ([CRT05]). *Let U be the set of vertices that lie in components in subgraph H with fewer than Γ vertices; and all of these vertices in the original graph are of degree at most $d^{(G)}$. The random variable β_i in Algorithm 1 is given by:*

$$\beta_i = \begin{cases} 0, & \text{if } u_i \notin U \\ 2^{\left\lceil \log\left(\frac{\mathrm{vol}(C_{u_i})}{d_{u_i}^{(G)}}\right)\right\rceil} \frac{d_{u_i}^{(G)}}{\mathrm{vol}(C_{u_i})}, & \text{w.p. } 2^{-\left\lceil \log\left(\frac{\mathrm{vol}(C_{u_i})}{d_{u_i}^{(G)}}\right)\right\rceil}, \text{ if } \mathrm{vol}(C_{u_i}) \neq 0 \\ 1, & \text{w.p.1, if } \mathrm{vol}(C_{u_i}) = 0 \\ 0, & \text{otherwise} \end{cases}$$

The expectation and variance of β_i is $\mathbf{E}[\beta_i] = c_U$ and $\mathrm{Var}[\beta_i] \leq \frac{2c_U}{n}$.

Proof. We let H_{nis} be the subgraph of H induced by all non-isolated vertices w.r.t. H. Note that if a vertex $u \in H \setminus H_{\mathrm{nis}}$, then u is isolated in H. The number of vertices in H_{nis} is defined as n', and let c'_U denote the number of connected components in $H_{nis}[U]$. For each $1 \leq i \leq r$, the expectation of β_i is given by conditional expectation:

$$\begin{aligned} \mathbf{E}[\beta_i] &= \Pr[u_i \in H_{nis}]\,\mathbf{E}[\beta_i | u_i \in H_{nis}] + \Pr[u_i \in H \setminus H_{nis}]\,\mathbf{E}[\beta_i | u_i \in H \setminus H_{nis}] \\ &= \frac{n'}{n} \cdot \frac{1}{n'} \left(\sum_{u \in H_{nis} \setminus U} 0 + \sum_{u \in H_{nis} \cap U} 2^{-\left\lceil \log\left(\frac{\mathrm{vol}(C_u)}{d_u^{(G)}}\right)\right\rceil} \cdot 2^{\left\lceil \log\left(\frac{\mathrm{vol}(C_u)}{d_u^{(G)}}\right)\right\rceil} \frac{d_u^{(G)}}{\mathrm{vol}(C_u)} \right) + \frac{n-n'}{n} \cdot 1 \\ &= \frac{1}{n} \cdot c'_U + \frac{1}{n}(n-n') && \text{(since } \textstyle\sum_{u \in U} \frac{d_u^{(G)}}{\mathrm{vol}(C_u)} = c_U\text{)} \\ &= \frac{1}{n} \cdot c'_U + \frac{1}{n}(c_U - c'_U) && \text{(since } n - n' = c_U - c'_U \text{ is the number of isolated vertices)} \\ &= \frac{c_U}{n} \end{aligned}$$

Since when $\mathrm{vol}(C_{u_i}) \neq 0$, $\beta_i \leq 2^{\log\left(\frac{\mathrm{vol}(C_{u_i})}{d_{u_i}^{(G)}}\right)+1} \frac{d_{u_i}^{(G)}}{\mathrm{vol}(C_{u_i})} \leq 2$, we have $\mathrm{Var}[\beta_i] \leq \mathbf{E}[\beta_i^2] \leq 2 \cdot \mathbf{E}[\beta_i] \leq \frac{2c_U}{n}$. □

Now we are ready to prove Lemma 3.1.

Proof of Lemma 3.1. Let U be the set of vertices defined in Lemma 3.2, and let c_U be the number of connected components in the vertex-induced subgraph $G[U]$. The number of connected components containing vertices with degree greater than $d^{(G)}$, is at most $\frac{n \cdot d}{d^{(G)}} = \frac{n}{\Gamma}$. Furthermore, the number of connected components with size greater than Γ is at most $\frac{n}{\Gamma}$. Then we have that,

$$c - \frac{2n}{\Gamma} \leq c_U \leq c.$$

By Lemma 3.2, $\mathbf{E}[\beta_i] = \frac{c_U}{n}$ and $\mathrm{Var}[\beta_i] \leq \frac{2c_U}{n}$. Thus, $\mathbf{E}[\hat{c}] = \frac{n}{r} \cdot r \cdot \mathbf{E}[\beta_i] = c_U$, leading to: $c - \frac{2n}{\Gamma} \leq \mathbf{E}[\hat{c}] \leq c$. Furthermore, $\mathrm{Var}[\hat{c}] = \frac{n^2}{r^2} \cdot r \cdot \mathrm{Var}[\beta_1] \leq \frac{n^2}{r} \cdot \frac{2c_U}{n} \leq \frac{2nc}{r}$.

We bound the error of $\hat{c}$ by two cases: $c < \frac{n}{k}$, and $c \geq \frac{n}{k}$. If $c < \frac{n}{k}$, by Chebyshev's inequality,

$$\Pr[|\hat{c} - c_U| \geq \frac{\varepsilon n}{2k}] \leq \frac{\mathrm{Var}[\hat{c}] \cdot 4k^2}{\varepsilon^2 n^2} \leq \frac{2nc \cdot 4k^2}{r \cdot \varepsilon^2 n^2} \leq \frac{8c \cdot k^2}{64k \cdot n} \leq \frac{1}{8}$$

The last inequality holds as $c < \frac{n}{k}$. Therefore, the error of $\hat{c}$ is

$$|\hat{c}-c| \le |\hat{c}-c_U| + |c_U - c| \le \frac{\varepsilon n}{2k} + \frac{2n}{\Gamma} \le \frac{\varepsilon n}{2k} + \frac{\varepsilon n}{2k} = \frac{\varepsilon n}{k}$$

Else, when $c \ge \frac{n}{k}$,

$$\Pr[|\hat{c}-c_U| \ge \frac{\varepsilon c}{2}] \le \frac{4\mathrm{Var}[\hat{c}]}{\varepsilon^2 c^2} \le \frac{8nc}{r\cdot\varepsilon^2 c^2} \le \frac{8n}{64k\cdot c} \le \frac{1}{8}$$

The last inequality holds as $c \ge \frac{n}{k}$. And the error of $\hat{c}$ is

$$|\hat{c}-c| \le |\hat{c}-c_U| + |c_U - c| \le \frac{\varepsilon c}{2} + \frac{2n}{\Gamma} = \frac{\varepsilon c}{2} + \frac{\varepsilon n}{2k} \le \varepsilon c$$

Combining both cases, we have that $|\hat{c}-c| \le \varepsilon \max\{\frac{n}{k}, c\}$, with probability more than $\frac{7}{8}$.

Running time analysis. Denote $\mathbf{E}[T]$ as the expected running time of Algorithm 1, and $\mathbf{E}[T(u)]$ as the expected running time of BFS when vertex u is sampled. Then,

$$\mathbf{E}[T] = O(r)\frac{1}{n}\sum_{u\in V}\mathbf{E}[T(u)],$$

as we sampled r vertices, each vertex $u \in V$ is sampled with probability $\frac{1}{n}$.

Note that by our truncation, there are at most Γ vertices visited in BFS, and each of them has degree less than $d^{(G)} = O(d\cdot\Gamma)$. Therefore, at most $O(\Gamma^2 d)$ edges are visited in BFS, the number of coin flips is at most $O(\log(\Gamma^2 d)) = O(\log(\Gamma d))$.

If a sampled vertex u flips coins for t times, the running time of BFS is $(d_u + 2d_u + \cdots + 2^{t-1}\cdot d_u) \le 2^t d_u$. The coin is flipped at most $O(\log(\Gamma d))$ times, so the expected time of BFS is

$$\mathbf{E}[T(u)] \le \sum_{t=1}^{O(\log(\Gamma d))} 2^{-t}\cdot 2^t d_u = O(d_u \log(\Gamma d))$$

So the expected running time of Algorithm 1 is $O(r\cdot\frac{1}{n}\sum_{u\in V}[d_u\log(\Gamma d)]) = O(r\cdot d\log(\Gamma d)) = O(\frac{k}{\varepsilon^2}d\log(\frac{k}{\varepsilon}d))$. □

Note that [CRT05] achieves an additive error of $\varepsilon\cdot n$, with a running time that is quadratic in ε. Utilizing their approach to attain an additive error of $\varepsilon\frac{n}{k}$ would require a running time that is quadratic in k. In contrast, our method achieves this with a running time that scales linearly with k in the regime when $n/k \ge c$. By amplifyingAlgorithm 1 we obtain Algorithm 2 that satisfies the following corollary.

Algorithm 2: APPNCCMEDIANTRICK$(G, H, \varepsilon, k, d, \delta)$

Let $C > 0$ be a sufficiently large constant
for $i = 1, \ldots, C\cdot\log(1/\delta)$ **do**
    invoke i-th independent instance of Algorithm 1 and get output out_i
return $\hat{c}$, which is the median of $\mathrm{out}_1, \ldots, \mathrm{out}_k$

Corollary 3.3. *Let $\varepsilon, \delta \in (0,1)$ and $k \geq 1$. Suppose (an upper bound on) the average degree d of graph G is known. Given access to the graph G and an implicit subgraph $H \subseteq G$ (where H shares the same vertex set as G, and the edges of H are determined by evaluating conditions on the edges of G), Algorithm 2 outputs $\hat{c}$ satisfying*

$$|\hat{c} - c| \leq \varepsilon \cdot \max\left\{\frac{n}{k}, c\right\},$$

with probability at least $1-\delta$. Here, c is the number of connected components in the subgraph H. The algorithm runs in time $O(\frac{k\log(1/\delta)}{\varepsilon^2} d \log(\frac{k}{\varepsilon}d))$ in expectation.

4 Cost Formula for Distance-Based Clustering

Recall that the cost of a hierarchical SLC is $\text{cost}(G) = \sum_{i=1}^{n-1}(n-i)\cdot w_i$, where w_i is the i-th smallest edge weight in the MST. Similarly to the work on approximating the cost of an MST [CRT05], we will express $\text{cost}(G)$ as a function of the number of connected components in certain subgraphs of G. For this purpose, let G_j denote the subgraph induced by edges of weight at most j and let c_j denote the number of connected components in G_j. In particular, we let G_0 denote the graph consisting of n singleton vertices, so $c_0 = n$. We assume the graph G is connected, so that the top cluster (corresponding to 1-SLC) contains all vertices.

We assume that the edge weights of G are integers. If this is not the case, one may rescale the weights by a factor of $\approx 1/\varepsilon$ and round them to their closest integer. We refer to Section A.2 for details.

Theorem 4.1. *Let G be a connected graph on n vertices, with edge weights from $\{1,\ldots,W\}$. Then*

$$\text{cost}(G) = \frac{n(n-1)}{2} + \frac{1}{2}\cdot\sum_{j=1}^{W-1}(c_j^2 - c_j).$$

Proof. Let n_j be the number of edges in the MST with weight j. Observe that the number of edges in the MST with weight 1 is $n_1 = n - c_1 = c_0 - c_1$ and the number of edges with weight 2 is $n_2 = c_1 - c_2$, …, the number of edges with weight j is $n_j = c_{j-1} - c_j$ for any integer $j \in \{1,\ldots,W\}$. By our assumption that the graph G is connected, $c_W = 1$. Then we have,

$$\begin{aligned}
\text{cost}(G) &= \sum_{i=1}^{n-1}(n-i)\cdot w_i && \text{(by Eq. (1))}\\
&= \sum_{i=1}^{n_1}(n-i)\cdot 1 + \sum_{i=n_1+1}^{n_1+n_2}(n-i)\cdot 2 + \ldots + \sum_{i=n_1+\ldots+n_{W-1}+1}^{n_1+\ldots+n_W}(n-i)\cdot W \\
&&\text{(reorganize the sum by grouping the contributions by edge weights)}\\
&= \sum_{i=1}^{n-c_1}(n-i)\cdot 1 + \sum_{i=n-c_1+1}^{n-c_2}(n-i)\cdot 2 + \cdots + \sum_{i=n-c_{W-1}+1}^{n-c_W}(n-i)\cdot W && \text{(since } n_j = c_{j-1}-c_j\text{)}\\
&= \sum_{i=1}^{n-c_W}(n-i) + \sum_{i=n-c_1+1}^{n-c_2}(n-i)\cdot 1 + \cdots + \sum_{i=n-c_{W-1}+1}^{n-c_W}(n-i)\cdot(W-1) \\
&&\text{(factoring out } (n-i) \text{ from each summation and combining the terms)}\\
&= \sum_{i=1}^{n-c_W}(n-i) + \sum_{i=n-c_1+1}^{n-c_W}(n-i) + \cdots + \sum_{i=n-c_{W-1}+1}^{n-c_W}(n-i) \\
&&\text{(repeating this decomposition until all summations end with } i = n-c_W\text{)}
\end{aligned}$$

$$= \sum_{i=1}^{n-1} i + \sum_{i=1}^{c_1-1} i + \cdots + \sum_{i=1}^{c_{W-1}-1} i \qquad \text{(substituting } (n-i) \text{ with } i\text{, and noting that } c_W = 1\text{)}$$

$$= \sum_{j=0}^{W-1} \frac{(1+c_j-1)\cdot(c_j-1)}{2} \qquad \text{(noting that } c_0 = n\text{)}$$

$$= \frac{n(n-1)}{2} + \frac{1}{2} \cdot \sum_{j=1}^{W-1} (c_j^2 - c_j)$$

□

5 Problem Abstraction and Binary Search

As mentioned earlier, our goal is to efficiently approximate the total clustering cost $\text{cost}(G)$, which reduces to estimating the sum $\sum_{j=1}^{W-1}(c_j^2 - c_j)$. A key observation is that the sequence $(c_1, \ldots, c_W)$ is non-increasing: as j increases, the subgraph G_j includes more edges, leading to fewer connected components. This monotonicity suggests the possibility of summarizing the sequence succinctly using only a few representative values. However, we only have access to estimated values $\hat{c}_j$, and estimation errors may disrupt the monotonicity. This motivates us to study an abstract version of the problem: how to efficiently approximate the sum over a non-increasing sequence when only noisy estimates are available. This abstraction also arises in our algorithm for similarity-based clustering, and we believe it could be of independent interest in other sublinear or local computation settings.

Now, let us describe this abstract problem in a bit more formal manner: Given a noisy version of a *non-increasing* sequence X with elements $(x_1, x_2, \ldots, x_W)$, and each element x_j from $[L, R]$ (where L and R are lower and upper bounds for the smallest and largest element of X), we want to find a succinct approximation of every x_j without approximating each x_j separately.

Initially, assume the exact values of X are known. We divide the range $[L, R]$ into $t-1$ intervals, for some integer t to be chosen later. The intervals are specified by some numbers $L = B_t < B_{t-1} < \cdots < B_1 = R$. Specifically, we divide $[L, R]$ into:

$$[B_t, B_{t-1}], (B_{t-1}, B_{t-2}], \ldots, (B_2, B_1]$$

Then to get our estimates we do the following. For each $i \in \{1, \ldots, t\}$ we perform binary search on the sequence $X' = (x_1, \ldots, x_W, L)$ to find the smallest index $j_i \in \{1, \ldots, W+1\}$, $1 \le i \le t-1$ such that $x_{j_i} \le B_i$ and where we define $j_t = W+1$. This results in a non-decreasing sequence $(j_1, j_2, \ldots, j_t)$ that partitions $\{1, \ldots, W\}$ sets into $J_i = \{j_i, \ldots, j_{i+1} - 1\}, 1 \le i \le t-1$ (where the set is empty when $j_i = j_{i+1}$). Then every x_j with $j \in J_i$ is estimated by B_i.

Now consider that we only have access to estimates $\hat{x}_j \in [L, R]$ of the x_j in X. In particular, we assume that there is a sequence $T = (T_1, \ldots, T_W)$ of error bounds T_j, such that $|\hat{x}_j - x_j| \le T_j$. To find succinct approximation of all entries in the sequence X, we can still apply the approach sketched above and perform a variant of the standard binary search on the sequence $\widehat{X}' = (\hat{x}_1, \ldots, \hat{x}_W, L)$ corresponding to the estimates, searching for each interval endpoint B_i. For completeness, we give the pseudocode in Algorithm 3. The algorithm is initialized with $\ell = 1$, $r = W+1$, $B = B_i$, and sequence $\hat{X}'$. We note that, since we only have access to the estimates $\hat{x}_j$, the sequence $\widehat{X}' = (\hat{x}_1, \ldots, \hat{x}_W, L)$ may not be non-increasing. Furthermore, we note that Algorithm 3 always returns the index that is reached at the end of the recursion when ℓ and r become equal. We show that our binary search approach satisfies a certain monotonicity property, namely,

Algorithm 3: BINARYSEARCH($\widehat{X}', \ell, r, B$)

input : Estimated array $\widehat{X}'$, leftmost and rightmost indices ℓ and r, search key B
output: Index i

```
if ℓ = r then
    return ℓ
else
    m = ⌊(ℓ+r)/2⌋
    if x̂_m ≤ B then
        return BINARYSEARCH(X̂', ℓ, m, B)
    else
        return BINARYSEARCH(X̂', m+1, r, B)
```

that the indices $\hat{j}_i$ found by the searches for the interval endpoints B_i are in non-decreasing order. This is stated formally in the following lemma.

Lemma 5.1. *Let $a, b \in [L,R]$ with $a < b$. Let $\hat{j}_a$ and $\hat{j}_b$ be the two indices returned by Algorithm 3 on a (possibly unsorted) input sequence $\widehat{X}'$ with $B = a$ and $B = b$, respectively. Then we have $\hat{j}_a \geq \hat{j}_b$.*

Proof. As a thought experiment for the proof, perform a binary search for $B = a$ and $B = b$ in parallel. During binary search, the only place where we are using B is Algorithm 3 where we compare to the value $\hat{x}_m$. If $\hat{x}_m \leq a < b$ or $\hat{x}_m > b > a$, the comparison $\hat{x}_m \leq B$ in Algorithm 3 has the same outcome for both search keys a and b. The two searches behave differently only if $a < \hat{x}_m \leq b$. For a, the algorithm continues to search in the range $[m+1, r]$ which ensures that the returned index $\hat{j}_a$ is at least $m+1$. For b, setting the range to be $[\ell, m]$ ensures the returned index $\hat{j}_b$ is at most m. This guarantees that regardless of the cases encountered during the binary search, we always end up with $\hat{j}_a \geq \hat{j}_b$. □

We can apply the above lemma directly on the B_i to obtain the following corollary.

Corollary 5.2. *For $i \in \{1, \ldots, t-1\}$ let $\hat{j}_i$ be the index returned when Algorithm 3 is run on a (possibly unsorted) input sequence $\widehat{X}'$ with $B = B_i$ and let $\hat{j}_t = W + 1$. Then $\hat{j}_1 \leq \hat{j}_2 \leq \cdots \leq \hat{j}_t$.*

Given the above corollary, our plan is to simply search for the interval endpoints B_i and denote the corresponding resulting indices $\hat{j}_i$ and we always set $\hat{j}_t = W + 1$. If we then invoke Algorithm 3 with parameter $B = \hat{x}_j$ that satisfies $B_{i+1} \leq \hat{x}_j \leq B_i$ and let i^* be the index returned by our binary search procedure, then we know by Lemma 5.1 that $\hat{j}_i \leq i^* \leq \hat{j}_{i+1}$. Thus, it seems that we can simply estimate $\hat{x}_j$ by B_i and the error resulting from the binary search is T_j plus $B_i - B_{i+1}$. However, we may encounter a scenario where several consecutive $\hat{j}_i$ values are mapped to the same value, in which case it is not immediately clear that this approach works. We can define $\hat{J}_i = \{\hat{j}_i, \ldots, \hat{j}_{i+1} - 1\}, 1 \leq i \leq t-1$, as in the case of binary search with error. This ensures that the $\hat{J}_i$ form a partition of $\{1, \ldots, W\}$, but in the case that many $\hat{j}_i$ are mapped to the same value we could still potentially get a large error. In the following we show that this will not be the case. We will now define the notion of a valid discretization with respect to (w.r.t.) X and T that is a partition of $[L,R]$ into intervals, such that for every x_i, the error of the interval containing x_i is at most the minimum length of the neighboring intervals.

Definition 5.3. Let $X = (x_1, \ldots, x_W)$ be a non-increasing sequence with $x_j \in [L,R]$ for all $j \in \{1, \ldots, W\}$ and let $T = (T_1, \ldots, T_W)$ be a sequence of error bounds. Let $\widehat{X} = (\hat{x}_1, \ldots, \hat{x}_W)$, $\hat{x}_j \in [L,R]$ with $|\hat{x}_j - x_j| \leq T_j$. We say that a sequence of interval endpoints $L = B_t < B_{t-1} < \cdots < B_1 = R$ is a *valid discretization* of $[L,R]$ w.r.t. X and T, if for every $j \in \{1, \ldots, W\}$ and $i \in \{1, \ldots, t-1\}$ such that $B_{i+1} \leq x_j \leq B_i$ we have that

- $T_j < B_{i-1} - B_i$, if $i \geq 2$, and
- $T_j < B_{i+1} - B_{i+2}$, if $i \leq t-2$.

Intuitively, this ensures that whenever we search for a key $\hat{x}_j$ with $B_{i+1} \leq x_j \leq B_i$ we will end up in the set of indices $\hat{J}_i$ or a neighboring one.

For every $j \in \hat{J}_i$ we now define the estimator for x_j to be $\bar{x}_j = B_i$. In the following lemma, we formalize our intuition and show that the value of x_j is between B_{i+2} and B_{i-1} and thus $\bar{x}_j$ is close to x_j if our discretization is sufficiently fine.

Lemma 5.4. *Let $X = (x_1, \ldots, x_W)$ be a non-increasing sequence with $x_j \in [L,R]$ for all $j \in \{1, \ldots, W\}$ and let $T = (T_1, \ldots, T_W)$ be a sequence of error bounds. Let $\widehat{X} = (\hat{x}_1, \ldots, \hat{x}_W)$, where $\hat{x}_j \in [1,n]$ with $|\hat{x}_j - x_j| \leq T_j$ for any $j \in [W]$. Let $L = B_t < \cdots < B_1 = R$ be a sequence of interval endpoints that is a valid discretization of $[L,R]$ w.r.t. X and T. Let $\hat{j}_i$ be the index returned by Algorithm 3 on $\widehat{X}' = (\hat{x}_1, \ldots, \hat{x}_W, L)$ for search key $B = B_i$ and let $\hat{J}_i = \{\hat{j}_i, \ldots, \hat{j}_{i+1} - 1\}$ and define $\hat{j}_t = W + 1$. For any $1 \leq i \leq t-1$ and every $j \in \hat{J}_i$ we have*

$$B_{i+2} \leq x_j \leq B_{i-1},$$

where we define $B_0 = \infty$ and $B_{t+1} = -\infty$.

Proof. Let $i \in \{1, \ldots, t-1\}$. We will show that if $x_j > B_{i-1}$ then $x_j \notin \hat{J}_i$ and if $x_j < B_{i+2}$ then $x_j \notin \hat{J}_i$. This implies the theorem. Now define j^* to be the largest index such that $x_{j^*} > B_{i-1}$ (if no such x_{j^*} exists there is nothing to prove). Since the B_i form a valid discretization, we have for every $j \leq j^*$ with $B_{i'+1} \leq x_j \leq B_{i'}$ that

$$\hat{x}_j \geq x_j - T_j \geq B_{i'+1} - T_j > B_{i'+1} - (B_{i'+1} - B_{i'+2}) = B_{i'+2} \geq B_i$$

since $x_j \geq x_{j^*} > B_{i-1}$ and thus $i' + 1 \leq i - 1$. Now consider an invocation of Algorithm 3 with $B = B_i$. Whenever we compare an element $\hat{x}_m$, $m \leq j^*$, with $B = B_i$ we have $\hat{x}_m > B$ and we recurse with the sequence $\hat{x}_{m+1}, \ldots, \hat{x}_r$. Since we have $\hat{x}_{W+1} = L \leq B_i$ this implies that we always have $\hat{j}_i > j^*$ and so $j^* \notin \hat{J}_i$.

Now consider the smallest index j^* such that $x_{j^*} < B_{i+2}$ (again, if such an x_{j^*} does not exist, there is nothing to prove). Since the B_i's form a valid discretization, we have for every $j \geq j^*$ with $B_{i'+1} \leq x_j \leq B_{i'}$ that

$$\hat{x}_j \leq x_j + T_j \leq B_{i'} + T_j < B_{i'} + (B_{i'-1} - B_{i'}) = B_{i'-1} \leq B_{i+1}$$

since $x_j \leq x_{j^*} < B_{i+2}$ and thus $i' \geq i + 2$. Now consider an invocation of Algorithm 3 with $B = B_{i+1}$. Whenever we compare an element $\hat{x}_m$, $m \geq j^*$, with $B = B_{i+1}$ we have $\hat{x}_m \leq B$ and we recurse with the sequence $\hat{x}_\ell, \ldots, \hat{x}_m$. This implies that $j_{i+1} \leq j^*$. Combining both cases, we observe that there is no $j \in \hat{J}_i$ with $x_j > B_{i-1}$ or $x_j < B_{i+2}$. Hence the lemma follows. □

6 Sublinear Algorithms in Distance Case

We now give a sublinear time algorithm to estimate the SLC cost $\text{cost}(G)$ in the distance graph, with a $(1+\varepsilon)$-approximation factor. The straightforward approach is to estimate each c_j using Algorithm 1, and it

can be easily verified that this results in a $(1+\varepsilon)$-approximation. However, this method results in a running time that is linear in W. To improve on this, we leverage the non-increasing nature of c_j with the binary search technique given in Section 5 to achieve our goal with a more efficient running time. Once we obtain the algorithm for the SLC cost, we then build upon it to give a sublinear time algorithm for estimating the vector $(\text{cost}_1, \ldots, \text{cost}_n)$, which we define as the *SLC profile vector in the distance graph*.

6.1 From Binary Search to Clustering Cost Estimation

We will apply binary search to the sequence of estimates $\hat{c}_j$ of the number of connected components, where $\hat{c}_j$ and its error bound are defined in the following lemma.

Lemma 6.1. *For any $1 \le j \le W$, let $\hat{c}_j = \min\{\max\{\hat{c}'_j, 1\}, n\}$, where $\hat{c}'_j$ is the output of Algorithm 2 with input G, $H = G_j$, $\varepsilon/8$, $k = \sqrt{W}$, d, $\delta = 1/(4W)$. Then with probability at least $3/4$, it holds that for* all *$1 \le j \le W$, $\hat{c}_j \in [1,n]$, and*

$$\left|\hat{c}_j - c_j\right| \le T_j := \frac{\varepsilon}{8} \cdot \max\left\{\frac{n}{\sqrt{W}}, c_j\right\}.$$

Proof. By Corollary 3.3, for any fixed $j \in [W]$, it holds that with probability at least $1 - 1/(4W)$, $\left|\hat{c}'_j - c_j\right| \le \frac{\varepsilon}{8} \cdot \max\left\{\frac{n}{\sqrt{W}}, c_j\right\} = T_j$. Besides, rounding $\hat{c}'_j$ to $\hat{c}_j = \min\{\max\{\hat{c}'_j, 1\}, n\}$ guarantees that $\hat{c}_j \in [1,n]$. Furthermore, if $\hat{c}_j < 1$ or $\hat{c}_j > n$, this rounding process decreases the gap between $\hat{c}_j$ and c_j, and ensures the error remains within the error bound T_j.

Therefore, by the union bound, with probability at least $\frac{3}{4}$, for *all* $j \in [W]$, $\left|\hat{c}_j - c_j\right| \le T_j$, and $\hat{c}_j \in [1,n]$. □

Throughout the following, we will assume that for all *j, the inequality $\left|\hat{c}_j - c_j\right| \le T_j$ holds, and $\hat{c}_j \in [1,n]$.* According to Lemma 6.1, this occurs with probability at least $3/4$.

Now we define the endpoints of intervals which partition $[1,n]$.

Definition 6.2. Let $0 < \varepsilon < 1$, t_1 be the largest integer such that $\frac{n}{(1+\varepsilon)^{t_1-1}} \ge \frac{n}{\sqrt{W}}$, and t_2 be the largest integer such that $\frac{n}{\sqrt{W}}(1 - \varepsilon \cdot t_2) \ge \frac{\varepsilon n}{\sqrt{W}}$. Note that $t_1 = \lfloor \log_{1+\varepsilon} \sqrt{W} + 1 \rfloor$ and $t_2 = \lfloor \frac{1}{\varepsilon} - 1 \rfloor$. Define B_i such that

$$B_i = \begin{cases} \frac{n}{(1+\varepsilon)^{i-1}} & \text{if } 1 \le i \le t_1 \\ \frac{n}{\sqrt{W}}(1 - \varepsilon \cdot (i - t_1)) & \text{if } t_1 < i \le t_1 + t_2 \\ 1 & \text{if } i = t := t_1 + t_2 + 1 \end{cases}$$

According to the above definition, we have the following fact.

Fact 6.3. *It holds that*

1. $t = t_1 + t_2 + 1 = O(\log_{1+\varepsilon} \sqrt{W} + 1/\varepsilon) = O(\log W/\varepsilon)$;
2. $\frac{n}{(1+\varepsilon)^{t_1}} < \frac{n}{\sqrt{W}}$; *and thus,* $B_{t_1} = \frac{n}{(1+\varepsilon)^{t_1-1}} = (1+\varepsilon)\frac{n}{(1+\varepsilon)^{t_1}} < (1+\varepsilon)\frac{n}{\sqrt{W}}$;
3. $B_{t_1} - B_{t_1+1} \ge \frac{n}{\sqrt{W}} - (1-\varepsilon)\frac{n}{\sqrt{W}} = \frac{\varepsilon n}{\sqrt{W}}$, *and* $B_{t_1} - B_{t_1+1} < (1+\varepsilon)\frac{n}{\sqrt{W}} - (1-\varepsilon)\frac{n}{\sqrt{W}} = 2\frac{\varepsilon n}{\sqrt{W}}$;
4. *when $i \le t_1$,* $B_{i-1} - B_i = (1+\varepsilon)B_i - B_i = \varepsilon B_i$; *and when $i > t_1$,* $B_{i-1} - B_i \ge \frac{\varepsilon n}{\sqrt{W}}$.

5. *as for* all $j \in [W]$, $\hat{c}_j \le n$, *when we invoke* BINARYSEARCH$(\widehat{C}, 1, W, B_1)$, *we will never access* $\hat{c}_m > B_1$. *Thus,* ℓ *remains to be* 1, *and when* $\ell = r$, *the returned index* $j_1 = 1$.

We then prove that the endpoints B_i's defined in this way form a valid discretization of $[1,n]$ w.r.t. C and $T = (T_1, \ldots, T_W)$.

Lemma 6.4. *Assume that the event stated in Lemma 6.1 holds. The sequence of endpoints* $(B_1, \ldots, B_t)$ *defined in Definition 6.2 is a valid discretization of* $[1,n]$ *w.r.t. C and error bound* $T_j = \frac{\varepsilon}{8} \cdot \max\left\{\frac{n}{\sqrt{W}}, c_j\right\}$ *on each* $c_j, j \in [W]$.

Proof. We define $B_0 = \infty$ and $B_{t+1} = -\infty$. We consider the sequence $(\hat{c}_1, \ldots, \hat{c}_W)$ as the approximation of $C = (c_1, \ldots, c_W)$. According to the previous assumption, it holds that for all j, $\left|\hat{c}_j - c_j\right| \le T_j$, and $\hat{c}_j \in [1,n]$.

Thus, to show that $(B_1, \ldots, B_t)$ is a valid discretization of $[1,n]$ w.r.t. C and $T = (T_1, \ldots, T_W)$, it suffices to prove that for any $1 \le j \le W, 1 \le i \le t-1$ such that $B_{i+1} \le c_j \le B_i$, it holds that

$$T_j < \min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\}.$$

Now consider any fixed $j \in [W]$ and the corresponding interval $[B_{i+1}, B_i]$ that contains c_j for some $i \in [t-1]$. We analyze the following cases by using properties given in Fact 6.3.

Case (I): $i \le t_1 - 2$. In this case, we have $B_{i-1} - B_i = \varepsilon B_i$, for any $2 \le i \le t_1 - 2$ and $B_{i-1} - B_i = \infty$ if $i = 1$. Furthermore, $B_{i+1} - B_{i+2} = \varepsilon B_{i+2} = \frac{\varepsilon}{(1+\varepsilon)^2} B_i$. Thus,

$$\min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\} = \frac{\varepsilon}{(1+\varepsilon)^2} B_i > \frac{\varepsilon}{8} B_i,$$

where the last inequality follows from the fact that $(1+\varepsilon)^2 \le 4$.

Now note that in this case, it holds that $B_i \ge c_j \ge B_{i+1} > \frac{n}{\sqrt{W}}$. Therefore, $T_j = \frac{\varepsilon}{8} \cdot \max\{\frac{n}{\sqrt{W}}, c_j\} = \frac{\varepsilon}{8} \cdot c_j \le \frac{\varepsilon}{8} \cdot B_i < \min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\}$.

Case (II): $i = t_1 - 1$. Here, we have that $B_{i-1} - B_i = B_{t_1-2} - B_{t_1-1} = \varepsilon B_{t_1-1}$, and $B_{i+1} - B_{i+2} = B_{t_1} - B_{t_1+1} \ge \frac{\varepsilon n}{\sqrt{W}} > \frac{\varepsilon}{1+\varepsilon} B_{t_1} = \frac{\varepsilon}{(1+\varepsilon)^2} B_{t_1-1}$. Furthermore, $B_i = B_{t_1-1} \ge c_j \ge B_{i+1} = B_{t_1} \ge \frac{n}{\sqrt{W}}$, and thus

$$T_j = \frac{\varepsilon}{8} \cdot \max\{\frac{n}{\sqrt{W}}, c_j\} = \frac{\varepsilon}{8} c_j \le \frac{\varepsilon}{8} B_{t_1-1} < \frac{\varepsilon}{(1+\varepsilon)^2} B_{t_1-1} = \min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\}$$

Case (III): $i = t_1$. Here, we have $B_{i-1} - B_i = \varepsilon B_i = \varepsilon B_{t_1}$, and $B_{i+1} - B_{i+2} = \frac{\varepsilon n}{\sqrt{W}} > \frac{\varepsilon}{1+\varepsilon} B_{t_1}$. Furthermore, $c_j \le B_{t_1} < (1+\varepsilon)\frac{n}{\sqrt{W}}$. Thus,

$$T_j = \frac{\varepsilon}{8} \cdot \max\left\{\frac{n}{\sqrt{W}}, c_j\right\} \le \frac{(1+\varepsilon)\varepsilon}{8} \frac{n}{\sqrt{W}} < \frac{\varepsilon}{1+\varepsilon} B_{t_1} = \min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\}$$

Case (IV): $i > t_1$. In this case, we have $B_{i-1} - B_i \ge \frac{\varepsilon n}{\sqrt{W}}$ and $B_{i+1} - B_{i+2} \ge \frac{\varepsilon n}{\sqrt{W}}$ for any $t_1 < i \le t-2$ and $B_{i+1} - B_{i+2} = \infty$ for $i = t-1$. Thus,

$$\min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\} \ge \frac{\varepsilon n}{\sqrt{W}}$$

Furthermore, $c_j \le B_i \le B_{t_1+1} < \frac{n}{\sqrt{W}}$. Thus, $T_j = \frac{\varepsilon}{8} \cdot \max\left\{\frac{n}{\sqrt{W}}, c_j\right\} = \frac{\varepsilon}{8} \cdot \frac{n}{\sqrt{W}} < \min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\}$.

This completes the proof of the lemma. □

Now we are ready to exploit the binary search based algorithm for succinctly approximating the sequence $C = (c_1, \ldots, c_W)$ to estimate the single-linkage clustering cost $\mathrm{cost}(G)$. The algorithm simply performs binary search Algorithm 3 on the estimated array $\widehat{C} = \{\hat{c}_1, \ldots, \hat{c}_W, 0\}$ with search keys $B_1, \ldots, B_t$ as defined in Definition 6.2, where $\hat{c}_j$'s are estimators of c_j's as defined in Lemma 6.1. Then we obtain the corresponding indices $\hat{j}_1, \ldots, \hat{j}_t$. For $i \in \{1, \ldots, t-1\}$, let $\hat{J}_i = \{\hat{j}_i, \ldots, \hat{j}_{i+1} - 1\}$. Note that $\hat{J}_1, \ldots, \hat{J}_{t-1}$ form a partition of the index set $[W] = \{1, \ldots, W\}$. For any $j \in \hat{J}_i$, we define $\bar{c}_j = B_i$ as the estimate for c_j and then use $\bar{c}_j$'s to estimate the cost $\mathrm{cost}(G)$.

Algorithm 4: APPCOST(G, ε, W, d)

input : graph G, approximation parameter ε, maximum weight W, average degree d
output: $\widehat{\mathrm{cost}}(G)$, which estimates $\mathrm{cost}(G)$

1 set t according to Definition 6.2 and set parameters $k = \sqrt{W}$, $d^{(G)} = d \cdot \lceil \frac{4k}{\varepsilon} \rceil$
2 **for** $1 \le i \le t$ **do**
3 set B_i according to Definition 6.2
4 invoke BINARYSEARCH$(\widehat{C}, 1, W, B_i)$ and get output index $\hat{j}_i$, where $\widehat{C} = (\hat{c}_1, \ldots, \hat{c}_W, 1)$ and each $\hat{c}_j = \min\{\max\{\hat{c}'_j, 1\}, n\}$ and $\hat{c}'_j$ is the output of Algorithm 2 with parameters G, $H = G_j$, $\varepsilon/8$, $k = \sqrt{W}$, d, $\delta = 1/(4W)$ ▷ for consistency, reuse stored value of $\hat{c}_j$, if previously estimated
5 for each $i \in \{1, \ldots, t-1\}$, and $j \in \hat{J}_i := \{\hat{j}_i, \ldots, \hat{j}_{i+1} - 1\}$, define $\bar{c}_j = B_i$ ▷ $\bar{c}_j$'s are defined for the analysis only
6 let $\widehat{\mathrm{cost}}(G) = \frac{1}{2} \sum_{i=1}^{t-1} (\hat{j}_{i+1} - \hat{j}_i) \cdot (B_i^2 - B_i)$
7 output $\widehat{\mathrm{cost}}(G)$ and the sequence $\{\hat{j}_1, \ldots, \hat{j}_t\}$

The pseudocode of the algorithm is Described in Algorithm 4. It is important to note that we do not need to access all the values in the estimated array $\widehat{C}$. Instead, we only need to access each $\hat{c}_m$ in the search path corresponding to the binary search process. Besides, to ensure consistency in $\widehat{C}$, we reuse $\hat{c}_m$'s stored value, if it was accessed previously. Intuitively, this means that we only need to approximate the number of connected components for $O(\mathrm{poly}\log W)$ subgraphs G_j. The interval endpoints are set according to Definition 6.2, which ensures that the sequence $(B_1, \ldots, B_t)$ forms a valid discretization of $[1, n]$ w.r.t. C and an appropriately chosen error bound as described in Lemma 6.4. Furthermore, we note that given the values $\hat{j}_i$'s and B_i's , Algorithm 4 can be implemented in $O(t)$ time, as we only need to use the right-hand side (RHS) of the estimator $\widehat{\mathrm{cost}}(G)$ to estimate $\mathrm{cost}(G)$, and the definitions for $\bar{c}_j$'s are primarily used for the analysis.

We first show the following guarantee of the estimate $\bar{c}_j$, for $1 \le j \le W$.

Lemma 6.5. *Assume that for* all $j \in [W]$, *the inequality* $\left|\hat{c}_j - c_j\right| \le T_j$ *holds. Then for* all $j \in [W]$,

$$|\bar{c}_j - c_j| \le 4\varepsilon \cdot \max\left\{\frac{n}{\sqrt{W}}, c_j\right\}.$$

Proof. By Lemma 6.4, the preconditions of Lemma 5.4 are satisfied. By Lemma 5.4, we have $B_{i+2} \le c_j \le B_{i-1}$. By the definition of $\bar{c}_j$, we have that

- When $i = 1$, we have $\left|\bar{c}_j - c_j\right| \le B_i - B_{i+2} = ((1+\varepsilon)^2 - 1)B_{i+2} \le 3\varepsilon B_{i+2} \le 3\varepsilon c_j$.
- When $1 < i \le t_1 - 2$, we have if $c_j > \bar{c}_j$, then $0 < c_j - \bar{c}_j \le B_{i-1} - B_i = \varepsilon B_i \le \varepsilon c_j$; and if $c_j \le \bar{c}_j$, then $0 \le \bar{c}_j - c_j \le B_i - B_{i+2} = ((1+\varepsilon)^2 - 1)B_{i+2} \le 3\varepsilon \cdot c_j$.

- When $i = t_1 - 1$, we have if $c_j > \bar{c}_j$, then $0 < c_j - \bar{c}_j \le B_{i-1} - B_i = \varepsilon B_i \le \varepsilon c_j$; and if $c_j \le \bar{c}_j$, then $0 \le \bar{c}_j - c_j \le B_{t_1-1} - B_{t_1+1} < ((1+\varepsilon)^2 - (1-\varepsilon))\frac{n}{\sqrt{W}} \le 4\varepsilon\frac{n}{\sqrt{W}}$, in which we make use of the fact that $B_{t_1-1} < (1+\varepsilon)^2\frac{n}{\sqrt{W}}$, which in turns follows from the fact that $B_{t_1} < (1+\varepsilon)\frac{n}{\sqrt{W}}$.
- When $i = t_1$, we have that if $c_j > \bar{c}_j$, then $0 \le c_j - \bar{c}_j \le B_{i-1} - B_i = \varepsilon B_i \le \varepsilon c_j$; and if $c_j \le \bar{c}_j$, then $0 \le \bar{c}_j - c_j \le B_{t_1} - B_{t_1+2} < ((1+\varepsilon) - (1-2\varepsilon))\frac{n}{\sqrt{W}} = 3\varepsilon\frac{n}{\sqrt{W}}$.
- When $i = t_1 + 1$, we have that if $c_j > \bar{c}_j$, then $0 < c_j - \bar{c}_j \le B_{t_1} - B_{t_1+1} < ((1+\varepsilon) - (1-\varepsilon))\frac{n}{\sqrt{W}} = 2\varepsilon\frac{n}{\sqrt{W}}$; and if $c_j \le \bar{c}_j$, then $0 \le \bar{c}_j - c_j \le B_{t_1+1} - B_{t_1+3}((1-\varepsilon) - (1-3\varepsilon))\frac{n}{\sqrt{W}} = 2\varepsilon\frac{n}{\sqrt{W}}$.
- When $t_1 + 2 \le i \le t - 2$, $B_{i-1} - B_i = \frac{\varepsilon n}{\sqrt{W}}$, and $B_i - B_{i+2} = B_i - B_{i+1} + B_{i+1} - B_{i+2} = 2\frac{\varepsilon n}{\sqrt{W}}$, and so

$$|\bar{c}_j - c_j| \le \max\{B_{i-1} - B_i, B_i - B_{i+2}\} \le 2\frac{\varepsilon n}{\sqrt{W}}$$

- When $i = t - 1$, $B_{i-1} - B_i = \frac{\varepsilon n}{\sqrt{W}}$ and $B_i - B_{i+1} \le 2\frac{\varepsilon n}{\sqrt{W}}$. Since $B_t \le c_j \le B_{t-2}$, we have $\left|\bar{c}_j - c_j\right| \le \max\{B_{t-2} - B_{t-1}, B_{t-1} - B_t\} \le 2\frac{\varepsilon n}{\sqrt{W}}$.
- When $i = t$, $\left|\bar{c}_j - c_j\right| \le B_{t-1} - B_t = \frac{\varepsilon n}{\sqrt{W}}$.

Therefore, for *all* $1 \le j \le W$, we have $\left|\bar{c}_j - c_j\right| \le 4\varepsilon \cdot \max\left\{\frac{n}{\sqrt{W}}, c_j\right\}$ □

Now we analyze the performance of Algorithm 4 and prove Theorem 1.1.

Theorem 1.1. *Let G be a weighted graph with edge weights in $\{1,\dots,W\}$ with average (unweighted) degree d. Assume that $\sqrt{W} \le n$ and let $0 < \varepsilon < 1$ be a parameter. Algorithm 4 outputs an estimate $\widehat{\text{cost}}(G)$ of the single-linkage clustering cost $\text{cost}(G)$ in the distance graph such that with probability at least $3/4$,*

$$(1-\varepsilon)\text{cost}(G) \le \widehat{\text{cost}}(G) \le (1+\varepsilon)\text{cost}(G).$$

The query complexity and running time of the algorithm are $O(\frac{\sqrt{W}d}{\varepsilon^3}\log^4(\frac{Wd}{\varepsilon}))$ in expectation.

Proof of Theorem 1.1. By Lemma 6.5, with probability at least $\frac{3}{4}$, we have $\left|\bar{c}_j - c_j\right| \le 4\varepsilon \cdot \max\left\{\frac{n}{\sqrt{W}}, c_j\right\}$ for all $j \in [W]$. Then, $\left|\bar{c}_j^2 - c_j^2\right| = \left|\bar{c}_j - c_j\right| \cdot \left|\bar{c}_j + c_j\right| \le \left|\bar{c}_j - c_j\right| \cdot (2c_j + |\bar{c}_j - c_j|) \le 4\varepsilon \cdot \max\left\{\frac{n}{\sqrt{W}}, c_j\right\} \cdot 6\max\left\{\frac{n}{\sqrt{W}}, c_j\right\} \le 24\varepsilon\left(\frac{n^2}{W} + c_j^2\right)$.

As $\widehat{\text{cost}}(G) = \frac{1}{2}\sum_{i=1}^{t-1}(\hat{j}_{i+1} - \hat{j}_i) \cdot (B_i^2 - B_i)$, and by the definition of $\bar{c}_j$, this is equivalent to $\widehat{\text{cost}}(G) = \frac{n(n-1)}{2} + \frac{1}{2}\sum_{j=1}^{W-1}(\bar{c}_j^2 - \bar{c}_j)$. We have that

$$\begin{aligned}
\left|\text{cost}(G) - \widehat{\text{cost}}(G)\right| &= \frac{1}{2}\left|\sum_{j=1}^{W-1}[(c_j^2 - c_j) - (\bar{c}_j^2 - \bar{c}_j)]\right| && \text{(by definitions of cost}(G)\text{ and }\widehat{\text{cost}}(G)\text{)} \\
&\le \frac{1}{2}\sum_{j=1}^{W-1}(\left|c_j^2 - \bar{c}_j^2\right| + \left|\bar{c}_j - c_j\right|) \\
&\le \frac{1}{2}\sum_{j=1}^{W-1}(24\varepsilon\frac{n^2}{W} + 24\varepsilon c_j^2 + 4\varepsilon\frac{n}{\sqrt{W}} + 4\varepsilon c_j) && \text{(applying the error bound for } \bar{c}_j\text{)}
\end{aligned}$$

$$\leq \frac{24}{2}\varepsilon n^2 + \frac{24}{2}\varepsilon \sum_{j=1}^{W-1}(c_j^2 - c_j) + 2\varepsilon n\sqrt{W} + 14\varepsilon \sum_{j=1}^{W-1} c_j$$

Since $\sqrt{W} \leq n$, it holds that $W \leq n^2$. By [CRT05], we also know that the weight of minimum spanning tree is $\text{cost}(\text{MST}) = n - W + \sum_{j=1}^{W-1} c_j$, and $\text{cost}(\text{MST}) = \sum_{i=1}^{n-1} w_i = \text{cost}_1 \leq \text{cost}(G)$. Thus, $\sum_{j=1}^{W-1} c_j = \text{cost}(\text{MST}) - n + W \leq \text{cost}(G) - n + W \leq \text{cost}(G) + n^2$, and we have

$$\left|\text{cost}(G) - \widehat{\text{cost}}(G)\right| \leq \frac{24}{2}\varepsilon n^2 + \frac{24}{2}\varepsilon \sum_{j=1}^{W-1}(c_j^2 - c_j) + 2\varepsilon^2 n^2 + 14\varepsilon(\text{cost}(G) + n^2) \leq 102\varepsilon\,\text{cost}(G)$$

The last inequality follows as $\text{cost}(G) = \frac{n(n-1)}{2} + \frac{1}{2}\sum_{j=1}^{W-1}(c_j^2 - c_j) \geq \frac{n(n-1)}{2}$. When $n \geq 2$, $\text{cost}(G) \geq \frac{1}{4}n^2$. Replacing ε with $\varepsilon/102$ achieves a $(1+\varepsilon)$ approximation factor.

Running time analysis. Note that Algorithm 4 invokes BINARYSEARCH for $t = O(\log W/\varepsilon)$ search keys, and each invocation of BINARYSEARCH takes $O(\log W)$ times. Thus, the algorithm accesses at most $O(\log^2 W/\varepsilon)$ estimates $\hat{c}_j$ in $\widehat{C}$. According to Corollary 3.3, each estimate of $\hat{c}_j$ can be obtained in $O(\frac{\sqrt{W}}{\varepsilon^2} d \log(\frac{\sqrt{W}}{\varepsilon} d) \cdot \log(W))$ time. Thus, the running time of Algorithm 4 is $O(\frac{\log^2 W}{\varepsilon} \cdot \frac{\sqrt{W}}{\varepsilon^2} d \log \frac{\sqrt{W} d}{\varepsilon} \cdot \log W) = O(\frac{\sqrt{W} d}{\varepsilon^3} \log^4 \frac{Wd}{\varepsilon}) = \tilde{O}(\frac{\sqrt{W} d}{\varepsilon^3})$. □

6.2 Estimating the Profile Vector

Recall that $\text{cost}(G) = \sum_{k=1}^{n} \text{cost}_k = \sum_{i=1}^{n} \sum_{i=1}^{n-k} w_i$, where cost_k is the cost of the k-SLC and $w_1, \ldots, w_{n-1}$ are the weights of the minimum spanning tree in non-decreasing order. Now we give an algorithm for approximating the SLC profile vector $(\text{cost}_1, \ldots, \text{cost}_n)$ and prove Theorem 1.2.

Theorem 1.2. *Assume that $\sqrt{W} \leq n$ and let $0 < \varepsilon < 1$ be a parameter. Algorithm 5 generates a succinct representation of an approximation $(\widehat{\text{cost}}_1, \ldots, \widehat{\text{cost}}_n)$ of the SLC profile vector $(\text{cost}_1, \ldots, \text{cost}_n)$ in the distance graph such that with probability at least $3/4$, it holds that*

$$\sum_{k=1}^{n} \left|\widehat{\text{cost}}_k - \text{cost}_k\right| \leq \varepsilon \cdot \text{cost}(G).$$

The query complexity and running time of the algorithm are $O(\frac{\sqrt{W} d}{\varepsilon^3} \log^4(\frac{Wd}{\varepsilon}))$ in expectation.

In the following, we first derive a formula for the quantity cost_k for each $k = c_j$, where $1 \leq j \leq W$. Using this formula, we define a corresponding quantity $\overline{\text{cost}}_{B_i}$ for each B_i, where $1 \leq i \leq t$, as introduced in Definition 6.2. These quantities constitute our succinct representation. Finally, we define a profile oracle that, given any specified k, outputs the estimator $\widehat{\text{cost}}_k$ of cost_k.

Formulas of Profile. By definition, we can equivalently express cost_k as follows:

$$\begin{aligned}
\text{cost}_k &= \sum_{i=1}^{n-k} w_i \\
&= (n - c_1) + (c_1 - c_2) \cdot 2 + \cdots + (c_{w_{n-k}-2} - c_{w_{n-k}-1}) \cdot (w_{n-k} - 1) + (c_{w_{n-k}-1} - k) \cdot w_{n-k} \\
&= n + c_1 + c_2 + \cdots + c_{w_{n-k}-1} - k \cdot w_{n-k}
\end{aligned}$$

$$= n + \sum_{j=1}^{w_{n-k}-1} c_j - k \cdot w_{n-k} \tag{3}$$

Now we observe that for any given integer $j \in \{1,\ldots,W\}$, we can determine the number of edges in the MST that have weights at most j. This allows us to compute cost_k, where k corresponds to the rank of the first edge in the MST with weight j. We have the following lemma.

Lemma 6.6. *Given any integer $j \in \{1,...,W\}$ and define $k = c_j$, where c_j is the number of connected components in G_j. Here, G_j is a subgraph of G, and contains all edges with weights at most j. Then we have,*

$$\text{cost}_k = n + \sum_{i=1}^{j-1} c_i - c_j \cdot j.$$

Proof. Let n_j denote the number of edges in the MST with weight j. From our earlier analysis in Theorem 4.1, we have that $n_j = c_{j-1} - c_j$ for any $1 \le j \le W$, where $c_0 = n$. Then the number of edges in the MST with weights at most j is

$$n_1 + n_2 + \cdots + n_j = (n - c_1) + (c_1 - c_2) + \cdots + (c_{j-1} - c_j) = n - c_j$$

As $k = c_j$, the $(n-k)$-th smallest edge in the MST has weight $w_{n-k} = w_{n-c_j} = j$. Then the statement of lemma follows from Equation (3). □

Algorithm to Estimate Profile. Our main idea of approximating the profile of clustering is to first define $\overline{\text{cost}}_{B_i}$ for non-integer B_i, $1 \le i \le t$, where B_i's are interval endpoints defined in Definition 6.2. We will make use of Algorithm 4, which performs a binary search using the search key B_i and parameters as described in Algorithm 4, ultimately returning the index $\hat{j}_i$. By substituting j with $\hat{j}_i$ and c_j with $\bar{c}_j = B_i$ in the formula given in Lemma 6.6, we obtain:

$$\overline{\text{cost}}_{B_i} = n + \sum_{j=1}^{\hat{j}_i - 1} \bar{c}_j - B_i \cdot \hat{j}_i = n + \sum_{k=1}^{i-1} (\hat{j}_{k+1} - \hat{j}_k) \cdot B_k - B_i \cdot \hat{j}_i. \tag{4}$$

This approach gives a t-dimensional vector $(\overline{\text{cost}}_{B_1}, \ldots, \overline{\text{cost}}_{B_t})$ and effectively summarizes the histogram of profile vector. The detailed algorithm is described in Algorithm 5.

Algorithm 5: APPPROFILE(G, ε, W, d)

```
get sequence {ĵ_1, ..., ĵ_t} from Algorithm 4 APPCOST(G, ε, W, d)
set t according to Definition 6.2
set cost-bar_{B_1} = 0
for i ∈ {2, ..., t} do
    set B_i according to Definition 6.2
    set cost-bar_{B_i} = n − B_i · ĵ_i + Σ_{k=1}^{i−1} (ĵ_{k+1} − ĵ_k) · B_k
output the vector (cost-bar_{B_1}, ..., cost-bar_{B_t})
```

Given $\{B_1,\ldots,B_t\}$, the vector $(\overline{\text{cost}}_{B_1},\ldots,\overline{\text{cost}}_{B_t})$, and any specified integer $k \in [1,n]$ such that $B_{i+1} \le k < B_i$, we define $\widehat{\text{cost}}_k = \overline{\text{cost}}_{B_{i+1}}$ to be the estimate for cost_k, as described in Algorithm 6.

We call the vector $(\overline{\text{cost}}_{B_1},\ldots,\overline{\text{cost}}_{B_t})$ a *succinct representation* of the vector $(\widehat{\text{cost}}_1,\ldots,\widehat{\text{cost}}_n)$. We call Algorithm 6 a *profile oracle*, as it can take any index k as input and answer $\widehat{\text{cost}}_k$.

Algorithm 6: PROFILEORACLE($k, \{B_1, \ldots, B_t\}, (\overline{\text{cost}}_{B_1}, \ldots, \overline{\text{cost}}_{B_t})$)

1 define $B_0 = \infty$
2 use binary search over $(B_0, B_1, \ldots, B_t)$, and find the index i such that $B_{i+1} \le k < B_i$
3 output $\widehat{\text{cost}}_k := \overline{\text{cost}}_{B_{i+1}}$

Analysis of Algorithms 5 and 6. Now we analyze Algorithms 5 and 6. In particular, we will show that the vector $(\widehat{\text{cost}}_1, \ldots, \widehat{\text{cost}}_n)$ is a good approximation of the profile vector $(\text{cost}_1(G), \ldots, \text{cost}_n(G))$. We first give an error bound for each individual $\widehat{\text{cost}}_k$, and then bound the sum of the error and give the proof of Theorem 1.2.

Fact 6.7. *For any* $1 \le i \le t-3$, $B_i - B_{i+3} \le 8\varepsilon \cdot \max\{B_{i+1}, \frac{n}{\sqrt{W}}\}$.

Proof. According to Definition 6.2, when $i+3 \le t_1$, i.e., $i \le t_1 - 3$, we have $B_i = \frac{n}{(1+\varepsilon)^{i-1}}$ and $B_{i+3} = \frac{n}{(1+\varepsilon)^{i+2}}$. Thus, since $\frac{1}{1+\varepsilon} \ge 1-2\varepsilon$,

$$B_i - B_{i+3} = ((1+\varepsilon) - \frac{1}{(1+\varepsilon)^2})\frac{n}{(1+\varepsilon)^i} \le (1+\varepsilon-(1-2\varepsilon)^2)B_{i+1} \le 5\varepsilon B_{i+1}$$

When $i = t_1 - 2$, $B_i - B_{i+3} = (1+\varepsilon)^2 B_{t_1} - B_{t_1+1} = (2\varepsilon+\varepsilon^2)B_{t_1} + (B_{t_1} - B_{t_1+1})$. Since $B_{t_1} - B_{t_1+1} < 2\frac{\varepsilon n}{\sqrt{W}}$ and $B_{t_1} < (1+\varepsilon)\frac{n}{\sqrt{W}} < 2\frac{n}{\sqrt{W}}$, $B_i - B_{i+3} < 3\varepsilon B_{t_1} + 2\frac{\varepsilon n}{\sqrt{W}} < 6\frac{\varepsilon n}{\sqrt{W}} + 2\frac{\varepsilon n}{\sqrt{W}} = 8\varepsilon\frac{n}{\sqrt{W}}$.

When $i = t_1 - 1$, $B_i - B_{i+3} = (1+\varepsilon)B_{t_1} - B_{t_1+2} = \varepsilon B_{t_1} + (B_{t_1} - B_{t_1+1}) + \frac{\varepsilon n}{\sqrt{W}} < 2\frac{\varepsilon n}{\sqrt{W}} + 2\frac{\varepsilon n}{\sqrt{W}} + \frac{\varepsilon n}{\sqrt{W}} = 5\varepsilon\frac{n}{\sqrt{W}}$.

When $i = t_1$, $B_i - B_{i+3} = B_{t_1} - B_{t_1+3} = (B_{t_1} - B_{t_1+1}) + 2\frac{\varepsilon n}{\sqrt{W}} < 2\frac{\varepsilon n}{\sqrt{W}} + 2\frac{\varepsilon n}{\sqrt{W}} = 4\varepsilon\frac{n}{\sqrt{W}}$.

When $i \ge t_1 + 1$, $B_i - B_{i+3} = 3\varepsilon\frac{n}{\sqrt{W}}$, which finishes the proof of the fact. □

The following lemma provides an error bound for individual $\widehat{\text{cost}}_k$, demonstrating that the error grows as k increases. Observe that $|\widehat{\text{cost}}_k - \text{cost}_k| = |\overline{\text{cost}}_{B_{i+1}} - \text{cost}_k| \le |\overline{\text{cost}}_{B_{i+1}} - \text{cost}_{c_{j_{i+1}}}| + |\text{cost}_{c_{j_{i+1}}} - \text{cost}_k|$, where j_{i+1} is the smallest index such that $\hat{c}_{j_{i+1}} \le B_{i+1}$. We bound these two terms separately. First, for each interval, B_{i+1} is close to $c_{j_{i+1}}$, so the gap $|\overline{\text{cost}}_{B_{i+1}} - \text{cost}_{c_{j_{i+1}}}|$ is small. Second, by definition, $\text{cost}_k = \sum_{j=1}^{n-k} w_j$, where w_j is the j-th smallest edge weight in the MST. Since both k and $c_{j_{i+1}}$ are constrained by nearby interval endpoints, the gap $|\text{cost}_{c_{j_{i+1}}} - \text{cost}_k|$ is also small. Consequently, the error $|\widehat{\text{cost}}_k - \text{cost}_k|$ is bounded.

Lemma 6.8. *Assume that* $\sqrt{W} \le n$ *and let* $0 < \varepsilon < 1$ *be a parameter. With probability at least 3/4, for any integer* $k \in \{1, \ldots, n\}$*, Algorithm 6 returns an estimate* $\widehat{\text{cost}}_k$ *for the k-clustering cost, i.e.,* cost_k*, such that*

$$|\widehat{\text{cost}}_k - \text{cost}_k| \le 4\varepsilon \cdot \text{cost}_k + 48\varepsilon \cdot \max\{k, \frac{n}{\sqrt{W}}\} \cdot \max\{w_{n-k}, j_{i+1}\} \le 4\varepsilon \cdot \text{cost}_k + 48\varepsilon \cdot \max\{k, \frac{n}{\sqrt{W}}\}W$$

Given the succinct representation $(\overline{\text{cost}}_{B_1}, \ldots, \overline{\text{cost}}_{B_t})$*, the running time of the algorithm is* $O(\log(\frac{\log W}{\varepsilon}))$.

Proof. We first note that in the case that $\varepsilon < \frac{\sqrt{W}}{n}$, we can simply find the exact minimum spanning tree in $\tilde{O}(nd) \ll \tilde{O}(\frac{n^3}{W}d) = \tilde{O}(\frac{\sqrt{W}}{\varepsilon^3}d)$, and calculate the exact SLC profile vector in linear time. Then the theorem trivially holds. Thus in the following, we assume that $\varepsilon \ge \frac{\sqrt{W}}{n}$.

Recall from Lemma 6.5 that with probability at least $\frac{3}{4}$, we have $|\bar{c}_j - c_j| \le 4\varepsilon \cdot \max\left\{\frac{n}{\sqrt{W}}, c_j\right\}$ for all $j \in [W]$. In the following, we will assume that this event holds.

For simplicity of notation, we use j_i to denote $\hat{j}_i$, the value returned by the binary search with key B_i as invoked in Algorithm 4. By definitions of $\overline{\text{cost}}_{B_i}$ and $\text{cost}_{c_{j_i}}(G)$, we have that

$$\begin{aligned}
|\overline{\text{cost}}_{B_i} - \text{cost}_{c_{j_i}}| &= \left|(n + \sum_{j=1}^{j_i-1} \bar{c}_j - B_i \cdot j_i) - (n + \sum_{j=1}^{j_i-1} c_i - c_{j_i} \cdot j_i)\right| \\
&= |\sum_{j=1}^{j_i-1} (\bar{c}_j - c_j) - B_i \cdot j_i + c_{j_i} \cdot j_i| \\
&\leq \sum_{j=1}^{j_i-1} |\bar{c}_j - c_j| + |B_i - c_{j_i}| \cdot j_i \\
&\leq 4\varepsilon \sum_{j=1}^{j_i-1} c_j + 4\varepsilon \cdot c_{j_i} \cdot j_i && \text{(applying the error bound for } \bar{c}_j) \\
&= 4\varepsilon(\text{cost}_{c_{j_i}} - n + c_{j_i} \cdot j_i) + 4\varepsilon \cdot c_{j_i} \cdot j_i && \text{(according to Lemma 6.6)} \\
&\leq 4\varepsilon \cdot \text{cost}_{c_{j_i}} + 8\varepsilon \cdot c_{j_i} \cdot j_i
\end{aligned}$$

When $k = B_1 = n$, as $\widehat{\text{cost}}_{B_1} = 0$ and $\text{cost}_n = 0$, we have $|\widehat{\text{cost}}_k - \text{cost}_k| = 0$.

For $B_{i+1} \leq k < B_i$ where $1 \leq i \leq t-1$, we estimate cost_k using $\text{cost}_{B_{i+1}}$, and thus we have

$$\begin{aligned}
|\widehat{\text{cost}}_k - \text{cost}_k| &\leq |\overline{\text{cost}}_{B_{i+1}} - \text{cost}_{c_{j_{i+1}}}| + |\text{cost}_{c_{j_{i+1}}} - \text{cost}_k| \\
&\leq 4\varepsilon(\text{cost}_{c_{j_{i+1}}} + 2c_{j_{i+1}} \cdot j_{i+1}) + |\text{cost}_{c_{j_{i+1}}} - \text{cost}_k| \\
&\leq 4\varepsilon(\text{cost}_k + |\text{cost}_{c_{j_{i+1}}} - \text{cost}_k| + 2B_i \cdot j_{i+1}) + |\text{cost}_{c_{j_{i+1}}} - \text{cost}_k| && \text{(since } c_{j_{i+1}} \leq B_i) \\
&\leq 4\varepsilon(\text{cost}_k + 2B_i \cdot j_{i+1}) + 5|\text{cost}_{c_{j_{i+1}}} - \text{cost}_k| && \text{(since } \varepsilon < 1) \\
&\leq 4\varepsilon \cdot \text{cost}_k + 8\varepsilon \cdot \max\{k, \frac{n}{\sqrt{W}}\} \cdot j_{i+1} + |\text{cost}_{c_{j_{i+1}}} - \text{cost}_k| \\
& && \text{(since } B_i \leq B_{i+1} + \varepsilon \cdot \max\{B_{i+1}, \tfrac{n}{\sqrt{W}}\} \leq 2\max\{k, \tfrac{n}{\sqrt{W}}\})
\end{aligned}$$

Thus, we only need to bound $|\text{cost}_{c_{j_{i+1}}} - \text{cost}_k|$.

Case (I): $c_{j_{i+1}} \leq k$. In this case, $\text{cost}_k \leq \text{cost}_{c_{j_{i+1}}}$ and $w_{n-k} \leq j_{i+1}$. Since $\text{cost}_k = \sum_{j=1}^{n-k} w_j$,

$$|\text{cost}_{c_{j_{i+1}}} - \text{cost}_k| = \text{cost}_{c_{j_{i+1}}} - \text{cost}_k = \sum_{j=n-k+1}^{n-c_{j_{i+1}}} w_j \leq (k - c_{j_{i+1}}) \cdot w_{n-c_{j_{i+1}}} = (k - c_{j_{i+1}}) \cdot j_{i+1}$$

When $t-1 \leq i \leq t-2$, $c_{j_{i+1}} \geq B_t = 1$ and $k \leq B_i \leq B_{t-2} = 2\frac{\varepsilon n}{\sqrt{W}}$, and thus $|\text{cost}_{c_{j_{i+1}}} - \text{cost}_k| \leq 2\varepsilon\frac{n}{\sqrt{W}} \cdot j_{i+1}$.

When $i+3 \leq t$, i.e., $i \leq t-3$, from Lemma 5.4, $B_{i+3} \leq c_{j_{i+1}} \leq B_i$, and thus $k - c_{j_{i+1}} \leq B_i - B_{i+3}$. According to Fact 6.7, $B_i - B_{i+3} \leq 8\varepsilon \cdot \max\{B_{i+1}, \frac{n}{\sqrt{W}}\} \leq 8\varepsilon \cdot \max\{k, \frac{n}{\sqrt{W}}\}$. Thus, $|\text{cost}_{c_{j_{i+1}}} - \text{cost}_k| \leq 8\varepsilon \cdot \max\{k, \frac{n}{\sqrt{W}}\} \cdot j_{i+1}$. Therefore

$$|\widehat{\text{cost}}_k - \text{cost}_k| \leq 4\varepsilon \cdot \text{cost}_k + (8\varepsilon + 5 \cdot 8\varepsilon)\max\{k, \frac{n}{\sqrt{W}}\} \cdot j_{i+1} = 4\varepsilon \cdot \text{cost}_k + 48\varepsilon \cdot \max\{k, \frac{n}{\sqrt{W}}\} \cdot j_{i+1}$$

Case (II): $k < c_{j_{i+1}}$. In this case, $\text{cost}_{c_{j_{i+1}}} \leq \text{cost}_k$ and $j_{i+1} \leq w_{n-k}$. We have,

$$|\text{cost}_{c_{j_{i+1}}} - \text{cost}_k| = \text{cost}_k - \text{cost}_{c_{j_{i+1}}} = \sum_{j=n-c_{j_{i+1}}+1}^{n-k} w_j \leq (c_{j_{i+1}} - k) \cdot w_{n-k} \leq (B_i - B_{i+1}) \cdot w_{n-k}$$

When $i < t_1$, we have $B_i - B_{i+1} = \varepsilon B_{i+1} \le \varepsilon k$; when $i \ge t_1$, we have $B_i - B_{i+1} = \varepsilon \frac{n}{\sqrt{W}}$. Thus, $B_i - B_{i+1} \le \varepsilon \cdot \max\{k, \frac{n}{\sqrt{W}}\}$. Then,

$$|\widehat{\text{cost}}_k - \text{cost}_k| \le 4\varepsilon \cdot \text{cost}_k + 8\varepsilon \cdot \max\{k, \frac{n}{\sqrt{W}}\} \cdot j_{i+1} + 8\varepsilon \cdot \max\{k, \frac{n}{\sqrt{W}}\} \cdot w_{n-k}$$

In a conclusion, as $\max\{w_{n-k}, j_{i+1}\} \le W$, we have

$$|\widehat{\text{cost}}_k - \text{cost}_k| \le 4\varepsilon \cdot \text{cost}_k + 48\varepsilon \cdot \max\{k, \frac{n}{\sqrt{W}}\} \cdot \max\{w_{n-k}, j_{i+1}\} \le 4\varepsilon \cdot \text{cost}_k + 48\varepsilon \cdot \max\{k, \frac{n}{\sqrt{W}}\} W$$

Running time analysis. For any k, given the succinct representation, one can simply perform binary search to find the first i with $B_{i+1} \le k < B_i$, and thus the running time of Algorithm 6 is $O(\log t) = O(\log(\frac{\log W}{\varepsilon}))$. □

Note that when k is very close to n, it becomes impossible to estimate cost_k within a constant factor in sublinear time. For instance, when $k = n-1$, estimating cost_k amounts to finding the minimum edge weight in the graph – an inherently hard task to perform in sublinear time. Consequently, no algorithm can provide an estimator $\widehat{\text{cost}}_k$ with a constant-factor approximation guarantee for arbitrary values of k under sublinear time constraints. However, in Theorem 1.2 we prove that the *average* error bound for $\widehat{\text{cost}}_k$ is $\varepsilon \cdot \text{cost}_k$, that is, $\sum_{k=1}^n |\widehat{\text{cost}}_k - \text{cost}_k| \le \varepsilon \sum_{k=1}^n \text{cost}_k$.

Proof of Theorem 1.2. For $i < t_1$, $B_i - B_{i+1} = \varepsilon B_{i+1} \ge \varepsilon B_{t_1} \ge \frac{\varepsilon n}{\sqrt{W}}$, and for $i \ge t_1$, $B_i - B_{i+1} \ge \frac{\varepsilon n}{\sqrt{W}}$. Since $\varepsilon \ge \frac{\sqrt{W}}{n}$, we have that for every i, $B_i - B_{i+1} \ge 1$, then by the error bound for individual $\widehat{\text{cost}}_k$ in Lemma 6.8, we have that

$$\begin{aligned}
\sum_{k=1}^n |\widehat{\text{cost}}_k - \text{cost}_k| &\le \sum_{i=1}^{t-1} \sum_{k \in [B_{i+1}, B_i)} \left(4\varepsilon \cdot \text{cost}_k + 48\varepsilon \cdot \max\{k, \frac{n}{\sqrt{W}}\} \cdot \max\{w_{n-k}, j_{i+1}\} \right) \\
&= 4\varepsilon \sum_{k=1}^n \text{cost}_k + 48\varepsilon \sum_{i=1}^{t} \sum_{k \in [B_{i+1}, B_i)} \max\{k, \frac{n}{\sqrt{W}}\} \cdot \max\{w_{n-k}, j_{i+1}\} \\
&\le 4\varepsilon \cdot \text{cost}(G) + 48\varepsilon \sum_{i=1}^{t_1} \sum_{k \in [B_{i+1}, B_i)} k \cdot \max\{w_{n-k}, j_{i+1}\} + 48\varepsilon \sum_{i=t_1+1}^{t} \sum_{k \in [B_{i+1}, B_i)} \frac{n}{\sqrt{W}} \cdot W \\
&\qquad \text{(by the definition of cost}(G), \text{ and } \max\{w_{n-k}, j_{i+1}\} \le W) \\
&\le 4\varepsilon \cdot \text{cost}(G) + 48\varepsilon \frac{1}{\varepsilon} \cdot \frac{\varepsilon n}{\sqrt{W}} \cdot \frac{n}{\sqrt{W}} W + 48\varepsilon \sum_{i=1}^{t_1} \sum_{k \in [B_{i+1}, B_i)} k \cdot \max\{w_{n-k}, j_{i+1}\} \\
&\qquad (\text{since } t - t_1 = t_2 \le \tfrac{1}{\varepsilon}, \text{ and when } i \ge t_1, B_i - B_{i+1} = \tfrac{n}{\sqrt{W}}) \\
&= 4\varepsilon \cdot \text{cost}(G) + 48\varepsilon n^2 + 48\varepsilon \sum_{i=1}^{t_1} \sum_{k \in [B_{i+1}, B_i)} k \cdot \max\{w_{n-k}, j_{i+1}\} \\
&\le 196\varepsilon \cdot \text{cost}(G) + 48\varepsilon \sum_{i=1}^{t_1} \sum_{k \in [B_{i+1}, B_i)} k \cdot \max\{w_{n-k}, j_{i+1}\} \qquad (\text{since } n^2 \le 4\text{cost}(G))
\end{aligned}$$

Define $(*) = 48\varepsilon \sum_{i=1}^{t_1} \sum_{k \in [B_{i+1}, B_i)} k \cdot \max\{w_{n-k}, j_{i+1}\}$, now we only need to bound $(*)$.

Case (I): $j_{i+1} \le w_{n-k}$. In this case, $(*) \le 48\varepsilon \sum_{k=1}^{n-1} k \cdot w_{n-k} = 48\varepsilon \cdot \text{cost}(G)$, by the fact that $\text{cost}(G) = \sum_{i=1}^{n-1} (n-i) w_i = \sum_{k=1}^{n-1} k \cdot w_{n-k}$.

Case (II): $w_{n-k} \le j_{i+1}$. In this case, since $k < B_i$, we have that

$$(*) \le 48\varepsilon \sum_{i=1}^{t_1} (B_i - B_{i+1}) B_i \cdot j_{i+1} = 48\varepsilon^2 \sum_{i=1}^{t_1} B_{i+1} B_i \cdot j_{i+1}$$

Next we bound $\sum_{i=1}^{t_1} B_{i+1} B_i \cdot j_{i+1}$. On one hand, $\sum_{i=1}^{t_1} B_{i+1} B_i \cdot j_{i+1} = (1+\varepsilon) \sum_{i=1}^{t_1-1} B_{i+1}^2 j_{i+1} = (1+\varepsilon) \sum_{i'=2}^{t_1} B_{i'}^2 j_{i'}$; on the other hand, $\sum_{i=1}^{t_1} B_{i+1} B_i \cdot j_{i+1} = \frac{1}{1+\varepsilon} \sum_{i=1}^{t_1-1} B_i^2 j_{i+1}$.

Therefore, $(1+\varepsilon - \frac{1}{1+\varepsilon}) \sum_{i=1}^{t_1} B_{i+1} B_i \cdot j_{i+1} = \sum_{i=1}^{t_1-1} B_i^2 j_{i+1} - \sum_{i=2}^{t_1} B_i^2 j_i = \sum_{i=1}^{t_1} B_i^2 (j_{i+1} - j_i) - B_{t_1}^2 j_{t_1+1} + B_1^2 j_1$. Note that according to Fact 6.3, $j_1 = 1$.

Moreover, when $i \le t_1$, $B_i \ge B_{t_1} \ge \frac{n}{\sqrt{W}}$. If $\frac{n}{\sqrt{W}} \ge 2$, then $B_i^2 - B_i \ge \frac{B_i}{2}$; and if $n \ge 2$, $\frac{n(n-1)}{2} \ge \frac{n^2}{4}$. Then $\widehat{\text{cost}}(G) = \frac{n(n-1)}{2} + \frac{1}{2} \sum_{i=1}^{t-1} (j_{i+1} - j_i) \cdot (B_i^2 - B_i) \ge \frac{n^2}{4} + \frac{1}{4} \sum_{i=1}^{t_1} B_i^2 (j_{i+1} - j_i)$. Since $\widehat{\text{cost}}(G)$ approximates $\text{cost}(G)$ by $(1+\varepsilon)$ factor, we have

$$\frac{(1+\varepsilon)^2 - 1}{1+\varepsilon} \sum_{i=1}^{t_1} B_{i+1} B_i \cdot j_{i+1} = \frac{3\varepsilon}{1+\varepsilon} \sum_{i=1}^{t_1} B_{i+1} B_i \cdot j_{i+1} \le \sum_{i=1}^{t_1} B_i^2 (j_{i+1} - j_i) + n^2 \le 4 \cdot \widehat{\text{cost}}(G) \le 4(1+\varepsilon)\text{cost}(G)$$

That is, $\varepsilon \sum_{i=1}^{t_1} B_{i+1} B_i \cdot j_{i+1} \le \frac{4}{3}(1+\varepsilon)^2 \text{cost}(G) \le 6\text{cost}(G)$. Then,

$$(*) = 48\varepsilon^2 \sum_{i=1}^{t_1} B_{i+1} B_i \cdot j_{i+1} \le 48\varepsilon \cdot 6\text{cost}(G) = 288\varepsilon \cdot \text{cost}(G)$$

Combining the above analysis, we have $\sum_{k=1}^{n} \left| \widehat{\text{cost}}_k - \text{cost}_k \right| \le (196+288)\varepsilon \cdot \text{cost}(G) = 484\varepsilon \cdot \text{cost}(G)$.

Replacing ε with $\varepsilon/484$, we get an succinct representation of $(\widehat{\text{cost}}_1, \ldots, \widehat{\text{cost}}_n)$ such that each $\widehat{\text{cost}}_k$ is a $(1+\varepsilon)$-estimator *on average*.

Running time analysis. Note that we first invoke Algorithm 4 to obtain the sequence $\{j_0, j_1, \ldots, j_t\}$, which takes $O(\frac{\sqrt{W}d}{\varepsilon^3} \log^4 \frac{Wd}{\varepsilon})$ time. Given the sequence, to estimate each $\overline{\text{cost}}_{B_i}$, we need to calculate $\sum_{k=1}^{i-1} (j_{k+1} - j_k) \cdot B_k$ in $O(t)$ time, for all $i \in [1,t]$. Thus, getting the estimated vector $(\overline{\text{cost}}_{B_1}, \ldots, \overline{\text{cost}}_{B_t})$ can be done in $O(t^2) = O(\log^2(\sqrt{W}/\varepsilon)/\varepsilon^2)$ time. Thus, Algorithm 5 runs in time $O(\frac{\sqrt{W}d}{\varepsilon^3} \log^4 \frac{Wd}{\varepsilon})$. □

7 Sublinear Algorithms in Similarity Case

Now, we consider single-linkage clustering in similarity graphs. In this case, the agglomerative clustering algorithm merges the two clusters with the highest similarity at each step. The similarity between two clusters is defined as the maximum similarity between any pair of their members. Recall that $w_1^{(s)}, w_2^{(s)}, \ldots, w_{n-1}^{(s)}$ are the weights of a maximum spanning tree (MaxST) of G in non-increasing order, and the cost of the SLC in the similarity graph is defined as

$$\text{cost}^{(s)}(G) = \sum_{k=1}^{n} \text{cost}_k^{(s)} = \sum_{i=1}^{n-1} (n-i) \cdot w_i^{(s)},$$

where $\text{cost}_k^{(s)} = \sum_{i=1}^{n-k} w_i^{(s)}$ is the cost of the corresponding k-clustering.

7.1 Cost Formula for Similarity-based Clustering

We first derive an equivalent formula for the clustering cost. We let n_j denote the number of edges of weight j in the MaxST. For each weight j, $G_j^{(s)}$ contains all edges of weight $\geq j$. We let $c_j^{(s)}$ be the number of connected components in $G_j^{(s)}$. We observe that $c_1^{(s)} = 1$ and $c_{W+1}^{(s)} = n$, if we assume the whole graph G is connected. Furthermore, it holds that $\sum_{i<j} n_i = c_j^{(s)} - 1$, which gives $n_j = c_{j+1}^{(s)} - c_j^{(s)}$.

Theorem 7.1. *Let G be a connected graph on n vertices, with edge weights from $\{1,\ldots,W\}$. Then*

$$\mathrm{cost}^{(s)}(G) = \sum_{j=1}^{W} \frac{(c_j^{(s)} + n - 1)(n - c_j^{(s)})}{2}$$

Proof. The cost of clustering in the similarity graph can be derived as:

$$\begin{aligned}
\mathrm{cost}^{(s)}(G) &= \sum_{i=1}^{n-1} (n-i) \cdot w_i^{(s)} && \text{(by Eq. (2))} \\
&= \sum_{i=1}^{n_W} (n-i)\cdot W + \sum_{i=n_W+1}^{n_W+n_{W-1}} (n-i)\cdot(W-1) + \cdots + \sum_{i=n_W+\cdots+n_2+1}^{n_W+\cdots+n_2+n_1} (n-i)\cdot 1 \\
&\qquad \text{(reorganizing the sum by grouping the terms according to edge weights)} \\
&= \sum_{i=1}^{n-c_W^{(s)}} (n-i)\cdot W + \cdots + \sum_{i=n-c_3^{(s)}+1}^{n-c_2^{(s)}} (n-i)\cdot 2 + \sum_{i=n-c_2^{(s)}+1}^{n-c_1^{(s)}} (n-i)\cdot 1 \qquad (\text{since } n_j = c_{j+1}^{(s)} - c_j^{(s)}) \\
&= \sum_{i=c_W^{(s)}}^{n-1} i\cdot W + \cdots + \sum_{i=c_2^{(s)}}^{c_3^{(s)}-1} i\cdot 2 + \sum_{i=c_1^{(s)}}^{c_2^{(s)}-1} i\cdot 1 && \text{(substituting } (n-i) \text{ with } i) \\
&= \sum_{i=c_W^{(s)}}^{n-1} i\cdot(W-1) + \cdots + \sum_{i=c_2^{(s)}}^{c_3^{(s)}-1} i\cdot(2-1) + \sum_{i=c_1^{(s)}}^{n-1} i\cdot 1 \\
&\qquad \text{(factoring out } i \text{ from each summation and combining the terms)} \\
&= \sum_{i=c_W^{(s)}}^{n-1} i + \cdots + \sum_{i=c_2^{(s)}}^{n-1} i + \sum_{i=c_1^{(s)}}^{n-1} i \\
&\qquad \text{(repeating this decomposition until all summations end with } i = n-1) \\
&= \sum_{j=1}^{W} \frac{(c_j^{(s)} + n - 1)(n - c_j^{(s)})}{2}
\end{aligned}$$

□

We remark that the total weight of the MaxST can be derived in an analogous way: $\mathrm{cost}(\mathrm{MaxST}) = \sum_{j=1}^{W} j\cdot(c_{j+1}^{(s)} - c_j^{(s)}) = 1\cdot(c_2^{(s)} - c_1^{(s)}) + 2\cdot(c_3^{(s)} - c_2^{(s)}) + 3\cdot(c_4^{(s)} - c_3^{(s)}) + \cdots + W\cdot(c_{W+1}^{(s)} - c_W^{(s)}) = nW - \sum_{j=1}^{W} c_j^{(s)} = \sum_{j=1}^{W}(n - c_j^{(s)})$.

By the above theorem and the fact that $c_j^{(s)} \geq c_1^{(s)} = 1$ for any j, we have the following facts.

Fact 7.2. *It holds that* $\mathrm{cost}^{(s)}(G) \geq \frac{(c_1^{(s)}+n-1)(n-c_1^{(s)})}{2} = \frac{(1+n-1)(n-1)}{2} = \frac{n(n-1)}{2}$.

Fact 7.3. *It holds that* $\mathrm{cost}^{(s)}(G) \geq \frac{n}{2} \cdot \mathrm{cost}(\mathrm{MaxST})$.

7.2 Estimating the Clustering Cost

The formula of $\mathrm{cost}^{(s)}(G)$ in Theorem 7.1 includes a sum of products $(c_j^{(s)}+n-1)\cdot(n-c_j^{(s)})$. Our approach is to estimate each product in $\tilde{O}(Wd\mathrm{poly}(1/\varepsilon))$ time with small additive error. To do so, we note that by using Algorithm 1, the first term $c_j^{(s)}+n-1$ can be estimated with an additive error of $O(\varepsilon n)$, which is sufficient for our purpose. For the second term $D_j := n - c_j^{(s)}$, it is also tempting to directly applying Algorithm 1 to approximate it. However, the resulting additive error will be too large when $c_j^{(s)}$ is large, i.e., close to n.

7.2.1 Approximating D_j

High Level Idea of Approximating D_j. To resolve the above issue, we relate D_j to the number of *non-isolated* vertices and the connected components in the graph induced by such vertices in the threshold graph $G_j^{(s)}$.

Specifically, for a given graph G and its subgraph H with the same vertex set, we define H_{nis} to be the subgraph of H induced by all non-isolated vertices w.r.t. H. The number of vertices in H_{nis} is defined as n', and the number of connected components is c'. It is important to note that the number of connected components $\mathrm{cc}(H)$ of H is exactly $n-n'+c'$. Thus, the quantity $n-\mathrm{cc}(H) = n'-c'$, and one can approximate $n-\mathrm{cc}(H)$ by approximating n' and c' respectively. We remark that the edges of subgraph H are implicitly defined, so we must inspect *all* edges incident to a vertex v in G to find its neighbors in H. Based on this, we give an algorithm for approximating the $n-\mathrm{cc}(H) = n'-c'$ with small enough additive error in Algorithm 7.

Now we give a bit more details of Algorithm 7, we sample vertices to simultaneously estimate the number $\mathrm{cc}(H)$ of connected components, the number c' of connected components in H_{nis} (each of which is a component with at least 2 vertices in H) and the number n' of non-isolated vertices. If there are only a few isolated vertices, we estimate $n-\mathrm{cc}(H)$ using $n-\hat{c}$, where $\hat{c}$ is an estimate of $\mathrm{cc}(H)$. Otherwise, we estimate $n-\mathrm{cc}(H) = n'_j - c'_j$ using $\hat{n'} - \hat{c'}$, where $\hat{n'}$ and $\hat{c'}$ are estimates of n' and c', respectively.

Then to approximate $D_j = n - c_j^{(s)}$, we let H be the threshold graph $G_j^{(s)}$, and let and n'_j, c'_j be the number of vertices, and the number of connected components of H_{nis}, respectively. Then, one can observe that $\mathrm{cc}(H) = c_j^{(s)} = n - n'_j + c'_j$, so $D_j = n'_j - c'_j$, which can be approximated by Algorithm 7 with appropriate input parameters.

Analysis of Algorithm 7. We first provide an approximation guarantee of Algorithm 7 in the following lemma.

Lemma 7.4. *Let* $0 < \varepsilon < 1$*, and k is an integer greater than* 10*. Let* $r = \lceil \frac{64k}{\varepsilon^2} \rceil$*, and* $\Gamma = \lceil \frac{4k}{\varepsilon} \rceil$*. Suppose the average degree d of graph G is known, and set* $d^{(G)} = d \cdot \Gamma = d \cdot \lceil \frac{4k}{\varepsilon} \rceil$*. Given access to the graph G and an implicit subgraph $H \subseteq G$ (where H shares the same vertex set as G, and the edges of H are determined by evaluating conditions on the edges of G), Algorithm 7 runs in time* $O(\frac{k}{\varepsilon^2} d \log(\frac{k}{\varepsilon} d))$ *and outputs a value* $\widehat{D}$ *such that*

$$\left|\widehat{D} - D\right| \leq \varepsilon \cdot \max\left\{\frac{n}{k}, \min\{D, n-D\}\right\}$$

with error probability at least $\frac{3}{4}$*, where* $D = n - c$ *and c is the number of connected components in H.*

Algorithm 7: APPNCCSIM(G,H,ε,k,d)

input : graph G, subgraph H, approximation parameter ε, threshold k, avg. degree d
output: an estimate $\widehat{D}$ of $n-c$, where c is the number of connected component in H

choose $r = \lceil 64\frac{k}{\varepsilon^2} \rceil$ vertices $v_1, \ldots, v_r$ uniformly at random
for *each sampled vertex* v_i **do**
    identify neighbors of v_i in H by examining all incident edges in G
    if v_i *is isolated in* H **then**
        set $x_i = 0$
    else
        set $x_i = 1$
set $\hat{n'} = \frac{n}{r}\sum_{i=1}^{r} x_i$
choose another $r = \lceil 64\frac{k}{\varepsilon^2} \rceil$ vertices $u_1, \ldots, u_r$ uniformly at random
set threshold $\Gamma = \lceil 4\frac{k}{\varepsilon} \rceil$ and $d^{(G)} = d \cdot \Gamma$
for *each sampled vertex* u_i **do**
    set $\alpha_i = 0$, $\beta_i = 0$
    identify neighbors of u_i in H by examining all incident edges in G
    let $d_{u_i}^{(G)}$ be the degree of u_i in G
    if u_i *is isolated in* H **then**
        set $\beta_i = 1$
    else
        (*) flip a coin
        if *(heads) & (# vertices visited* $\in H < \Gamma$ *during BFS) & (no visited vertex* $\in H$ *has degree in* $G > d^{(G)}$ *during BFS)* **then**
            resume BFS on H, doubling the number of visited edges in G
            if *this allows BFS on* H *to complete* **then**
                set $\alpha_i = \beta_i = d_{u_i}^{(G)} \cdot 2^{\#\text{ coin flips}}/\#\text{edges visited in } G$
            else
                go to (*)
set $\hat{c} = \frac{n}{r}\sum_{i=1}^{r} \beta_i$, $\hat{c'} = \frac{n}{r}\sum_{i=1}^{r} \alpha_i$
if $\hat{n'} < 0.5n$ **then**
    return $\widehat{D} = \hat{n'} - \hat{c'}$
else
    return $\widehat{D} = n - \hat{c}$

Consider the Algorithm 7, we have the following lemma on the estimators $\hat{n'}$ and $\hat{c'}$.

Lemma 7.5. *Let* $r, \Gamma, d^{(G)}$ *be the same parameters set in Lemma 7.4. Let* U *be the set of vertices that lie in components in subgraph* H_{nis} *with fewer than* Γ *vertices; and all of these vertices in the original graph are of degree at most* $d^{(G)}$*. Let* c'_U *denote the number of connected components in* $H_{nis}[U]$*. Then we have,*

$$\mathbf{E}[\hat{n'}] = n', \ \mathrm{Var}[n'] = \frac{n'(n-n')}{r}$$

And $\left|\hat{c'}-c'\right| \le \max\{\frac{n}{k}, c'\}$*, with probability at least* $\frac{7}{8}$.

Proof. **Analysis on** $\hat{n'}$**.** For each sampled vertex v_i, $x_i = 1$ if $v_i \in H_{nis}$; and $x_i = 0$ otherwise. Then,

$$\mathbf{E}[x_i^2] = \mathbf{E}[x_i] = \Pr[v_i \in H_{nis}] = \frac{1}{n}\sum_{v_i \in V}\frac{n'}{n} = \frac{n'}{n},\ \mathrm{Var}[x_i] = \frac{n'}{n}(1-\frac{n'}{n})$$

As $\hat{n'} = \frac{n}{r}\sum_{i=1}^{r} x_i$, so $\mathbf{E}[\hat{n'}] = \frac{n}{r}\cdot r \cdot \mathbf{E}[x_1] = n'$, and $\mathrm{Var}[\hat{n'}] = (\frac{n}{r})^2 \cdot r \cdot \mathrm{Var}[x_1] = \frac{n'(n-n')}{r}$.
Analysis on $\hat{c'}$**.** By the definition of c'_U, we have, $c' - \frac{2n}{\Gamma} \le c'_U \le c'$.
Since α_i has the same value as β_i in Lemma 3.2, except that when u_i is isolated in H, $\alpha_i = 0$, so

$$\begin{aligned}
\mathbf{E}[\alpha_i] &= \Pr[u_i \in H_{nis}]\,\mathbf{E}[\alpha_i | u_i \in H_{nis}] + \Pr[u_i \in H \setminus H_{nis}]\,\mathbf{E}[\alpha_i | u_i \in H \setminus H_{nis}] \\
&= \frac{n'}{n}\cdot\frac{1}{n'}\left(\sum_{u\in H_{nis}\setminus U} 0 + \sum_{u \in H_{nis}\cap U} 2^{-\left\lceil \log\left(\frac{\mathrm{vol}(C_u)}{d_u^{(G)}}\right)\right\rceil}\cdot 2^{\left\lceil \log\left(\frac{\mathrm{vol}(C_u)}{d_u^{(G)}}\right)\right\rceil}\frac{d_u^{(G)}}{\mathrm{vol}(C_u)}\right) + \frac{n-n'}{n}\cdot 0 \\
&= \frac{c'_U}{n} \qquad \text{(since } \textstyle\sum_{u\in U}\frac{d_u^{(G)}}{\mathrm{vol}(C_u)} = c_U\text{)}
\end{aligned}$$

And $\mathrm{Var}[\alpha_i] \le \mathbf{E}[\alpha_i^2] \le 2\cdot\mathbf{E}[\alpha_i] = \frac{2c'_U}{n}$. Then,

$$\mathbf{E}[\hat{c'}] = \frac{n}{r}\cdot\sum_{i=1}^{r}\mathbf{E}[\alpha_i] = \frac{n}{r}\cdot r\cdot\frac{c'_U}{n} = c'_U,\ \mathrm{Var}[\hat{c'}] \le \frac{n^2}{r^2}\cdot r\cdot\frac{2c'_U}{n} = \frac{2c'_U n}{r} \le \frac{2c'n}{r}$$

If $c' < \frac{n}{k}$, by Chebyshev's inequality,

$$\Pr[|\hat{c'} - c'_U| \ge \frac{\varepsilon n}{2k}] \le \frac{\mathrm{Var}[\hat{c'}]\cdot 4k^2}{\varepsilon^2 n^2} \le \frac{2nc'\cdot 4k^2}{r\cdot\varepsilon^2 n^2} \le \frac{8c'\cdot k^2}{64k\cdot n} \le \frac{1}{8}$$

The last inequality holds as $c' < \frac{n}{k}$. Therefore, the error of $\hat{c'}$ is

$$|\hat{c'} - c'| \le |\hat{c'} - c'_U| + |c'_U - c'| \le \frac{\varepsilon n}{2k} + \frac{2n}{\Gamma} \le \frac{\varepsilon n}{2k} + \frac{\varepsilon n}{2k} = \frac{\varepsilon n}{k}$$

Else, when $c' \ge \frac{n}{k}$,

$$\Pr[|\hat{c'} - c'_U| \ge \frac{\varepsilon c'}{2}] \le \frac{4\mathrm{Var}[\hat{c'}]}{\varepsilon^2 c'^2} \le \frac{8nc}{r\cdot\varepsilon^2 c'^2} \le \frac{8n}{64k\cdot c'} \le \frac{1}{8}$$

The last inequality holds as $c' \ge \frac{n}{k}$. And the error of $\hat{c'}$ is

$$|\hat{c'} - c'| \le |\hat{c'} - c'_U| + |c'_U - c'| \le \frac{\varepsilon c'}{2} + \frac{2n}{\Gamma} = \frac{\varepsilon c'}{2} + \frac{\varepsilon n}{2k} \le \varepsilon c'$$

Combining both cases, we have that $\left|\hat{c'} - c'\right| \le \varepsilon\max\{\frac{n}{k}, c'\}$, with probability more than $\frac{7}{8}$. □

Now we are ready to prove Lemma 7.4.

Proof of Lemma 7.4. In the graph H_{nis}, each connected component has at least two vertices, implying $2c' \le n'$. Therefore, $D = n' - c' \ge 2c' - c'$, which simplifies to $0 \le c' \le D$. Consequently, $D \le n' \le 2D$. According to Lemma 7.5, $\left|\hat{c'} - c'\right| \le \varepsilon\cdot\max\{\frac{n}{k}, c'\}$. Therefore, $\left|\hat{c'} - c'\right| \le \varepsilon\cdot\max\left\{\frac{n}{k}, D\right\}$, with probability at least $\frac{7}{8}$.

- Case (I): $0 \le D \le \frac{n}{k}$. Then $n' \le 2D \le 2\frac{n}{k}$. By Chebyshev's inequality,

$$\Pr[\left|\hat{n'} - n'\right| \ge \varepsilon \frac{n}{k}] \le \frac{\mathrm{Var}[\hat{n'}]k^2}{\varepsilon^2 n^2} = \frac{n'(n-n')k^2}{r \cdot \varepsilon^2 n^2} \le \frac{2\frac{n}{k}(n-n')k^2}{64 \cdot kn^2} \le \frac{1}{32}$$

  Therefore, $\hat{n'} \le n' + \varepsilon\frac{n}{k} < (2+1)\frac{n}{k} < 0.5n$, as $k \ge 10$ and $\varepsilon < 1$. In this case, we let $\widehat{D} = \hat{n'} - \hat{c'}$, then

$$\left|\widehat{D} - D\right| \le \left|\hat{n'} - n\right| + \left|\hat{c'} - c'\right| \le \varepsilon\frac{n}{k} + \varepsilon \cdot \max\left\{\frac{n}{k}, D\right\} \le 2\varepsilon \cdot \frac{n}{k}$$

  By union bound over $\hat{n'}$ and $\hat{c'}$, the above inequality is true with probability more than $1 - (\frac{1}{32} + \frac{1}{8}) \ge \frac{3}{4}$.

- Case (II): $\frac{n}{k} < D \le 0.2n$. Since $D \le n' \le 2D$, by Chebyshev's inequality,

$$\Pr[\left|\hat{n'} - n'\right| \ge \frac{\varepsilon}{2}D] \le \frac{4\mathrm{Var}[\hat{n'}]}{\varepsilon^2 D^2} = \frac{4n'(n-n')}{r \cdot \varepsilon^2 D^2} \le \frac{8 \cdot D \cdot (n-D)}{64 \cdot k \cdot D^2} \le \frac{n - \frac{n}{k}}{8 \cdot k \cdot \frac{n}{k}} \le \frac{1}{8}$$

  Therefore, we have $\hat{n'} \le n' + \frac{\varepsilon}{2}D \le (2 + \frac{\varepsilon}{2})D < 0.5n$. In this case, we let $\widehat{D} = \hat{n'} - \hat{c'}$. Then,

$$\left|\widehat{D} - D\right| \le \left|\hat{n'} - n\right| + \left|\hat{c'} - c'\right| \le \frac{\varepsilon}{2}D + \varepsilon \cdot \max\left\{\frac{n}{k}, D\right\} \le 2\varepsilon D$$

  The above inequality is true with probability at least $1 - (\frac{1}{8} + \frac{1}{8}) = \frac{3}{4}$.

- Case (III): $0.6n < D \le n$. Then $n - n' \le n - D \le 0.4n$. By Chebyshev's inequality,

$$\Pr[\left|\hat{n'} - n'\right| \ge \varepsilon \cdot 0.1n] \le \frac{\mathrm{Var}[\hat{n'}]}{\varepsilon^2 \cdot 0.01n^2} = \frac{n'(n-n')}{r \cdot \varepsilon^2 \cdot 0.01n^2} \le \frac{n-n'}{r \cdot \varepsilon^2 \cdot 0.01n} \le \frac{0.4n}{64k \cdot 0.01n} \le \frac{1}{16}$$

  The last inequality follows from the assumption that $k \ge 10$.

  Then with probability at least $1 - \frac{1}{16}$, $\hat{n'} \ge n' - \varepsilon \cdot 0.1n \ge 0.6n - \varepsilon \cdot 0.1n \ge 0.5n$. Conditioned on this event, we let $\widehat{D} = n - \hat{c}$. From the proof of Lemma 3.1, we know that $\left|\widehat{D} - D\right| \le |\hat{c} - c| \le \varepsilon \cdot \max\{\frac{n}{k}, c\} = \varepsilon \cdot \max\{\frac{n}{k}, n - D\}$, with probability at least $\frac{7}{8}$. Thus, with probability at least $1 - (\frac{1}{16} + \frac{1}{8}) \ge \frac{3}{4}$, $\left|\widehat{D} - D\right| \le \varepsilon \cdot \max\{\frac{n}{k}, n - D\}$.

- Can (IV): $0.2n < D \le 0.6n$. If the algorithm uses $\widehat{D} = n - \hat{c}$ to estimate D, then with probability at least $\frac{7}{8} \ge \frac{3}{4}$, we have $\left|\widehat{D} - D\right| \le |\hat{c} - c| \le \varepsilon c \le 4\varepsilon D$ (as when $D > 0.2n$, $c = n - D < 0.8n < 4D$).

  On the other hand, if the algorithm uses $\widehat{D} = \hat{n'} - \hat{c'}$ to estimate D, we have that $0.2n \le n' < n$. By Chebyshev's inequality,

$$\Pr[\left|\hat{n'} - n'\right| \ge \varepsilon \cdot 0.2n] \le \frac{\mathrm{Var}[\hat{n'}]}{\varepsilon^2 \cdot 0.04n^2} = \frac{n'(n-n')}{r \cdot \varepsilon^2 \cdot 0.04n^2} \le \frac{n'0.8n}{r \cdot \varepsilon^2 \cdot 0.04n^2} \le \frac{2}{64k \cdot 0.1} \le \frac{1}{16}$$

  Therefore, with probability more than $1 - (\frac{1}{16} + \frac{1}{8}) \ge \frac{3}{4}$, we have

$$\left|\widehat{D} - D\right| \le \left|\hat{n'} - n\right| + \left|\hat{c'} - c'\right| \le \varepsilon \cdot 0.2n + \varepsilon \cdot \max\left\{\frac{n}{k}, D\right\} \le \varepsilon D + \varepsilon D \le 2\varepsilon D.$$

Combining the above case analysis, we have that with probability more than $\frac{3}{4}$,

$$\left|\widehat{D}-D\right| \leq \begin{cases} 4\varepsilon \cdot \max\left\{\frac{n}{k}, D\right\} & \text{if } 0 \leq D \leq 0.5n \\ 2\varepsilon \cdot \max\{\frac{n}{k}, n-D\} & \text{if } 0.5n < D \leq n \end{cases}$$

Replacing ε with 4ε, the statement of the lemma is satisfied.

Running time analysis. Similar to analysis of running time of Algorithm 1, consider our truncation approach. It ensures that there are at most $O(\Gamma^2 d)$ edges visited during BFS, starting from a sampled vertex u. Consequently, the number of coin flips is at most $O(\log(\Gamma d)$, and the expected running time associated with the sampled vertex u is $O(d_u \log(\Gamma d)$. Since we sample r vertices and each vertex $u \in V$ is sampled with probability $\frac{1}{n}$, the overall running time of Algorithm 7 is $O(r \cdot \frac{1}{n}\sum_{u\in V} d_u \log(\Gamma d))$. Given that we set $r = \frac{k}{\varepsilon^2}$, and $\Gamma = \frac{k}{\varepsilon}$, so the running time is $O(\frac{k}{\varepsilon^2} d \log(\frac{k}{\varepsilon} d))$.

□

Similar to Algorithm 2, by applying median trick to Algorithm 7, we have the following corollary.

Corollary 7.6. *For every $1 \leq i \leq W$, an appropriately amplified version of Algorithm 7 accepts the same inputs as Algorithm 7, except for an additional success parameter δ; and it outputs $\widehat{D}$ satisfying*

$$\left|\widehat{D}-D\right| \leq \varepsilon \cdot \max\left\{\frac{n}{k}, \min\{D, n-D\}\right\}$$

with probability at least $1-\delta$. Here, $D = n - c$ where c is the number of connected components in the subgraph H. The algorithm runs in time $O(\frac{k\log(1/\delta)}{\varepsilon^2} d \log(\frac{k}{\varepsilon} d))$.

7.2.2 Adapting Binary Search for Clustering Cost Estimation

Note that $(D_1, \ldots, D_W)$ is a non-increasing array with values in the range $[0, n-1]$. Therefore, we can utilize binary search as described in Section 5, similar to the approach used for the distance case, to obtain effective estimators for all elements in this sequence. Specifically, we will apply binary search on the sequence $\widehat{D} = (\widehat{D}_1, \ldots, \widehat{D}_W)$ with some newly defined search keys B_i's. Now we define $\widehat{D}_j$'s in the following lemma.

Lemma 7.7. *For any $1 \leq j \leq W$, let $\widehat{D}_j = \min\{\max\{\widehat{D}'_j, 0\}, n-1\}$, where $\widehat{D}'_j$ is the output of Corollary 7.6 with input G, $H = G_j^{(s)}$, $\varepsilon/8$, $k = W$, d, $\delta = 1/(8W)$. Then with probability at least $7/8$, it holds that for* all *$1 \leq j \leq W$, $\widehat{D}_j \in [0, n-1]$, and*

$$\left|\widehat{D}_j - D_j\right| \leq T_j := \frac{\varepsilon}{8} \cdot \max\left\{\frac{n}{W}, \min\{D_j, n-D_j\}\right\}.$$

Proof. According to Corollary 7.6, for any $j \in [W]$, we know that with probability of $1 - \varepsilon/(8W)$, the estimator $\widehat{D}'_j$ satisfies $\left|\widehat{D}'_j - D_j\right| \leq \frac{\varepsilon}{8} \cdot \max\left\{\frac{n}{W}, \min\{D_j, n-D_j\}\right\} = T_j$. Besides, by rounding $\widehat{D}'_j$ to $\widehat{D}_j = \min\{\max\{\widehat{D}'_j, 0\}, n-1\}$, we can guarantee that $D_j \in [0, n-1]$. If $\widehat{D}'_j$ is out of the range $[0, n-1]$, then the rounding process decreases the gap between $\widehat{D}_j$ and D_j, and ensures the error remains within the error bound T_j.

By the union bound on all estimates $\widehat{D}_1, \ldots, \widehat{D}_W$, with probability at least $\frac{7}{8}$, for *all* $j \in [W]$, $\widehat{D}_j \in [0, n-1]$, and $\left|\widehat{D}_j - D_j\right| \leq T_j$. □

Throughout the following, we will assume that for all *j, the inequality* $\left|\widehat{D}_j - D_j\right| \le T_j$ *holds, and* $\widehat{D}_j \in [0, n-1]$. According to Lemma 7.7, this occurs with probability at least $7/8$.

Now we define the endpoints of intervals which partition $[0, n-1]$.

Definition 7.8. Let $0 < \varepsilon < 1$, t_1 be the largest integer such that $n - t_1 \frac{\varepsilon n}{W} \ge n - \frac{n}{W}$, t_2 be the largest integer such that $n - (1+\varepsilon)^{t_2} \frac{n}{W} \ge 0.5n$ (i.e. the largest integer such that $\frac{n}{2(1+\varepsilon)^{t_2}} \ge \frac{n}{W}$), and let t_3 be the largest integer such that $\frac{n}{W}(1 - t_3\varepsilon) \ge \frac{\varepsilon n}{W}$. Note that $t_1 = \lfloor \frac{1}{\varepsilon} \rfloor$, $t_2 = \lfloor \log_{1+\varepsilon} \frac{W}{2} \rfloor$, and $t_3 = \lfloor \frac{1-\varepsilon}{\varepsilon} \rfloor$. Then we define B_i as

$$B_i = \begin{cases} n-1 & \text{if } i = 1 \\ n - i\frac{\varepsilon n}{W} & \text{if } 1 < i \le t_1 \\ n - (1+\varepsilon)^{i-t_1}\frac{n}{W} & \text{if } t_1 < i \le t_1 + t_2 \\ \frac{n}{2(1+\varepsilon)^{i-(t_1+t_2)}} & \text{if } t_1 + t_2 < i \le t_1 + 2t_2 \\ \frac{n}{W}(1 - (i - t_1 - 2t_2)\varepsilon) & \text{if } t_1 + 2t_2 < i \le t_1 + 2t_2 + t_3 \\ 0 & \text{if } i = t := t_1 + 2t_2 + t_3 + 1 \end{cases}$$

We have the following fact.

Fact 7.9. *It holds that*

1. $t = t_1 + 2t_2 + t_3 + 1 = O(\frac{2}{\varepsilon} + 2\log_{1+\varepsilon} W) = O(\log W/\varepsilon)$.
2. *When* $1 \le i \le t_1$, $n - \frac{n}{W} \le B_i \le n$;
   when $t_1 < i \le t_1 + t_2$, $0.5n \le B_i < n - \frac{n}{W}$;
   when $t_1 + t_2 < i \le t_1 + 2t_2$, $\frac{n}{W} < B_i \le 0.5n$;
   and when $t_1 + 2t_2 < B_i \le t_1 + 2t_2 + t_3$, $\frac{\varepsilon n}{W} \le B_i < \frac{n}{W}$.
3. *the gap between neighboring endpoints is,*

$$B_{i-1} - B_i = \begin{cases} \frac{\varepsilon n}{W} & \text{if } 1 < i \le t_1 \\ \varepsilon(1+\varepsilon)^{i-1}\frac{n}{W} = \varepsilon(n - B_{i-1}) & \text{if } t_1 + 2 \le i \le t_1 + t_2 \\ (1+\varepsilon)B_i - B_i = \varepsilon B_i & \text{if } t_1 + t_2 + 2 \le i \le t_1 + 2t_2 \\ \frac{\varepsilon n}{W} & \text{if } t_1 + 2t_2 + 2 \le i \le t_1 + 2t_2 + t_3 \end{cases}$$

4. $B_{t_1} - B_{t_1+1} \ge n - \frac{n}{W} - (n - (1+\varepsilon)\frac{n}{W}) = \frac{\varepsilon n}{W}$;
   $B_{t_1+t_2} - B_{t_1+t_2+1} \ge 0.5n - \frac{n}{2(1+\varepsilon)} = \frac{\varepsilon n}{2(1+\varepsilon)} = \varepsilon B_{t_1+t_2+1}$;
   $B_{t_1+2t_2-1} - B_{t_1+2t_2} = \varepsilon B_{t_1+2t_2} \ge \frac{\varepsilon n}{W}$;
   $B_{t_1+2t_2} - B_{t_1+2t_2+1} \ge \frac{n}{W} - \frac{n}{W}(1-\varepsilon) = \frac{\varepsilon n}{W}$
5. $n - (t_1+1)\frac{\varepsilon n}{W} < n - \frac{n}{W}$, *and thus* $B_{t_1} = n - t_1\frac{n}{W} < n - \frac{n}{W}(1-\varepsilon)$;
   $n - (1+\varepsilon)^{t_2+1}\frac{n}{W} < \frac{n}{2}$, *and thus* $n - B_{t_1+t_2} > \frac{n}{2(1+\varepsilon)}$;
   $\frac{n}{2(1+\varepsilon)^{t_2+1}} < \frac{n}{W}$, *and thus* $B_{t_1+2t_2} < (1+\varepsilon)\frac{n}{W}$.

For each $j \in [W]$, we define the error bound as $T_j = \frac{\varepsilon}{8}\max\{\frac{n}{W}, \min\{D_j, n-D_j\}\}$. Then, we prove that the sequence of B_i's defined above form a valid discretization of $[0, n-1]$ w.r.t. $(D_1, \ldots, D_W)$ and error bound $T = (T_1, \ldots, T_W)$.

Lemma 7.10. *The sequence of endpoints $(B_1, \ldots, B_t)$ defined in Definition 7.8 is a valid discretization of $[0, n-1]$ w.r.t. $(D_1, \ldots, D_W)$ and error bound $T_j = \frac{\varepsilon}{8}\max\{\frac{n}{W}, \min\{D_j, n-D_j\}\}$.*

Proof. We consider the sequence $(\widehat{D}_1, \ldots, \widehat{D}_W)$ as the approximation of $D = (D_1, \ldots, D_W)$, where each $\widehat{D}_j$ is defined as in Lemma 7.7. According to the previous assumption, it holds that for all j, $\widehat{D}_j \in [0, n-1]$ and $\left|\widehat{D}_j - D_j\right| \leq T_j$.

Now we consider any $j \in [W]$. Suppose that $D_j \in [B_{i+1}, B_i]$.

When $i = 1$, $B_{i+1} - B_{i+2} = \frac{\varepsilon n}{W} > T_j$.

When $2 \leq i \leq t_1 - 1$, we have $B_{i-1} - B_i = \frac{\varepsilon n}{W}$ and $B_{i+1} - B_{i+2} \geq \frac{\varepsilon n}{W}$. In this case, $D_j \geq B_{i+1} > n - \frac{n}{W}$, and as $(1+\varepsilon)^2 \leq 4$, we have $T_j = \frac{\varepsilon n}{8W} < \min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\}$.

When $i = t_1$, we have $B_{i-1} - B_i = \frac{\varepsilon n}{W}$, and $B_{i+1} - B_{i+2} = \varepsilon(n - B_{i+1}) = \varepsilon(1+\varepsilon)\frac{n}{W}$. In this case,

$$T_j = \frac{\varepsilon}{8}\max\{\frac{n}{W}, n - D_j\} \leq \frac{\varepsilon}{8}\max\{\frac{n}{W}, n - B_{i+1}\} = \frac{\varepsilon}{8}(1+\varepsilon)\frac{n}{W} \leq \min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\}$$

When $i = t_1 + 1$, we have $B_{i-1} - B_i \geq \frac{\varepsilon n}{W}$, and $B_{i+1} - B_{i+2} = \varepsilon(n - B_{i+1}) = \varepsilon(1+\varepsilon)^2\frac{n}{W}$. In this case,

$$T_j = \frac{\varepsilon}{8}\max\{\frac{n}{W}, n - D_j\} \leq \frac{\varepsilon}{8}\max\{\frac{n}{W}, n - B_{i+1}\} = \frac{\varepsilon}{8}(1+\varepsilon)^2\frac{n}{W} \leq \max\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\}$$

When $t_1 + 2 \leq i \leq t_1 + t_2 - 2$, we have $B_{i-1} - B_i = \varepsilon(n - B_{i-1}) = \frac{\varepsilon}{(1+\varepsilon)^2}(n - B_{i+1})$, and $B_{i+1} - B_{i+2} = \varepsilon(n - B_{i+1})$. In this case, $D_j \leq B_{t_1+2} < n - \frac{n}{W}$, and $D_j \geq B_{t_2-2} > 0.5n$. Thus,

$$T_j = \frac{\varepsilon}{8}(n - D_j) \leq \frac{\varepsilon}{8}(n - B_{i+1}) < \frac{\varepsilon}{(1+\varepsilon)^2}(n - B_{i+1}) = \min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\}$$

When $i = t_1 + t_2 - 1$, we have $B_{i-1} - B_i = \frac{\varepsilon}{(1+\varepsilon)^2}(n - B_{i+1})$. Since $B_{i+1} \geq 0.5n$, then $B_{i+1} - B_{i+2} = B_{t_1+t_2} - B_{t_1+t_2+1} \geq \frac{\varepsilon n}{2(1+\varepsilon)} \geq \frac{\varepsilon}{1+\varepsilon}(n - B_{i+1})$. In this case,

$$T_j = \frac{\varepsilon}{8}(n - D_j) \leq \frac{\varepsilon}{8}(n - B_{i+1}) \leq \frac{\varepsilon}{(1+\varepsilon)^2}(n - B_{i+1}) = \min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\}$$

When $i = t_1 + t_2$, we have $B_{i-1} - B_i = B_{t_1+t_2-1} - B_{t_1+t_2} = \frac{\varepsilon}{1+\varepsilon}(n - B_{t_1+t_2}) > \frac{\varepsilon n}{2(1+\varepsilon)^2}$, and $B_{i+1} - B_{i+2} = \varepsilon B_{i+2} = \frac{\varepsilon n}{2(2+\varepsilon)^2}$. In this case,

$$T_j = \frac{\varepsilon}{8}\min\{n - D_j, D_j\} \leq \frac{\varepsilon}{16}n \leq \frac{\varepsilon}{2(1+\varepsilon)^2}n = \min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\}$$

When $t_1 + t_2 + 1 \leq i \leq t_1 + 2t_2 - 2$, we have $B_{i-1} - B_i \geq \varepsilon B_i$, and $B_{i+1} - B_{i+2} = \varepsilon B_{i+2} = \frac{\varepsilon}{(1+\varepsilon)^2}B_i$. In this case, $D_j \leq B_{t_1+t_2+2} < 0.5n$, and $D_j \geq B_{t_1+t_2+t_3-2} > \frac{n}{W}$. Thus,

$$T_j = \frac{\varepsilon}{8}D_j \leq \frac{\varepsilon}{8}B_i < \frac{\varepsilon}{(1+\varepsilon)^2}B_i = \min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\}$$

When $i = t_1 + 2t_2 - 1$, we have $B_{i-1} - B_i \geq \varepsilon B_i$. Since $\frac{B_{i+1}}{1+\varepsilon} = \frac{B_{t_1+2t_2}}{1+\varepsilon} < \frac{n}{W}$, $B_i = (1+\varepsilon)B_{i+1} \leq (1+\varepsilon)^2 \frac{n}{W}$. Then, $B_{i+1} - B_{i+2} \geq \frac{\varepsilon n}{W} \geq \frac{\varepsilon}{(1+\varepsilon)^2} B_i$. Thus,

$$T_j = \frac{\varepsilon}{8} D_j \leq \frac{\varepsilon}{8} B_i < \frac{\varepsilon}{(1+\varepsilon)^2} B_i = \min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\}$$

When $t_1 + 2t_2 \leq i$, we have $B_{i-1} - B_i \geq \frac{\varepsilon n}{W}$ and $B_{i+1} - B_{i+2} \geq \frac{\varepsilon n}{W}$. In this case, $D_j \leq B_i \leq B_{t_1 t_2 + t_3 + 2} < \frac{n}{W}$. Thus, $T_j = \frac{\varepsilon n}{8W} < \min\{B_{i-1} - B_i, B_{i+1} - B_{i+2}\}$.

Therefore, the sequence $(B_1, \ldots, B_t)$ is a valid discretization of $[0, n-1]$ w.r.t. $(D_1, \ldots, D_W)$ and the error bound $T = (T_1, \ldots, T_W)$. □

The Algorithm Now we are ready to exploit the binary search based algorithm for succinctly approximating the sequence $D = (D_1, \ldots, D_W)$ to estimate the clustering cost $\text{cost}^{(s)}(G)$. The algorithm simply performs binary search Algorithm 3 on the estimated array $\widehat{D} = \{\widehat{D}_1, \ldots, \widehat{D}_W, 0\}$ with search keys $B_1, \ldots, B_t$ as defined in Definition 7.8, where $\widehat{D}_j$'s are estimators of D_j's as defined in Lemma 7.7. Then we obtain the corresponding indices $\hat{j}_1, \ldots, \hat{j}_t$. For $i \in \{1, \ldots, t-1\}$, let $\hat{J}_i = \{\hat{j}_i, \ldots, \hat{j}_{i+1} - 1\}$. Note that $\hat{J}_1, \ldots, \hat{J}_{t-1}$ form a partition of the index set $[W] = \{1, \ldots, W\}$. For any $j \in \hat{J}_i$, we define $\overline{D}_j = B_i$ as the estimate for D_j and then use $\overline{D}_j$'s to estimate the cost $\text{cost}^{(s)}(G)$. The algorithm is described in Algorithm 8. Note that we estimate $c_j^{(s)}$ and D_j separately, and then estimate $\text{cost}^{(s)}(G)$ by multiplying the estimates of $c_j^{(s)}$ and D_j. Furthermore, for the same reason as in distance case, we do not need to access the whole array $\widehat{D}$, but only access $\text{poly}(\log W)$ values $\widehat{D}_i$ in it. We also store previously estimated value of $\widehat{D}_i$ to maintain consistency.

Algorithm 8: APPCOSTSIM(G, ε, d, W)

input : graph G, approximation parameter ε

output: $\widehat{\text{cost}^{(s)}}(G)$, which estimates $\text{cost}^{(s)}(G)$

set t according to Definition 7.8

get estimates $\{\hat{c}_1^{(s)}, \ldots, \hat{c}_W^{(s)}\}$ from Corollary 3.3 with parameters $G, H = G_j, \varepsilon, k = 1, d^{(G)} = d \cdot \lceil \frac{4}{\varepsilon} \rceil, \delta = 1/8W$

for $1 \leq i \leq t$ **do**

  set B_i according to Definition 7.8

  invoke BINARYSEARCH$(\widehat{D}, 1, W, B_i)$ and get output index $\hat{j}_i$, where $\widehat{D} = (\widehat{D}_1, \ldots, \widehat{D}_W, 0)$ and each $\widehat{D}_j = \min\{\max\{\widehat{D}'_j, 0\}, n-1\}$ and $\widehat{D}'_j$ is the output of the algorithm in Corollary 7.6 with input $G, H = G_j^{(s)}, \varepsilon/8, k = W$, $\delta = 1/(8W)$ ▷ for consistency, reuse stored value of $\widehat{D}_j$, if previously estimated

for any $j \in \hat{J}_i := \{\hat{j}_i, \ldots, \hat{j}_{i+1} - 1\}$, we let $\overline{D}_j = \overline{D}_{\hat{j}_i} = B_i$

set $\widehat{\text{cost}^{(s)}}(G) = \frac{1}{2} \sum_{j=1}^{W} (\hat{c}_j^{(s)} + n - 1) \cdot \overline{D}_j$

output $\widehat{\text{cost}^{(s)}}(G)$ and the sequence $(\hat{j}_1, \ldots, \hat{j}_t)$

Now we first give the following guarantee on the estimates $\overline{D}_j$'s.

Lemma 7.11. *Assume that for* all $j \in [W]$*, the inequality* $\left|\widehat{D}_j - D_j\right| \leq T_j$ *holds. Then it holds that for all* $j \in [W]$*,*

$$\left|\overline{D}_j - D_j\right| \leq 5\varepsilon \cdot \max\left\{\frac{n}{W}, \min\{D_j, n - D_j\}\right\}.$$

Proof. By Lemma 7.10, the sequence of endpoints $(B_1,\dots,B_t)$ defined in Definition 7.8 is a valid discretization of $[0,n-1]$ w.r.t. $(D_1,\dots,D_W)$ and error bound $T=(T_1,\dots,T_W)$. Thus, by Lemma 5.4, for every $i\in[1,t-1]$, and every $j\in\hat{J}_i$, we have $B_{i+2}\le D_j\le B_{i-1}$. By case distinction,

- When $i=1$, $B_{i+2}\le D_j\le B_1=n$, then $\left|\overline{D}_j-D_j\right|\le B_1-B_{i+2}=2\frac{\varepsilon n}{W}$.
- When $i\le t_1-2$, we have $B_{i-1}-B_i=\frac{\varepsilon n}{W}$, and $B_i-B_{i+2}=2\frac{\varepsilon n}{W}$.
  Thus, $\left|\overline{D}_j-D_j\right|\le\max\{B_{i-1}-B_i,B_i-B_{i+2}\}=2\frac{\varepsilon n}{W}$.
- When $i=t_1-1$, since $B_{t_1}<n-\frac{n}{W}(1-\varepsilon)$, then $B_{t_1-1}-B_{t_1+1}=(B_{t_1}+\frac{\varepsilon n}{W})-(n-(1+\varepsilon)\frac{n}{W})<3\frac{\varepsilon n}{W}$. Besides, $B_{i-1}-B_i=\frac{\varepsilon n}{W}$. Thus, $\left|\overline{D}_j-D_j\right|\le\max\{B_{i-1}-B_i,B_i-B_{i+2}\}<3\frac{\varepsilon n}{W}\le3\varepsilon\max\{\frac{n}{W},n-D_j\}$.
- When $i=t_1$, we have $B_i-B_{i+2}=B_{t_1}-B_{t_1+2}<(n-\frac{n}{W}(1-\varepsilon))-(1-(1+\varepsilon)^2)\frac{n}{W}\le4\frac{\varepsilon n}{W}$. Besides, $B_{i-1}-B_i=\frac{\varepsilon n}{W}$. Thus, $\left|\overline{D}_j-D_j\right|<4\frac{\varepsilon n}{W}\le4\varepsilon\max\{\frac{n}{W},n-D_j\}$.
- When $i=t_1+1$, we have $\begin{cases}\text{if } D_j>\overline{D}_j, & D_j-\overline{D}_j\le B_{i-1}-B_i<(n-\frac{n}{W}(1-\varepsilon))-(n-(1+\varepsilon)\frac{n}{W})=2\frac{\varepsilon n}{W}\\ \text{else,} & \overline{D}_j-D_j\le B_i-B_{i+2}=((1+\varepsilon)^2-1)(n-B_i)\le3\varepsilon(n-D_j)\end{cases}$
- When $t_1+2\le i\le t_1+t_2-2$, since $D_j\le B_{i-1}$, then $n-D_j\ge n-B_{i-1}$; and if $D_j\le\overline{D}_j=B_i$, then $n-D_j\ge n-B_i$. Thus, $\begin{cases}\text{if } D_j>\overline{D}_j, & D_j-\overline{D}_j\le B_{i-1}-B_i=\varepsilon(n-B_{i-1})\le\varepsilon(n-D_j)\\ \text{else,} & \overline{D}_j-D_j\le B_i-B_{i+2}=((1+\varepsilon)^2-1)(n-B_i)\le3\varepsilon(n-D_j)\end{cases}$
- When $i=t_1+t_2-1$, since $n-B_{t_1+t_2}>\frac{n}{2(1+\varepsilon)}$, then $n-B_i=\frac{n-B_{t_1+t_2}}{1+\varepsilon}>\frac{n}{2(1+\varepsilon)^2}$. Thus, $B_i-B_{i+2}<n-\frac{n}{2(1+\varepsilon)^2}-\frac{n}{2(1+\varepsilon)}\le\frac{5\varepsilon n}{2(1+\varepsilon)^2}$. As $\frac{5\varepsilon n}{2(1+\varepsilon)^2}=\frac{5\varepsilon}{1+\varepsilon}B_{i+2}\le5\varepsilon D_j$, and $\frac{5\varepsilon n}{2(1+\varepsilon)^2}<5\varepsilon(n-B_i)$, we have
  $\begin{cases}\text{if } D_j>\overline{D}_j, & D_j-\overline{D}_j\le\varepsilon(n-B_{i-1})\le\varepsilon(n-D_j)\\ \text{else,} & \overline{D}_j-D_j\le B_i-B_{i+2}<\frac{5\varepsilon n}{2(1+\varepsilon)^2}<5\varepsilon\min\{D_j,n-D_j\}\end{cases}$
- When $i=t_1+t_2$, $B_i-B_{i+2}<n-\frac{n}{2(1+\varepsilon)}-\frac{n}{2(1+\varepsilon)^2}\le\frac{5\varepsilon n}{(1+\varepsilon)^2}$. As $\frac{5\varepsilon n}{2(1+\varepsilon)^2}=5\varepsilon B_{i+2}\le5\varepsilon D_j$, and $\frac{5\varepsilon n}{2(1+\varepsilon)^2}<\frac{5\varepsilon}{1+\varepsilon}(n-B_i)<5\varepsilon(n-B_i)$, we have $\begin{cases}\text{if } D_j>\overline{D}_j, & D_j-\overline{D}_j\le\varepsilon(n-B_{i-1})\le\varepsilon(n-D_j)\\ \text{else,} & \overline{D}_j-D_j\le B_i-B_{i+2}<\frac{5\varepsilon n}{2(1+\varepsilon)^2}<5\varepsilon\min\{n-D_j,D_j\}\end{cases}$
- When $i=t_1+t_2+1$, $B_{i-1}-B_i<n-\frac{n}{2(1+\varepsilon)}-\frac{n}{2(1+\varepsilon)}=\frac{\varepsilon n}{1+\varepsilon}$. Note that $\frac{\varepsilon n}{1+\varepsilon}=2\varepsilon B_i$ and $\frac{\varepsilon n}{1+\varepsilon}<2\varepsilon(n-B_{i-1})$. Then, $\begin{cases}\text{if } D_j>\overline{D}_j, & D_j-\overline{D}_j\le B_{i-1}-B_i<\frac{\varepsilon n}{1+\varepsilon}<2\varepsilon\min\{B_i,(n-B_{i-1})\}\le2\varepsilon\min\{\varepsilon D_j,n-D_j\}\\ \text{else,} & \overline{D}_j-D_j\le B_i-B_{i+2}=((1+\varepsilon)^2-1)B_{i+2}\le3\varepsilon B_{i+2}\le3\varepsilon D_j\end{cases}$
- When $t_1+t_2+2\le i\le t_1+2t_2-2$, we have $B_{i-1}-B_i=\varepsilon B_i$, and $B_i-B_{i+2}=((1+\varepsilon)^2-1)B_{i+2}\le3\varepsilon B_{i+2}$. Since $D_j\ge B_{i+1}$, then $\begin{cases}\text{if } D_j>\overline{D}_j=B_i, & D_j-\overline{D}_j\le B_{i-1}-B_i\le\varepsilon D_j\\ \text{else,} & \overline{D}_j-D_j\le B_i-B_{i+2}\le3\varepsilon D_j\end{cases}$
- When $i=t_1+2t_2-1$, since $B_{i+1}=B_{t_1+2t_2}<(1+\varepsilon)\frac{n}{W}$, then $B_i-B_{i+2}=(1+\varepsilon)B_{i+1}-B_{i+2}<(1+\varepsilon)^2\frac{n}{W}-\frac{n}{W}(1-\varepsilon)\le4\frac{\varepsilon n}{W}$. Thus, $\begin{cases}\text{if } D_j>\overline{D}_j, & D_j-\overline{D}_j\le B_{i-1}-B_i=\varepsilon B_i\le\varepsilon D_j\\ \text{else,} & \overline{D}_j-D_j\le B_i-B_{i+2}<4\frac{\varepsilon n}{W}\end{cases}$
- When $i=t_1+2t_2$, we have $\begin{cases}\text{if } D_j>\overline{D}_j, & D_j-\overline{D}_j\le B_{i-1}-B_i=\varepsilon B_i\le\varepsilon D_j\\ \text{else,} & \overline{D}_j-D_j\le B_i-B_{i+2}<(1+\varepsilon)\frac{n}{W}-\frac{n}{W}(1-2\varepsilon)=3\frac{\varepsilon n}{W}\end{cases}$

- When $i = t_1 + 2t_2 + 1$, we have $\begin{cases} \text{if } D_j > \overline{D}_j, & D_j - \overline{D}_j \le B_{i-1} - B_i < (1+\varepsilon)\frac{n}{W} - \frac{n}{W}(1-\varepsilon) = 2\frac{\varepsilon n}{W} \\ \text{else,} & \overline{D}_j - D_j \le B_i - B_{i+2} = 2\frac{\varepsilon n}{W} \end{cases}$
- When $t_1 + 2t_2 + 2 \le i$, we have $B_{i-1} - B_i = \frac{\varepsilon n}{W}$, and $B_i - B_{i+2} = 2\frac{\varepsilon n}{W}$.
  Thus, $\left|\overline{D}_j - D_j\right| \le \max\{B_{i-1} - B_i, B_i - B_{i+2}\} = 2\frac{\varepsilon n}{W}$.
- When $i = t$, $\left|\overline{D}_j - D_j\right| \le B_{t-1} - B_t = B_{t_1+2t_2+t_3} - B_t < 2\frac{\varepsilon n}{W}$.

Therefore, for *all* $j \in [W]$, we have that $\left|\overline{D}_j - D_j\right| \le 5\varepsilon \max\{\frac{n}{W}, \min\{D_j, n - D_j\}\}$. □

Analysis of Algorithm 8 Now we give the analysis on the approximation ratio of Algorithm 8 as stated in Theorem 1.4.

Theorem 1.4. *Let G be a weighted graph with edge weights in $\{1, \ldots, W\}$ with average (unweighted) degree d. Assume that $W \le n$ and let $0 < \varepsilon < 1$ be a parameter. Algorithm 8 outputs an estimate $\widehat{\text{cost}^{(s)}}(G)$ of the single-linkage clustering cost $\text{cost}^{(s)}(G)$ in the similarity graph such that with probability at least $3/4$,*

$$(1-\varepsilon)\text{cost}^{(s)}(G) \le \widehat{\text{cost}^{(s)}}(G) \le (1+\varepsilon)\text{cost}^{(s)}(G).$$

The query complexity and running time of the algorithm are $O(\frac{Wd}{\varepsilon^3}\log^4(\frac{Wd}{\varepsilon}))$ in expectation.

Proof of Theorem 1.4. Setting $\delta = \frac{1}{8W}$ and $k = 1$, we obtain estimates $\{\hat{c}_1^{(s)}, \ldots, \hat{c}_W^{(s)}\}$ from Corollary 3.3, such that each estimate $\hat{c}_j^{(s)}$ satisfies $\left|\hat{c}_j^{(s)} - c_j^{(s)}\right| \le \varepsilon n$ with probability at least $1 - \frac{1}{8W}$. Using the union bound, all these estimates simultaneously have additive error εn with probability at least $\frac{7}{8}$.

Besides, from Lemma 7.7 and Lemma 7.11, with probability at least $\frac{7}{8}$, it holds that for *all* $j \in [W]$, $\left|\overline{D}_j - D_j\right| \le 5\varepsilon \cdot \max\left\{\frac{n}{W}, \min\{D_j, n - D_j\}\right\}$. Therefore, for *all* $j \in [W]$, both $\overline{D}_j$ and $\hat{c}_j^{(s)}$ satisfy the above error guarantee with probability at least $1 - \frac{1}{8} - \frac{1}{8} = \frac{3}{4}$. For the remainder of the proof, we assume that this event holds.

Denote $A_j = c_j^{(s)} + n - 1$, and $\hat{A}_j = \hat{c}_j^{(s)} + n - 1$. Since $1 \le c_j \le n$, we have $n \le A_j \le 2n$. Furthermore, it holds that $\left|\hat{A}_j - A_j\right| = \left|\hat{c}_j^{(s)} - c_j^{(s)}\right| \le \varepsilon n \le \varepsilon A_j$.

Case (I): $0.5n < D_j \le n$. In this case, we have that $n < 2D_j$, and thus

$$\left|\overline{D}_j - D_j\right| \le 5\varepsilon \cdot \max\left\{\frac{n}{W}, n - D_j\right\} \le 5\varepsilon\left(\frac{2}{W} + 2 - 1\right)D_j \le 15\varepsilon D_j.$$

Furthermore,

$$(1-\varepsilon)(1-15\varepsilon)A_j \cdot D_j \le \hat{A}_j \cdot \overline{D}_j \le (1+\varepsilon)(1+15\varepsilon)A_j \cdot D_j$$

That is,

$$\left|\hat{A}_j \cdot \overline{D}_j - A_j \cdot D_j\right| \le (16\varepsilon + 15\varepsilon^2)A_j \cdot D_j \le 31\varepsilon A_j \cdot D_j.$$

Case (II): $0 \le D \le 0.5n$. In this case, we have that $\left|\overline{D}_j - D_j\right| \le 5\varepsilon \cdot \max\left\{\frac{n}{W}, D_j\right\} \le 5\varepsilon\left(\frac{n}{W} + D_j\right)$. Thus,

$$\hat{A}_j \cdot \overline{D}_j \le (1+\varepsilon)A_j\left(D_j + 5\varepsilon D_j + 5\varepsilon\frac{n}{W}\right) \le (1+\varepsilon)(1+5\varepsilon)A_j \cdot D_j + 5\varepsilon(1+\varepsilon)\frac{2n^2}{W}$$

The last inequality follows from $A_i \le 2n$. Furthermore,

$$\hat{A}_j \cdot \overline{D}_j \ge (1-\varepsilon)A_j\left(D_j - 5\varepsilon D_j - 5\varepsilon\frac{n}{W}\right) \ge (1-\varepsilon)(1-5\varepsilon)A_j \cdot D_j - 5\varepsilon(1-\varepsilon)\frac{2n^2}{W}$$

Therefore,

$$\left|\hat{A}_j \cdot \overline{D}_j - A_j \cdot D_j\right| \le (6\varepsilon + 5\varepsilon^2)A_j \cdot D_j + 10\varepsilon(1+\varepsilon)\frac{n^2}{W} \le 11\varepsilon A_j \cdot D_j + 20\varepsilon\frac{n^2}{W}.$$

Combining the analysis above, we have that for any $j \le W$, it holds that

$$\left|\hat{A}_j \cdot \overline{D}_j - A_j \cdot D_j\right| \le 31\varepsilon A_j \cdot D_j + 20\varepsilon\frac{n^2}{W}.$$

By Fact 7.2, $\mathrm{cost}^{(s)}(G) \ge \frac{n(n-1)}{2}$, and when $n \ge 2$, $\mathrm{cost}^{(s)}(G) \ge \frac{1}{4}n^2$. Besides, $\mathrm{cost}^{(s)}(G) = \frac{1}{2}\sum_{j=1}^{W}(c_j^{(s)} + n - 1)(n - c_j^{(s)}) = \frac{1}{2}\sum_{j=1}^{W} A_j \cdot D_j$. Now we can bound the error of $\widehat{\mathrm{cost}^{(s)}}(G)$ by

$$\left|\widehat{\mathrm{cost}^{(s)}}(G) - \mathrm{cost}^{(s)}(G)\right| \le \frac{1}{2}\sum_{j=1}^{W}\left|\hat{A}_j \cdot \overline{D}_j - A_j \cdot D_j\right| \le \frac{31\varepsilon}{2}\sum_{j=1}^{W} A_j \cdot D_j + 20\varepsilon n^2 \le 120\varepsilon\mathrm{cost}^{(s)}(G)$$

Replacing ε with $\varepsilon/120$ achieves a $(1+\varepsilon)$ approximation factor.

Running time analysis. According to Corollary 3.3, each $\hat{c}_j^{(s)}$ is obtained in $O(\frac{d}{\varepsilon^2}\log(\frac{d}{\varepsilon})\cdot\log(\frac{1}{\delta})) = O(\frac{d}{\varepsilon^2}\log(\frac{d}{\varepsilon})\cdot\log W)$ time, and thus the the sequence of estimates $\{\hat{c}_1^{(s)},\ldots,\hat{c}_W^{(s)}\}$ are obtained in $O(\frac{Wd}{\varepsilon^2}\log(\frac{d}{\varepsilon})\cdot\log W)$ time.

Besides, Algorithm 8 invokes BINARYSEARCH for $t = O(\log(W/\varepsilon)/\varepsilon)$ search keys, and each invocation of BINARYSEARCH accesses $O(\log W)$ estimates $\widehat{D}_j$. Thus, the algorithm accesses at most $O(\log^2(\frac{W}{\varepsilon})/\varepsilon)$ estimates $\widehat{D}_j$ in $\widehat{D}$. According to Corollary 7.6, each estimate of $\widehat{D}_j$ can be obtained in $O(\frac{W}{\varepsilon^2}d\log(\frac{W}{\varepsilon}d)\cdot\log(1/\delta)) = O(\frac{Wd}{\varepsilon^2}\log^2(\frac{Wd}{\varepsilon}))$ time.

Thus, the running time of Algorithm 8 is:

$$O(\frac{Wd}{\varepsilon^2}\log(\frac{d}{\varepsilon})\cdot\log W) + O(\log^2(\frac{W}{\varepsilon})/\varepsilon)\cdot O(\frac{Wd}{\varepsilon^2}\log^2(\frac{Wd}{\varepsilon})) = O(\frac{Wd}{\varepsilon^3}\log^4(\frac{Wd}{\varepsilon})) = \tilde{O}(\frac{Wd}{\varepsilon^3})$$

□

7.3 Estimating the Profile Vector

Recall that $\mathrm{cost}^{(s)}(G) = \sum_{i=1}^{n}\mathrm{cost}_k^{(s)}$, where $\mathrm{cost}_k^{(s)} = \sum_{i=1}^{n-k} w_i^{(s)}$ and $w_i^{(s)}$ is the i-th largest weight in MaxST. We now give a sublinear time algorithm to approximate the SLC profile vector $(\mathrm{cost}_1^{(s)},\ldots,\mathrm{cost}_n^{(s)})$ and prove the following theorem.

Theorem 1.5. *Assume that $W \le n$ and let $\varepsilon < 1$. Algorithm 9 generates a succinct representation of an approximation $(\widehat{\mathrm{cost}^{(s)}}_1,\ldots,\widehat{\mathrm{cost}^{(s)}}_n)$ of the SLC profile vector $(\mathrm{cost}_1^{(s)},\ldots,\mathrm{cost}_n^{(s)})$ in the similarity graph such that with probability at least $3/4$, it holds that*

$$\sum_{k=1}^{n}|\widehat{\mathrm{cost}^{(s)}}_k - \mathrm{cost}_k^{(s)}| \le \varepsilon\cdot\mathrm{cost}^{(s)}(G).$$

The query complexity and running time of the algorithm are $O(\frac{Wd}{\varepsilon^3}\log^4(\frac{Wd}{\varepsilon}))$ in expectation.

We will first give a formula to calculate $\mathrm{cost}_k^{(s)}$, which is slightly different from the one for cost_k in the distance case. Next we give an algorithm to estimate the profile, and the corresponding analysis.

Formulas of Profile Recall that $w_i^{(s)}$ is the weight of the i-th maximum edge's weight, in MaxST. We can derive $\text{cost}_k^{(s)}$ as follows:

$$\begin{aligned}
\text{cost}_k^{(s)} &= \sum_{i=1}^{n-k} w_i^{(s)} \\
&= n_W \cdot W + n_{W-1} \cdot (W-1) + \cdots + (n-k-n_W - \cdots - n_{w_{n-k}^{(s)}+1}) \cdot w_{n-k}^{(s)} \\
&= (c_{W+1}^{(s)} - c_W^{(s)}) \cdot W + (c_W^{(s)} - c_{W-1}^{(s)}) \cdot (W-1) + \cdots + (n-k-c_{W+1}^{(s)} + c_{w_{n-k}^{(s)}+1}^{(s)}) \cdot w_{n-k}^{(s)} \\
&\qquad\qquad \text{(since } n_j = c_{j+1}^{(s)} - c_j^{(s)}\text{)} \\
&= (n - c_W^{(s)}) \cdot W + (c_W^{(s)} - c_{W-1}^{(s)}) \cdot (W-1) + \cdots + (c_{w_{n-k}^{(s)}+1}^{(s)} - k) \cdot w_{n-k}^{(s)} \qquad \text{(since } c_{W+1}^{(s)} = n\text{)} \\
&= n \cdot W - c_W^{(s)} - c_{W-1}^{(s)} + \cdots - c_{w_{n-k}^{(s)}+1}^{(s)} - k \cdot w_{n-k}^{(s)} \\
&= \sum_{j=w_{n-k}^{(s)}+1}^{W} (n - c_j^{(s)}) + (n-k) \cdot w_{n-k}^{(s)} \qquad (5)
\end{aligned}$$

We can derive the following lemma, which is similar to Lemma 6.6. The main idea of the lemma is, for a given weight j, we approximate the rank of the first edge with weight j in MaxST. Then, we can take k as its rank, and calculate $\text{cost}_k^{(s)}$.

Lemma 7.12. *Given any integer $j \in \{1, ..., W\}$ and let $k = c_j^{(s)}$, where $c_j^{(s)}$ is the number of connected components in $G_j^{(s)}$. Then*

$$\text{cost}_k^{(s)} = \sum_{i=j+1}^{W} (n - c_i^{(s)}) + (n - c_j^{(s)}) \cdot j.$$

Proof. Let n_j be the number of edges in MaxST with weight j, then $n_j = c_{j+1}^{(s)} - c_j^{(s)}$, $\forall 1 \le j \le W$, and $c_{W+1} = n$. Thus, the number of edges in MaxST with weights at least j is:

$$n_W + n_{W-1} + \cdots + n_j = (n - c_W^{(s)}) + (c_W^{(s)} - c_{W-1}^{(s)}) + \cdots + (c_{j+1}^{(s)} - c_j^{(s)}) = n - c_j^{(s)}$$

As $k = c_j^{(s)}$, the edge ordering $n-k = n - c_j^{(s)}$ in MaxST has weight $w_{n-k}^{(s)} = j$. Then according to Eq. (5), we have

$$\text{cost}_k^{(s)} = \text{cost}_{c_j}^{(s)}(G) = \sum_{i=j+1}^{W} (n - c_i^{(s)}) + (n - c_j^{(s)}) \cdot j$$

□

Algorithm to Estimate Profile Similarly to the distance case, we first round each c_j for $1 \le j \le W$ to its nearest interval, such as $(B_{i+1}, B_i]$. This allows us to estimate cost_k for k corresponding to all the interval endpoints. The detailed algorithm is given in Algorithm 9.

We note that B_i serves as an approximation for $D_{\hat{j}_i} = n - c_{\hat{j}_i}$. In Lemma 7.12, if we substitute k with $n - \overline{D}_{\hat{j}_i} = n - B_i$ and j with $\hat{j}_i$, we arrive at the following expression:

$$\overline{\text{cost}^{(s)}}_{n-B_i} = \sum_{j=\hat{j}_i+1}^{W} \overline{D}_j + \overline{D}_{\hat{j}_i} \cdot \hat{j}_i = B_i \cdot (\hat{j}_i - 1) + \sum_{k=i}^{t-1} (\hat{j}_{k+1} - \hat{j}_k) \cdot B_k.$$

Algorithm 9: APPPROFILESIM(G,ε,d,W)

get sequence $\{\hat{j}_1,\ldots,\hat{j}_t\}$ from APPCOSTSIM(G,ε,d,W)
set t according to Definition 7.8
set $\overline{\text{cost}}_{n-B_t}=0$
for $i=\{1,\ldots,t-1\}$ **do**
    set B_i according to Definition 7.8
    $\overline{\text{cost}^{(s)}}_{n-B_i}=B_i\cdot(\hat{j}_i-1)+\sum_{k=i}^{t-1}(\hat{j}_{k+1}-\hat{j}_k)\cdot B_k$
Output estimated vector $(\overline{\text{cost}^{(s)}}_{n-B_1},\ldots,\overline{\text{cost}^{(s)}}_{n-B_t})$

In the last equation, we have $\hat{j}_i-1$ because $\overline{D}_{\hat{j}_i}$ is considered in $\hat{J}_i$, which should be excluded in the summation $\sum_{j=\hat{j}_i+1}^{W}\overline{D}_j$. Once we have such a succinct representation, we can construct an oracle, that given any $k\in[1,n]$, and $n-B_i\le k<n-B_{i+1}$, we let $\widehat{\text{cost}^{(s)}}_k=\overline{\text{cost}^{(s)}}_{B_i}$.

Algorithm 10: PROFILEORACLESIM$(k,\{B_1,\ldots,B_t\},(\overline{\text{cost}^{(s)}}_{n-B_1},\ldots,\overline{\text{cost}^{(s)}}_{n-B_t}))$

define $B_{t+1}=-\infty$
use binary search over $(B_1,\ldots,B_t)$, and find the index i such that $n-B_i\le k<n-B_{i+1}$
output $\widehat{\text{cost}^{(s)}}_k:=\overline{\text{cost}^{(s)}}_{n-B_i}$

We call the vector $(\overline{\text{cost}^{(s)}}_{n-B_1},\ldots,\overline{\text{cost}^{(s)}}_{n-B_t})$ a *succinct representation* of the vector $(\widehat{\text{cost}^{(s)}}_1,\ldots,\widehat{\text{cost}^{(s)}}_n)$, and we call Algorithm 10 a *profile oracle*, as it can take as input any index k and answer $\widehat{\text{cost}^{(s)}}_k$.

Analysis of Algorithm 9 and Algorithm 10 Now we analyze Algorithm 9 and Algorithm 10. We will show that the vector $(\widehat{\text{cost}^{(s)}}_1,\ldots,\widehat{\text{cost}^{(s)}}_n)$ is a good approximation of the profile vector $(\text{cost}_1^{(s)},\ldots,\text{cost}_n^{(s)})$. We first prove an error bound for each individual $\widehat{\text{cost}^{(s)}}_k$, and then bound the sum of the error and give the proof of Theorem 1.5.

Fact 7.13. *For any* $2\le i\le t_1+2t_2-1$, $B_{i-1}-B_{i+1}\le 4\varepsilon B_{i+1}$.

Proof. We develop the following proof based on Definition 7.8 and Fact 7.9.

- When $2\le i\le t_1-1$,

$$B_{i-1}-B_{i+1}=(n-(i-1)\frac{\varepsilon n}{W})-(n-(i+1)\frac{\varepsilon n}{W})=2\varepsilon\frac{n}{W},$$

  Since $B_{i+1}\ge B_{t_1}\ge n-\frac{n}{W}$, and when $W\ge 2$, $n-\frac{n}{W}\ge\frac{n}{W}$, thus $B_{i-1}-B_{i+1}\le 2\varepsilon(n-\frac{n}{W})\le 2\varepsilon B_{i+1}$.

- When $i=t_1$, $B_{i-1}-B_{i+1}=B_{t_1-1}-B_{t_1+1}=B_{t_1}+\frac{\varepsilon n}{W}-B_{t_1+1}$. From Fact 7.9, $B_{t_1}<n-\frac{n}{W}(1-\varepsilon)$. Then,

$$B_{i-1}-B_{i+1}=B_{t_1}+\frac{\varepsilon n}{W}-B_{t_1+1}<(n-\frac{n}{W}(1-\varepsilon))+\frac{\varepsilon n}{W}-(n-(1+\varepsilon)\frac{n}{W})=3\frac{\varepsilon n}{W},$$

  Since $B_{t_1+1}\ge B_{t_1+2t_2}\ge\frac{n}{W}$, we have $B_{t_1-1}-B_{t_1+1}\le 3\varepsilon B_{t_1+1}$.

- When $i = t_1 + 1$, $B_{i-1} - B_{i+1} = B_{t_1} - B_{t_1+2} < (n - \frac{n}{W}(1-\varepsilon)) - (n - (1+\varepsilon)^2 \frac{n}{W}) = (3\varepsilon + \varepsilon^2)\frac{n}{W} < 4\varepsilon \frac{n}{W}$, as $B_{t_1} < n - \frac{n}{W}(1-\varepsilon)$ and $\varepsilon < 1$. Since $B_{t_1+2} \geq B_{t_1+2t_2} \geq \frac{n}{W}$, we have $B_{t_1} - B_{t_1+2} \leq 4\varepsilon B_{t_1+2}$.

- When $t_1 + 2 \leq i \leq t_1 + t_2 - 1$,

$$B_{i-1} - B_{i+1} = (n - (1+\varepsilon)^{i-1-t_1}\frac{n}{W}) - (n - (1+\varepsilon)^{i+1-t_1}\frac{n}{W}) = ((1+\varepsilon)^2 - 1) \cdot (1+\varepsilon)^{i-1-t_1}\frac{n}{W},$$

  Since $\varepsilon < 1$ and $B_{i-1} > B_{i+1}$, we have $B_{i-1} - B_{i+1} \leq (2\varepsilon + \varepsilon^2)(n - B_{i-1}) < 3\varepsilon(n - B_{i+1})$. As $B_{i+1} \geq B_{t_1+t_2} \geq \frac{n}{2}$, $B_{i-1} - B_{i+1} \leq 3\varepsilon B_{i+1}$.

- When $i = t_1 + t_2$, $B_{i-1} - B_{i+1} = B_{t_1+t_2-1} - B_{t_1+t_2+1}$. By Definition 7.8, $n - B_{t_1+t_2-1} = (1+\varepsilon)(n - B_{t_1+t_2})$. Besides, from Fact 7.9, $n - B_{t_1+t_2} > \frac{n}{2(1+\varepsilon)}$. Then we have $B_{t_1+t_2-1} < n - (1+\varepsilon)\frac{n}{2(1+\varepsilon)} = \frac{n}{2}$. Then, $B_{t_1+t_2-1} - B_{t_1+t_2+1} < \frac{n}{2} - \frac{n}{2(1+\varepsilon)} = \varepsilon \frac{n}{2(1+\varepsilon)} = \varepsilon B_{t_1+t_2+1}$. Thus, $B_{i-1} - B_{i+1} < \varepsilon B_{i+1}$.

- When $i = t_1 + t_2 + 1$, since $n - B_{t_1+t_2} > \frac{n}{2(1+\varepsilon)}$ and $\varepsilon < 1$, we have

$$B_{i-1} - B_{i+1} = B_{t_1+t_2} - B_{t_1+t_2+2} < (n - \frac{n}{2(1+\varepsilon)}) - \frac{n}{2(1+\varepsilon)^2} = (3\varepsilon + \varepsilon^2)\frac{n}{2(1+\varepsilon)^2} < 4\varepsilon B_{t_1+t_2+2} = 4\varepsilon B_{i+1}.$$

- When $t_1 + t_2 + 2 \leq i \leq t_1 + 2t_2 - 1$,

$$B_{i-1} - B_{i+1} = \frac{n}{2(1+\varepsilon)^{i-1-(t_1+t_2)}} - \frac{n}{2(1+\varepsilon)^{i+1-(t_1+t_2)}} = ((1+\varepsilon)^2 - 1)B_{i+1} = (2\varepsilon + \varepsilon^2)B_{i+1} \leq 3\varepsilon B_{i+1}.$$

In a conclusion, for any $2 \leq i \leq t_1 + 2t_2 - 1$, $B_{i-1} - B_{i+1} \leq 4\varepsilon B_{i+1}$. □

The following lemma provides an error bound for each individual $\widehat{\text{cost}^{(s)}}_k$.

Lemma 7.14. *Assume that $W \leq n$ and let $\varepsilon < 1$. With probability at least $3/4$, for any integer $1 \leq k \leq n-1$, Algorithm 10 returns an estimate $\widehat{\text{cost}^{(s)}}_k$ for the k-clustering cost in similarity case, i.e., $\text{cost}^{(s)}{}_k$, such that*

$$|\widehat{\text{cost}^{(s)}}_k - \text{cost}^{(s)}{}_k| \leq 30\varepsilon \cdot \max\{\text{cost}^{(s)}_k, n\}$$

Given the succinct representation $(\overline{\text{cost}^{(s)}}_{B_1}, \ldots, \overline{\text{cost}^{(s)}}_{B_t})$, the running time of the algorithm is $O(\log(\frac{\log W}{\varepsilon}))$.

Proof. From Lemma 7.11, we have $\left|\overline{D}_j - D_j\right| \leq 5\varepsilon \cdot \max\left\{\frac{n}{k}, \min\{D_j, n - D_j\}\right\}$ for all $j \in [W]$, with probability at least $\frac{7}{8}$. We assume that this event holds in the remaining part of the proof.

For simplicity of notation, we use j_i to denote $\hat{j}_i$ in the following of the proof. By definition of $\overline{\text{cost}^{(s)}}_{n-B_i}$ and $\text{cost}^{(s)}_{c_{j_i}}(G)$, the gap between them is,

$$\begin{aligned}
|\overline{\text{cost}^{(s)}}_{n-B_i} - \text{cost}^{(s)}_{c_{j_i}}| &= \left|(\sum_{j=j_i+1}^{W} \overline{D}_j + \overline{D}_{j_i} \cdot j_i) - (\sum_{j=j_i+1}^{W} D_j + D_{j_i} \cdot j_i)\right| \\
&\leq \sum_{j=j_i+1}^{W} |\overline{D}_j - D_j| + |\overline{D}_{j_i} - D_{j_i}| \cdot j_i \\
&\leq 5\varepsilon \sum_{j=j_i+1}^{W} D_j + 5\varepsilon \cdot D_{j_i} \cdot j_i
\end{aligned}$$

$$= 5\varepsilon \cdot \mathrm{cost}^{(s)}_{c_{j_i}}$$

For $n - B_i \le k < n - B_{i+1}$, where $1 \le i \le t-1$, we estimate cost_k using $\overline{\mathrm{cost}^{(s)}}_{n-B_i}$, and thus we have

$$\begin{aligned}
|\widehat{\mathrm{cost}^{(s)}}_k - \mathrm{cost}^{(s)}_k| &\le |\overline{\mathrm{cost}^{(s)}}_{n-B_i} - \mathrm{cost}^{(s)}_{c_{j_i}}| + |\mathrm{cost}^{(s)}_{c_{j_i}} - \mathrm{cost}^{(s)}_k| \\
&\le 5\varepsilon \cdot \mathrm{cost}^{(s)}_{c_{j_i}} + |\mathrm{cost}^{(s)}_{c_{j_i}} - \mathrm{cost}^{(s)}_k| \\
&\le 5\varepsilon(\mathrm{cost}^{(s)}_k + |\mathrm{cost}^{(s)}_{c_{j_i}} - \mathrm{cost}^{(s)}_k|) + |\mathrm{cost}^{(s)}_{c_{j_i}} - \mathrm{cost}^{(s)}_k| \\
&\le 5\varepsilon \cdot \mathrm{cost}^{(s)}_k + 6|\mathrm{cost}^{(s)}_{c_{j_i}} - \mathrm{cost}^{(s)}_k| \qquad \text{(since } \varepsilon < 1\text{)}
\end{aligned}$$

Thus, we only need to bound $|\mathrm{cost}^{(s)}_{c_{j_i}} - \mathrm{cost}^{(s)}_k|$.
Case (I): $c^{(s)}_{j_i} \le k$. In this case, $\mathrm{cost}^{(s)}_k \le \mathrm{cost}^{(s)}_{c_{j_i}}$ and $j_i \le w_{n-k}$. Since $\mathrm{cost}^{(s)}_k = \sum_{j=1}^{n-k} w^{(s)}_j$,

$$|\mathrm{cost}^{(s)}_{c_{j_i}} - \mathrm{cost}^{(s)}_k| = \mathrm{cost}^{(s)}_{c_{j_i}} - \mathrm{cost}^{(s)}_k = \sum_{j=n-k+1}^{n-c^{(s)}_{j_i}} w^{(s)}_j \le (k - c^{(s)}_{j_i}) \cdot w^{(s)}_{n-k}$$

From Lemma 5.4, when $i \le t-2$, $B_{i+2} \le D_{j_i} = n - c^{(s)}_{j_i} \le B_{i-1}$, then $n - B_{i-1} \le c^{(s)}_{j_i} \le n - B_{i+2}$, and thus $k - c^{(s)}_{j_i} \le B_{i-1} - B_{i+1}$. When $2 \le i \le t_1 + 2t_2 - 1$, according to Fact 7.13, $B_{i-1} - B_{i+1} \le 4\varepsilon B_{i+1} \le 4\varepsilon(n-k)$; when $t_1 + 2t_2 \le i \le t-1$, according to Definition 7.8, $B_{i-1} - B_{i+1} = 2\varepsilon\frac{n}{W} > 2\varepsilon(n-k)$. Thus, $|\mathrm{cost}^{(s)}_{c_{j_i}} - \mathrm{cost}^{(s)}_k| \le 4\varepsilon \cdot \max\{n-k, \frac{n}{W}\} \cdot w^{(s)}_{n-k}$. Since $\mathrm{cost}^{(s)}_k = \sum_{j=w^{(s)}_{n-k}+1}^{W} (n - c^{(s)}_j) + (n-k) \cdot w^{(s)}_{n-k} \ge (n-k) \cdot w^{(s)}_{n-k}$ and $w^{(s)}_{n-k} \le n$, we have,

$$|\widehat{\mathrm{cost}^{(s)}}_k - \mathrm{cost}^{(s)}_k| \le 5\varepsilon \cdot \mathrm{cost}^{(s)}_k + 6 \cdot 4\varepsilon \cdot \max\{n-k, \frac{n}{W}\} \cdot w^{(s)}_{n-k} \le 30\varepsilon \cdot \max\{\mathrm{cost}^{(s)}_k, n\}$$

Case(II): $k \le c^{(s)}_{j_i}$. In this case, $\mathrm{cost}^{(s)}_{c_{j_i}} \le \mathrm{cost}^{(s)}_k$ and $w_{n-k} \le j_i$. Then,

$$|\mathrm{cost}^{(s)}_{c_{j_i}} - \mathrm{cost}^{(s)}_k| = \mathrm{cost}^{(s)}_k - \mathrm{cost}^{(s)}_{c_{j_i}} = \sum_{j=n-c^{(s)}_{j_i}+1}^{n-k} w^{(s)}_j \le (c^{(s)}_{j_i} - k) \cdot j_i \le (B_i - B_{i+2}) \cdot j_i$$

When $1 \le i \le t_1 + 2t_2 - 2$, according to Fact 7.13, $B_i - B_{i+2} \le 4\varepsilon B_{i+2} < 4\varepsilon B_{i-1} \le 4\varepsilon(n - c^{(s)}_{j_i})$; when $t_1 + 2t_2 \le i \le t-2$, according to Definition 7.8, $B_i - B_{i+2} = 2\varepsilon\frac{n}{W} > 2\varepsilon(n - c^{(s)}_{j_i})$. From Lemma 7.12, we know that $\mathrm{cost}^{(s)}_{c_{j_i}} \ge (n - c^{(s)}_{j_i})$, and thus

$$|\widehat{\mathrm{cost}^{(s)}}_k - \mathrm{cost}^{(s)}_k| \le 5\varepsilon \cdot \mathrm{cost}^{(s)}_k + 6 \cdot 4\varepsilon \cdot \max\{n - c^{(s)}_{j_i}, \frac{n}{W}\} \cdot j_i \le 30\varepsilon \cdot \max\{\mathrm{cost}^{(s)}_k, n\}$$

In a conclusion, we have $|\mathrm{cost}^{(s)}_k - \widehat{\mathrm{cost}^{(s)}}_k| \le 30\varepsilon \cdot \max\{\mathrm{cost}^{(s)}_k, n\}$.

Running time analysis. Given the succinct representation, Algorithm 10 can be implemented in $O(\log t) = O(\log(\frac{\log W}{\varepsilon}))$ time for any k, as one can simply perform binary search to find the right index i with $n - B_i \le k < n - B_{i+1}$. □

With Lemma 7.14 established, we are ready to bound $\sum_{k=1}^{n} |\widehat{\text{cost}^{(s)}}_k - \text{cost}^{(s)}_k|$.

Proof of Theorem 1.5. From the error bound for individual $\widehat{\text{cost}^{(s)}}_k$ in Lemma 7.14, we have that

$$\sum_{k=1}^{n} |\widehat{\text{cost}^{(s)}}_k - \text{cost}^{(s)}_k| \leq \sum_{k=1}^{n} 30\varepsilon \cdot \max\{\text{cost}^{(s)}_k, n\} \leq 30\varepsilon \cdot \max\left\{\sum_{k=1}^{n} \text{cost}^{(s)}_k, n^2\right\}$$

According to Fact 7.2, $n^2 \leq 4\text{cost}^{(s)}(G)$, and thus $\sum_{k=1}^{n} |\widehat{\text{cost}^{(s)}}_k - \text{cost}^{(s)}_k| \leq 120\varepsilon \cdot \text{cost}^{(s)}(G)$.

Replacing ε with $\varepsilon/120$, we get an succinct representation of $(\widehat{\text{cost}^{(s)}}_1, \ldots, \widehat{\text{cost}^{(s)}}_n)$ such that each $\widehat{\text{cost}^{(s)}}_k$ is a $(1+\varepsilon)$-estimator *on average*.

Running time analysis. Note that we first invoke Algorithm 8 to obtain the sequence $\{j_0, j_1, \ldots, j_t\}$, which takes $O(\frac{Wd}{\varepsilon^3}\log^4(\frac{Wd}{\varepsilon}))$ time. Given the sequence, to estimate each $\overline{\text{cost}^{(s)}}_{n-B_i}$, we need to calculate $\sum_{k=i}^{t-1}(j_{k+1}-j_k)\cdot B_k$ in $O(t)$ time, for all $i \in [1,t]$. Thus, getting the estimated vector $(\overline{\text{cost}^{(s)}}_{n-B_1}, \ldots, \overline{\text{cost}^{(s)}}_{n-B_t})$ can be done in $O(t^2) = O(\log^2 W/\varepsilon^2)$ time. Thus, the total running time of Algorithm 9 is $O(\frac{Wd}{\varepsilon^3}\log^4(\frac{Wd}{\varepsilon})) = \tilde{O}(\frac{W}{\varepsilon^3}d)$. □

8 Lower Bounds

In this section, we give the proofs of the lower bounds on the query complexities for estimating the SLC costs $\text{cost}(G)$ in the distance case and $\text{cost}^{(s)}(G)$ in the similarity case. The results hold for the model of computation we introduced earlier, which requires $\Theta(i)$ time to access the i-th neighbor in the adjacency list, as well as for a model where we can query for the degree of a vertex and for its i-th neighbor in $O(1)$ time. The proofs built upon the query lower bounds for estimating the number of connected components and the weight of the minimum spanning tree in [CRT05] and can be seen as an adaptation of their proof to our setting (that is, we need to adjust them at various places to our objective function).

We first introduce a useful tool. For any $0 < q \leq 1/2$ and $t = 0, 1$, let $\mathscr{D}_q^t$ denote the distribution over $\{0,1\}$ that assigns 1 with probability $q_t = q(1+(-1)^t\varepsilon)$. We define a distribution $\mathscr{D}$ on n-bit strings as follows: (1) choose $t = 1$ with probability $1/2$ and $t = 0$ otherwise; (2) generate a random string b from $\{0,1\}^n$ by independently selecting each bit b_i from the distribution $\mathscr{D}_q^t$. The following result is shown in [CRT05].

Lemma 8.1 (Lemma 9 in [CRT05])**.** *Given query access to a random bit string generated from the above distribution $\mathscr{D}$, any algorithm that determines the value of t with a success probability of at least $3/4$ requires $\Omega(\varepsilon^{-2}/q)$ bit queries on average.*

Now we use the above lemma to prove our first lower bound.

Theorem 1.3. *Let $\frac{W^{1/4}}{\sqrt{40n}} < \varepsilon < \frac{1}{2}$ and $W > 1$. Any algorithm that $(1+\varepsilon)$-approximates the cost of SLC cost $\text{cost}(G)$ in the distance graph with success probability at least $3/4$ needs to make $\Omega(d\sqrt{W}/\varepsilon^2)$ queries.*

Proof. The proof follows closely a corresponding proof from [CRT05] while adapting it to our setting. We first define two families of weighted graphs, $\mathscr{P}_0$ and $\mathscr{P}_1$. Each graph in $\mathscr{P}_i$ (where $i = 0, 1$) is a path consisting of n vertices. In the family $\mathscr{P}_0$, we generate a random $(n-1)$-bit string $b_1 \ldots b_{n-1}$, with each bit independently drawn from $\mathscr{D}_q^0$, where $q = 1/\sqrt{W-1}$. We assign a weight of W (or 1) to the i-th edge along

the path if $b_i = 1$ (or 0, respectively). The construction for the family $\mathscr{P}_1$ is similar, except that each bit is drawn from $\mathscr{D}_q^1$.

Consider a graph G from $\mathscr{P}_0 \cup \mathscr{P}_1$. Let T_W be the number of edges of weight W in G. We note that the SLC cost of G in distance case is

$$\begin{aligned}
\mathrm{cost}(G) &= \sum_{i=1}^{n-1}(n-i)\cdot w_i = \sum_{i=1}^{n-1-T_W}(n-i)\cdot 1 + \sum_{i=n-T_W}^{n-1}(n-i)\cdot W = \sum_{i=T_W+1}^{n-1} i + \sum_{i=1}^{T_W} i\cdot W \\
&= \frac{(T_W+1+n-1)(n-1-T_W)}{2} + W\cdot\frac{(T_W+1)T_W}{2} \\
&= \frac{n(n-1)-T_W-T_W^2+W\cdot T_W^2+W\cdot T_W}{2} \\
&= \frac{n(n-1)+(W-1)(T_W^2+T_W)}{2}
\end{aligned}$$

Now observe that $\mathbf{E}[T_W] = \frac{n-1}{\sqrt{W-1}}(1+\varepsilon)$ or $\mathbf{E}[T_W] = \frac{n-1}{\sqrt{W-1}}(1-\varepsilon)$, depending on if $G \in \mathscr{P}_0$ or $G \in \mathscr{P}_1$. It further holds that $\mathbf{E}[\mathrm{cost}(G)] = \Theta(n^2 + (W-1)\cdot\frac{(n-1)^2}{W-1}) = \Theta(n^2)$.

Case I. If $G \in \mathscr{P}_0$, $\mathbf{E}[T_W] = \frac{n-1}{\sqrt{W-1}}(1+\varepsilon)$, then By Chernoff Bound in Theorem 2.1,

$$\Pr[T_W < (1-\frac{\varepsilon}{2})\,\mathbf{E}[T_W]] \le \mathrm{e}^{-\frac{\varepsilon^2(1+\varepsilon)(n-1)}{8\sqrt{W-1}}} \le \mathrm{e}^{-\frac{\varepsilon^2 n}{8\sqrt{W}}}.$$

Since $\varepsilon \le \frac{1}{2}$, it holds that with probability $1-\mathrm{e}^{-\varepsilon^2 n/8\sqrt{W}}$, $T_W \ge \frac{n-1}{\sqrt{W-1}}(1+\varepsilon)(1-\frac{\varepsilon}{2}) \ge \frac{n-1}{\sqrt{W-1}}(1+\frac{\varepsilon}{4})$. Now as $\varepsilon > \frac{W^{1/4}}{\sqrt{40n}}$, we have that with probability at least $1-\mathrm{e}^{-\frac{40}{24}} \ge 0.8$,

$$\mathrm{cost}(G) \ge \frac{n(n-1)}{2} + \frac{(n-1)^2}{2}(1+\frac{\varepsilon}{4})^2 + \frac{(n-1)\sqrt{W-1}}{2}(1+\frac{\varepsilon}{4}).$$

Case II. If $G \in \mathscr{P}_1$, $\mathbf{E}[T_W] = \frac{n-1}{\sqrt{W-1}}(1-\varepsilon)$ and then

$$\Pr[T_W > (1+\frac{\varepsilon}{2})\,\mathbf{E}[T_W]] \le \mathrm{e}^{-\frac{\varepsilon^2(1-\varepsilon)(n-1)}{12\sqrt{W-1}}} \le \mathrm{e}^{-\frac{\varepsilon^2 n}{24\sqrt{W}}}.$$

Similar as above, with probability at least $1-\mathrm{e}^{-\varepsilon^2 n/24\sqrt{W}}$, it holds that $T_W \le \frac{n-1}{\sqrt{W-1}}(1-\varepsilon)(1+\frac{\varepsilon}{2}) \le \frac{n-1}{\sqrt{W-1}}(1-\frac{\varepsilon}{2})$. By the fact that $\varepsilon > \frac{W^{1/4}}{\sqrt{40n}}$, with probability at least $1-\mathrm{e}^{-\frac{40}{24}} \ge 0.8$, it holds that

$$\mathrm{cost}(G) \le \frac{n(n-1)}{2} + \frac{(n-1)^2}{2}(1-\frac{\varepsilon}{2})^2 + \frac{(n-1)\sqrt{W-1}}{2}(1-\frac{\varepsilon}{2}).$$

Since $\varepsilon \le \frac{1}{2}$, $2-\frac{\varepsilon}{4} \ge \frac{15}{8} \ge \frac{4}{3}$. So the gap between the above two bounds is

$$\frac{(n-1)^2}{2}(2-\frac{\varepsilon}{4})\frac{3\varepsilon}{4} + \frac{(n-1)\sqrt{W-1}}{2}\frac{3\varepsilon}{4} \ge \varepsilon\frac{(n-1)^2}{2}.$$

Thus, any algorithm that estimates $\mathrm{cost}(G)$ with a relative error of ε/C for some large constant C, can be used to distinguish if $G \in \mathscr{P}_0$ or $G \in \mathscr{P}_1$. By Lemma 8.1, any algorithm that solves the latter problem requires $\Omega(\sqrt{W}/\varepsilon^2)$ queries on average.

Now we extend our result to the case of graphs with arbitrary average degree $d \geq 2 - 2/n$. Let $d = 2m/n$, i.e. our graph is supposed to have m edges. In order to obtain a graph with average degree d we add edges to the vertices of our path in such a way that every vertex has degree $\lceil d \rceil$ or $\lfloor d \rfloor$ (this can be done by greedily adding edges between the two vertices with smallest degree). All edges that have been newly added receive a weight of W. We observe that this does not change the weight of the minimum spanning tree and that this weight is still given by the weight of the edges of the initial path. For each vertex v of degree d_v we then choose a random permutation of d_v elements uniformly at random and put the adjacency list in the corresponding order. Let us call the resulting graph H and the initial path G.

We will argue that if an algorithm makes q queries to H then in expectation it queries $O(q/d)$ edges from G. Observe that, conditioned on an arbitrary history of queries and answers, the probability that a query to a previously unqueried neighbor of vertex v yields an edge from G is at most $2/i$, where i denotes the number of unqueried neighbors of v.

To simplify the analysis, we count only the first $\lfloor d/2 \rfloor$ queries to a vertex; any remaining edges are revealed for free. The cost for revealing these additional edges can be charged to the first $\lfloor d/2 \rfloor$ queries. As a result, any vertex with unknown neighbors is still in the phase where queries are being charged, and thus must have at least $\lfloor d/2 \rfloor$ unknown neighbors. Therefore, conditioned on the query history, the probability that a query reveals an edge from the original path G is at most $2/\lfloor d/2 \rfloor$. By linearity of expectation, the expected number of such queried edges is at most $2q/\lfloor d/2 \rfloor + R$, where $R = O(q/d)$ is the number of edges revealed for free. Thus, the expected number of revealed edges is at most $C'q/d$ for some constant $C' > 0$.

Now consider an arbitrary algorithm that computes a $(1+\varepsilon)$-approximation of $\mathrm{cost}(G)$ using q queries in expectation. Then this algorithm makes at most $C'q/d$ queries to the path on average. We can then use this algorithm to solve the problem on G using $C'q/d$ queries on average (by first building H as described above and then querying G whenever we query an edge of the path in H). However, we know that the latter problem requires $\Omega(\sqrt{W}/\varepsilon^2)$ queries on average as proven above. Thus, the algorithm has to make $\Omega(d\sqrt{W}/\varepsilon^2)$ queries on H. □

Now we consider the lower bound for the similarity case and prove the following theorem.

Theorem 1.6. *Let $\sqrt{\frac{W}{40n}} < \varepsilon < \frac{1}{2}$ and $W > 10$. Any algorithm that $(1+\varepsilon)$-approximates the SLC cost $\mathrm{cost}^{(s)}(G)$ in the similarity graph with success probability at least $3/4$ needs to make $\Omega(dW/\varepsilon^2)$ queries.*

Proof. We construct two families of graphs, $\mathscr{P}_0$ and $\mathscr{P}_1$, in the same manner as described in the proof of Theorem 1.3, with the key difference that we now set $q = \frac{1}{W-1}$.

Now suppose that $G \in \mathscr{P}_0 \cup \mathscr{P}_1$. Let T_W be the number of edges of weight W in G. We note that the SLC cost of the similarity graph G is

$$\begin{aligned}
\mathrm{cost}^{(s)}(G) &= (n-1)W + (n-2)W + \cdots + (n-T_W)W + (n-T_W-1)\cdot 1 + \cdots + 1 \cdot 1 \\
&= \frac{(n-T_W)(n-T_W-1)}{2} + W \cdot \frac{(2n-T_W-1)T_W}{2} \\
&= \frac{n(n-1) + (2n-1-T_W)T_W(W-1)}{2} \\
&= \frac{n(n-1)}{2} - \frac{W-1}{2}T_W^2 + \frac{(W-1)(2n-1)}{2}T_W
\end{aligned}$$

Now since $q = \frac{1}{W-1}$, similar to the proof of Theorem 1.3, we have the following two cases.

Case I. If $G_0 \in \mathscr{P}_0$, $\mathbf{E}[T_W] = \frac{n-1}{W-1}(1+\varepsilon)$. Then by Chernoff bound and the fact that $\varepsilon > \sqrt{\frac{W}{40n}}$, with probability at least $1 - e^{-\varepsilon^2 n/24W} \geq 1 - e^{-40/24} \geq 0.8$,

$$\frac{n-1}{W-1}(1+\varepsilon)(1-\frac{\varepsilon}{2}) \leq T_W \leq \frac{n-1}{W-1}(1+\varepsilon)(1+\frac{\varepsilon}{2}).$$

This further gives that

$$\text{cost}^{(s)}(G) \geq \frac{n(n-1)}{2} - \frac{W-1}{2} \cdot [\frac{n-1}{W-1}(1+\varepsilon)(1+\frac{\varepsilon}{2})]^2 + \frac{(W-1)(2n-1)}{2} \cdot [\frac{n-1}{W-1}(1+\varepsilon)(1-\frac{\varepsilon}{2})].$$

Case II. If $G \in \mathscr{P}_1$, $\mathbf{E}[T_W] = \frac{n-1}{W-1}(1-\varepsilon)$. Then by Chernoff bound and the fact that $\varepsilon > \sqrt{\frac{W}{40n}}$, with probability at least $1 - e^{-\varepsilon^2 n/24W} \geq 1 - e^{-40/24} \geq 0.8$, it holds that for $G \in \mathscr{P}_1$,

$$\frac{n-1}{W-1}(1-\varepsilon)(1-\frac{\varepsilon}{2}) \leq T_W \leq \frac{n-1}{W-1}(1-\varepsilon)(1+\frac{\varepsilon}{2}).$$

This further gives that

$$\text{cost}^{(s)}(G) \leq \frac{n(n-1)}{2} - \frac{W-1}{2} \cdot [\frac{n-1}{W-1}(1-\varepsilon)(1-\frac{\varepsilon}{2})]^2 + \frac{(W-1)(2n-1)}{2} \cdot [\frac{n-1}{W-1}(1-\varepsilon)(1+\frac{\varepsilon}{2})].$$

Thus, when $\varepsilon < \frac{1}{2}$, the gap between the above two bounds is at least

$$-\frac{(n-1)^2}{2(W-1)}(2+\varepsilon^2)3\varepsilon + \frac{(2n-1)(n-1)}{2}\varepsilon \geq \varepsilon(n-1)\left(\frac{2n-1}{2} - \frac{3}{2}\cdot\frac{9}{4}\cdot\frac{n-1}{W-1}\right) \geq \frac{1}{4}\varepsilon n^2,$$

where the last inequality follows from our assumption that $W > 10$.

Thus, any algorithm that estimates the cost $\text{cost}^{(s)}(G)$ with a relative error of ε/C for some large constant C, can be used to distinguish if $G \in \mathscr{P}_0$ or $G \in \mathscr{P}_1$. By Lemma 8.1, any algorithm that solves the latter problem requires $\Omega(W/\varepsilon^2)$ queries into G on average.

To extend our proof to the case of average degree d we proceed in the same way as in the previous proof (except that the weight of the new edges will be 1 instead of W). □

9 Experiments and Evaluation

To evaluate the performance of our algorithms, we conducted experiments on real-world datasets. All experiments were implemented in C++, using an Intel(R) Xeon(R) Platinum 8358 Processor @ 2.60 GHZ, with 504 GB RAM. We use Kruskal's algorithm for minimum/maximum spanning tree computation as our baseline, implemented in C++ boost library [BD98][2].

Dataset Preprocessing Under our assumption of connectivity in graphs, we preprocessed the datasets to find the largest connected components as input of the algorithm. The information of preprocessed datasets is detailed in Table 1.

For datasets in distance setting (localization based datasets and road networks), we assign the distance of two neighboring vertices to the edge weight. For datasets in similarity setting (Spotify co-listening graph, and co-citation graphs), we assign the number of collaborations (or the number of times two songs are co-listened) to the edge weight between two neighboring vertices.

[2]Our source code can be accessed in: https://anonymous.4open.science/r/sublinear-clustering

Table 1: All of the datasets used: for distance case, we have road network for different countries, and friendship networks with location (loc-brightkite/gowalla); For similarity case, we have co-authorship datasets on different fields, and a co-listened dataset for an music application called Spotify. Each dataset listed in the table has been preprocessed to extract the largest connected component from the original graph. The parameters n, m, and W represent the number of vertices, the number of edges, and the largest weight value, respectively, in these preprocessed graphs.

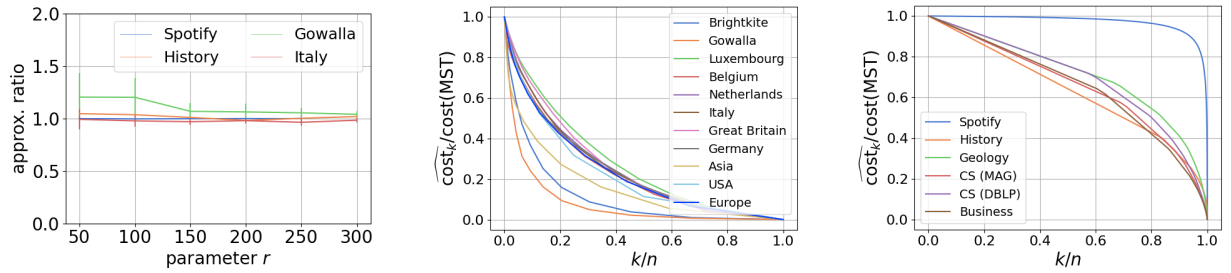
Name of dataset	n	m	W
Dataset in distance case			
Location-Brightkite [CML11]	49,011	386,716	19,985
Location-Gowalla [CML11]	96,953	910,052	19,883
Luxembourg road network [DIM11]	114,599	119,666	2,065
Belgium road network [DIM11]	1,441,295	1,549,970	6,408
Netherland road network [DIM11]	2,216,688	2,441,238	7,027
Italy road network [DIM11]	6,686,493	7,013,978	9,719
Great-Britain road network [DIM11]	7,733,822	8,156,517	10,520
Germany road network [DIM11]	11,548,845	12,369,181	15,337
Asia road network [DIM11]	11,950,757	12,711,603	87,377
USA road network [DIM10]	23,947,347	58,333,344	24,394
Europe road network [DIM11]	50,912,018	54,054,660	368,855
Dataset in similarity case			
Co-authorship Business [BKT18, BAS$^+$18, SSS$^+$15]	40,383	59,513	63
Co-authorship CS (MAG) [AVB20, SSS$^+$15]	59,709	415,430	36
Co-authorship History [BAS$^+$18, SSS$^+$15]	219,435	1,614,992	606
Co-authorship Geology [BAS$^+$18, SSS$^+$15]	898,648	9,782,224	192
Co-authorship CS (DBLP) [BAS$^+$18]	1,431,475	7,886,713	121
Co-listen Spotify [KLCB20]	3,061,417	85,467,545	99,536

Implementation Changes to Our Algorithm In the theoretical part, we assumed that the average degree d is given, and set $d^{(G)}$ to be $d \cdot \Gamma$ as the threshold degree. To be able to deal with a setting when the average degree is not given, our algorithms more general, we sample Γ vertices in the experiments, and let $d^{(G)}$ be the maximum degree among these sampled vertices, similar as in [CRT05].

We pick different constants from theory when choosing parameters. Because we are always assuming the worst case in theory, and try to bound the performance of each run; while in the real world, we can get good results with much smaller constants, and we can also refine the result by calculating the average cost among multiple runs, so we don't need to bound the performance of each run. In the implementation, the algorithm is given parameter r as sample size, and parameter Γ is set accordingly: $\Gamma = 1 \cdot k\sqrt{\frac{r}{k}} = \sqrt{rk}$, as $r = O(k/\varepsilon^2)$ and $\Gamma = O(k/\varepsilon)$, where k's value depends on different measurement settings.

To evaluate the robustness of our algorithms, we computed the deviation among 30 experiments by the standard error of them: $std\_err = \sqrt{\frac{\sum_{i=1}^{30}|x_i - avg|^2}{30}}$, where x_i is the estimated cost(G) value returned from the algorithm.

Experimental Results Fig. 1a shows the accuracy of our algorithm to approximate clustering cost cost(G) and cost$^{(s)}(G)$, among both distance and similarity datasets. Our algorithm already has a very good approx-

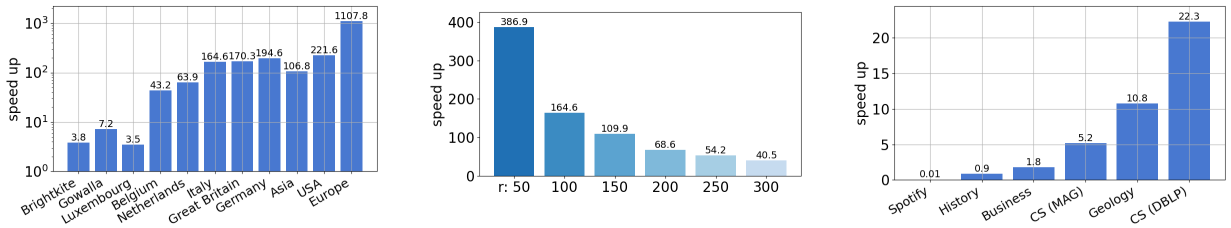


(a) Approx. ratio for distance (Gowalla and Italy) and similarity (Spotify and History) datasets

(b) Profiles for distance datasets

(c) Profile for similarity datasets

Figure 1: Approximation ratio and normalized profiles



(a) Distance with $r = 100$

(b) Speed up on 'Italy' dataset, with various r

(c) Similarity with $r = 100$

Figure 2: Datasets speed up

imation ratio, even when r is small; and accuracy becomes better as r increases, as we have finer intervals and larger sample size. For the sake of space, we put more experiments in appendix. For the bias where the approximation ratio begins at a value higher than 1.0 when r is small, it is likely from the algorithm overlooking some large connected components, when the number of sampled vertices is small. However, our results verified that the bias diminishes as r increases.

For most of the datasets, our running time beats the baseline, Kruskal's MST algorithm, shown in Fig. 2. We remark that when the dataset contains relatively small number of vertices and edges, then it is best to simply run Kruskal's MST algorithm to compute cost(G). Our speedup on dataset Brightkite, Gowalla, and Luxembourg are lower, because they are small datasets. As Spotify has a relatively large value of W, and since our running time is linear in W, it is more efficient to simply run an MST algorithm.

Fig. 1b and Fig. 1c shows the hierarchical clustering structures. We normalized the profile vectors by computing the fraction of the clustering size and cost, i.e., $\frac{k}{\max\{1,\ldots,n\}} = \frac{k}{n}$ and $\frac{\text{cost}_k}{\max\{\text{cost}_1,\ldots,\text{cost}_n\}} = \frac{\text{cost}_k}{\text{cost}(\text{MST})}$.

Table 2: The accumulated profile error compared to total clustering cost, $\sum_{k=1}^{n} |\widehat{\text{cost}}_k - \text{cost}_k|/\text{cost}(G)$

r	Gowalla	Italy	History	Geology
100	0.578	0.230	0.054	0.023
1000	0.123	0.067	0.009	0.005
10000	0.033	0.024	0.003	0.003
20000	0.023	0.017	0.001	0.002

As shown in the figures, different countries have different structures, especially, Asia has a very different structure from other countries; and road networks differ from localization based datasets. Besides, co-citation graphs in various areas also have different structures, and Spotify is far more different from them. The accumulated approximation error of $\widehat{\text{cost}}_k$ is shown in Table 2, which proves that we can estimate the profiles efficiently and with bounded error, compared to total clustering cost $\text{cost}(G)$.

For more details on experiment results, refer to Section D.

Acknowledgment

The work was conducted in part while Pan Peng and Christian Sohler were visiting the Simons Institute for the Theory of Computing as participants in the Sublinear Algorithms program.

References

[ACL06] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486. IEEE, 2006.

[ACM+22] Sepehr Assadi, Vaggos Chatziafratis, Vahab Mirrokni, Chen Wang, et al. Hierarchical clustering in graph streams: Single-pass algorithms and space lower bounds. In *Conference on Learning Theory*, pages 4643–4702. PMLR, 2022.

[AKLL25] Hyung-Chan An, Mong-Jen Kao, Changyeol Lee, and Mu-Ting Lee. Handling lp-rounding for hierarchical clustering and fitting distances by ultrametrics, 2025.

[AKLP22] Arpit Agarwal, Sanjeev Khanna, Huan Li, and Prathamesh Patil. Sublinear algorithms for hierarchical clustering. *Advances in Neural Information Processing Systems*, 35:3417–3430, 2022.

[AVB20] Ilya Amburg, Nate Veldt, and Austin R. Benson. Clustering in graphs and hypergraphs with categorical edge labels. In *Proceedings of the Web Conference*, 2020.

[BAS+18] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 2018.

[BBD+17] MohammadHossein Bateni, Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Raimondas Kiveris, Silvio Lattanzi, and Vahab Mirrokni. Affinity clustering: Hierarchical clustering at scale. *Advances in Neural Information Processing Systems*, 30, 2017.

[BD98] Rene Rivera Beman Dawes, David Abrahams. Cpplus library boost official website, 1998.

[BKT18] Austin R. Benson, Ravi Kumar, and Andrew Tomkins. Sequences of sets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2018.

[CAKMTM19] Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. Hierarchical clustering: Objective functions and algorithms. *Journal of the ACM (JACM)*, 66(4):1–42, 2019.

[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.

[CCN19] Moses Charikar, Vaggos Chatziafratis, and Rad Niazadeh. Hierarchical clustering better than average-linkage. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2291–2304. SIAM, 2019.

[CEF+05] Artur Czumaj, Funda Ergün, Lance Fortnow, Avner Magen, Ilan Newman, Ronitt Rubinfeld, and Christian Sohler. Approximating the weight of the euclidean minimum spanning tree in sublinear time. *SIAM Journal on Computing*, 35(1):91–109, 2005.

[CML11] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090, 2011.

[CRT05] Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM Journal on computing*, 34(6):1370–1379, 2005.

[CS09] Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. *SIAM Journal on Computing*, 39(3):904–922, 2009.

[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.

[DIM10] DIMACS. 9th dimacs implementation challenge: Shortest paths, 2010.

[DIM11] DIMACS. 10th dimacs implementation challenge: Graph partioning and graph clustering, 2011.

[DL05] Sanjoy Dasgupta and Philip M Long. Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences*, 70(4):555–569, 2005.

[For10] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

[GKL+21] Grzegorz Gluch, Michael Kapralov, Silvio Lattanzi, Aida Mousavifar, and Christian Sohler. Spectral clustering oracles in sublinear time. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1598–1617. SIAM, 2021.

[GR69] John C Gower and Gavin JS Ross. Minimum spanning trees and single linkage cluster analysis. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 18(1):54–64, 1969.

[HP19] Zengfeng Huang and Pan Peng. Dynamic graph stream algorithms in o (n) space. *Algorithmica*, 81:1965–1987, 2019.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[KKL+25] Michael Kapralov, Akash Kumar, Silvio Lattanzi, Aida Mousavifar, and Weronika Wrzos-Kaminska. Approximating dasgupta cost in sublinear time from a few random seeds, 2025.

[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.

[KLCB20] Raunak Kumar, Paul Liu, Moses Charikar, and Austin R. Benson. Retrieving top weighted triangles in graphs. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, 2020.

[LNRW10] Guolong Lin, Chandrashekhar Nagarajan, Rajmohan Rajaraman, and David P Williamson. A general approach for incremental approximation and hierarchical clustering. *SIAM Journal on Computing*, 39(8):3633–3669, 2010.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[MU17] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

[MW23] Benjamin Moseley and Joshua R Wang. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. *Journal of Machine Learning Research*, 24(1):1–36, 2023.

[Pen20] Pan Peng. Robust clustering oracle and local reconstructor of cluster structure of graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2953–2972. SIAM, 2020.

[PS18] Pan Peng and Christian Sohler. Estimating graph parameters from random order streams. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2449–2466. SIAM, 2018.

[SP23] Ranran Shen and Pan Peng. A sublinear-time spectral clustering oracle with improved preprocessing time. *Advances in Neural Information Processing Systems*, 36, 2023.

[SSS+15] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. An overview of microsoft academic service (MAS) and applications. In *Proceedings of the 24th International Conference on World Wide Web*. ACM Press, 2015.

[ST13] Daniel A Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on computing*, 42(1):1–26, 2013.

A Generalizing to Unknown d and Non-integer Edge Weights

A.1 Handling the Case When d Is Unknown

In Algorithm 1 (or Algorithm 7), in order to estimate the number of connected components c (or $D = n - c$), we sample several vertices, and do BFS from them, regarding to two thresholds, $\Gamma = \lceil 4k/\varepsilon \rceil$ and $d^{(G)} = d \cdot \Gamma$. Note that $k = \sqrt{W}$ in Algorithm 1 and $k = W$ in Algorithm 7.

In case when d is not given, we refer to [CRT05] and sample 2Γ vertices, and let $d^{(G)}$ to be the maximum degree among them. Note that we can compute $d^{(G)}$ in $d \cdot \Gamma$ time in expectation, because for each sampled vertex v, we know its degree in $\deg(v)$ time. We also argue that with high constant probability the rank of $d^{(G)}$ is $\Theta(\frac{n}{\Gamma})$, and the number of vertices with degree greater than $d^{(G)}$, is at most $\frac{n}{4\Gamma}$. We have

$$\Pr\left[\text{rank of } d^{(G)} > \frac{n}{4\Gamma}\right] \leq \left(1 - \frac{1}{4\Gamma}\right)^{2\Gamma} \leq \mathrm{e}^{-8}$$

On the other hand, we also have

$$\Pr\left[\text{rank of } d^{(G)} > \frac{n}{16\Gamma}\right] = \left(1 - \frac{1}{16\Gamma}\right)^{2\Gamma} \geq \mathrm{e}^{-\frac{2}{8}} \geq 1 - \frac{1}{4} = \frac{3}{4}$$

Therefore, the rank of $d^{(G)}$ is in the interval $(\frac{n}{16\Gamma}, \frac{n}{4\Gamma})$ with probability at least $1-(e^{-8}+1/4)$, which implies that $d^{(G)} \leq 16d \cdot \Gamma$ and that the number of connected components containing vertices with degree greater than $d^{(G)}$ (with rank lower than $d^{(G)}$'s rank), is at most $\frac{n}{\Gamma}$. Furthermore, the number of connected components with size greater than Γ is at most $\frac{n}{\Gamma}$. Then we have that,

$$c - \frac{2n}{\Gamma} \leq c_U \leq c.$$

For the remaining part to prove theoretical guarantee of c or $D = n - c$, one can refer to proofs of Lemma 3.1 or Lemma 7.4.

A.2 Handling Non-integer Edge Weights

When the weights come from the interval $[1,W]$ and are not necessarily integers and we have $1 > \varepsilon > 0$, we can first multiply the weights with $1/\varepsilon$ and then round them down to the nearest integer value. The resulting weights are integers from $\{1,\ldots,\lfloor W/\varepsilon \rfloor\}$. Let us call the scaled weights w' and the rounded and scaled weights w''. Each edge weight differs at most 1 from its correct value, that is for every each e we have $|w'(e) - w''(e)| \leq 1$. If we use M to denote the cost of the MST with the original weights, M' the cost of the MST with the scaled weights and M'' the cost of the MST with the scaled and rounded weights, we get $M = \varepsilon M'$ and $|M' - M''| \leq n-1$. It follows that $|M - \varepsilon M''| \leq \varepsilon(n-1) \leq \varepsilon M$. Thus, we can approximate the cost of the MST with scaled and rounded weights. The resulting running time in the distance case will be $\tilde{O}(\sqrt{W}/\varepsilon^{3.5})$. The error will be at most $(1 \pm \varepsilon)^2 \leq 1 \pm 3\varepsilon$. Replacing ε with $\varepsilon/3$ gives a $(1 \pm \varepsilon)$-approximation.

B Sublinear Algorithms in Metric Space: Distance Case

In this scenario, we consider a weighted graph G in metric space, where the weight of each edge is the distance between its two connected vertices, and edge weights satisfy the triangle inequality. Our only assumption is that the distance between any pair of vertices can be accessed in constant time. Using the same definition of w_i and $\text{cost}(G)$ as in the distance case, it holds that $\text{cost}_k = \sum_{i=1}^{n-k} w_i$, and $\text{cost}(G) = \sum_{k=1}^{n} \text{cost}_k = \sum_{i=1}^{n-1} (n-i) \cdot w_i$.

Here, the maximum weight W is the longest distance in G, which can be approximated in $O(n)$ time, with an approximation factor of $1/2$. Let this estimate be denoted as W^*, then $W^* \leq W \leq 2W^*$. We then rescale the distances such that $W^* = 2n^2/\varepsilon$. After scaling, all distances are in $[0, 4n^2/\varepsilon]$, since the longest distance is at most $2W^*$. By the triangle inequality, the cost of the Minimum Spanning Tree (MST), denoted as $\text{cost}(\text{MST})$, is at least as large as the longest distance, and hence at least W^*. Therefore, $\text{cost}(\text{MST}) \geq 2n^2/\varepsilon$. Since $\text{cost}(\text{MST}) = \text{cost}_1 \leq \text{cost}(G)$, it follows that $\text{cost}(G) \geq 2n^2/\varepsilon$.

To simplify our analysis, we round up all edge weights less than 1 to be exactly 1. Considering the formula $\text{cost}(G) = \sum_{i=1}^{n-1} (n-i) \cdot w_i$, this rounding operation affects the cost as follows: if every edge in the MST has a weight below 1, then $\text{cost}(G)$ is increased by at most $\sum_{i=1}^{n-1} (n-i) \cdot 1 = \frac{n(n-1)}{2} \leq \frac{\varepsilon}{4} \cdot \frac{2n^2}{\varepsilon} \leq \frac{\varepsilon}{4}\text{cost}(G)$, thereby introducing an error term of $\varepsilon \cdot \text{cost}(G)/4$. Moreover, this rounding operation preserves the triangle inequality. For any triangle with edge weights a, b and c, there are two relevant cases. First, consider the inequality $a+b>c$. The worst scenario is when c is increased from 0 to 1 and one of a or b is less than 1. After rounding, $a+b \geq 1$ and $c=1$, so $a+b>c$ still holds. Second, consider $a-b<c$. The worst scenario is when $b \leq a < 1$ and both become 1 after rounding. Then $a-b=0<c$ still holds. Thus, the triangle inequality is preserved.

We then increase the upper bound of weights to the nearest power of $(1+\varepsilon)$, i.e., we let $W = (1+\varepsilon)^r$, where $r = \lceil \log_{1+\varepsilon}(4n^2/\varepsilon) \rceil = O(\log(n/\varepsilon)/\varepsilon)$. Consequently, for the remainder of the metric space, we can assume that all the edge weights are within $[1,(1+\varepsilon)^r]$.

B.1 Cost Formula for $\mathrm{cost}(G)$

We begin by deriving a formula for the clustering cost, $\mathrm{cost}(G)$, which simplifies the estimation process.

Lemma B.1. *Let G be an n-point graph in metric space such that all pairwise distances are in the interval $[1,W]$, where $W = (1+\varepsilon)^r$. Let $\ell_j = (1+\varepsilon)^j$. For any $0 \le j \le r$, we let G_j denote the subgraph of G spanned by all edges with weights* at most *ℓ_j, and let c_j denote the number of connected components in G_j. Then we have*

$$\mathrm{cost}(G) \le \frac{n(n-1)}{2} + \frac{\varepsilon}{2} \cdot \sum_{j=0}^{r-1} (1+\varepsilon)^j \cdot (c_j^2 - c_j) \le (1+\varepsilon)\mathrm{cost}(G)$$

Proof. We let G' be the weighted graph obtained by rounding every edge weight in G to the nearest power of $(1+\varepsilon)$. For example, for any pair (u,v), if the distance is $(1+\varepsilon)^j < d(u,v) \le (1+\varepsilon)^{j+1}$, then we round $d(u,v)$ to be $(1+\varepsilon)^{j+1}$. After rounding, the i-th smallest edge weight on the MST of G' becomes w'_i. Since we only increased the edge weights by at most a factor of $(1+\varepsilon)$, the cost of clustering is also increased, by at most a factor of $(1+\varepsilon)$. That is,

$$\mathrm{cost}(G) \le \mathrm{cost}(G') \le (1+\varepsilon)\mathrm{cost}(G).$$

Note that after rounding, the threshold graphs in G' are not changed, and thus each c_j has the same value with respect to G. By running Kruskal's algorithm on G', we will add $n - c_0$ edges of weight $\ell_0 = 1$, $c_0 - c_1$ edges of weight ℓ_1, and so on. Thus, the clustering cost of G' can be written as,

$$\begin{aligned}
\mathrm{cost}(G') &= \sum_{i=1}^{n-1} (n-i) \cdot w'_i \\
&= \sum_{i=1}^{n-c_0} (n-i)\cdot \ell_0 + \sum_{i=n-c_0+1}^{n-c_1} (n-i)\cdot \ell_1 + \cdots + \sum_{i=n-c_{r-1}+1}^{n-c_r} (n-i)\cdot \ell_r \\
&= \sum_{i=1}^{n-c_r} (n-i)\cdot \ell_0 - \sum_{i=n-c_0+1}^{n-c_r} (n-i)\cdot \ell_0 + \sum_{i=n-c_0+1}^{n-c_r} (n-i)\cdot \ell_1 - \sum_{i=n-c_1+1}^{n-c_r} (n-i)\cdot \ell_1 + \\
&\quad \sum_{i=n-c_1+1}^{n-c_r} (n-i)\cdot \ell_2 - \sum_{i=n-c_2+1}^{n-c_r} (n-i)\cdot \ell_2 + \cdots + \sum_{i=n-c_{r-1}+1}^{n-c_r} (n-i)\cdot \ell_r \\
&= \frac{n(n-1)}{2} + (\ell_1 - \ell_0)\cdot \frac{1}{2}\cdot (c_0^2 - c_0) + (\ell_2 - \ell_1)\cdot \frac{1}{2}\cdot (c_1^2 - c_1) + \cdots + (\ell_r - \ell_{r-1})\cdot \frac{1}{2}\cdot (c_{r-1}^2 - c_{r-1}) \\
&= \frac{n(n-1)}{2} + \frac{1}{2}\sum_{j=0}^{r-1} (\ell_{j+1} - \ell_j)(c_j^2 - c_j) \\
&= \frac{n(n-1)}{2} + \frac{\varepsilon}{2}\sum_{j=0}^{r-1} (1+\varepsilon)^j \cdot (c_j^2 - c_j) \qquad (\text{since } \ell_{j+1} - \ell_j = (1+\varepsilon)^{j+1} - (1+\varepsilon)^j = \varepsilon(1+\varepsilon)^j)
\end{aligned}$$

This completes the proof of the lemma. □

B.2 Algorithm to Estimate Clustering Cost

We make use of the algorithm developed by [CS09] for estimating MST. This algorithm is fundamentally based on estimating the number of connected components of each subgraph G_j. The core method for estimating the number of connected components is called CLIQUE-TREE-TRAVERSAL.

The main idea of CLIQUE-TREE-TRAVERSAL is as follows: If two vertices u, v have distance less than $\varepsilon(1+\varepsilon)^j$, then they have the same neighbors in G_j, according to triangle inequality. Therefore, we can select vertices with pairwise distances at least $\varepsilon(1+\varepsilon)^j$ as *representative vertices*, and do traversal based on them. To save running time, the two thresholds during the traversal are:

- Before traversal, pick a value X with distribution $\Pr[X \geq k] = 1/k$. If more than X vertices are explored (including both *representative* and *non-representative vertices*), then quit the traversal.
- If more then $4r/\varepsilon$ *representative vertices* are explored, then quit the traversal.

Having the subroutine CLIQUE-TREE-TRAVERSAL, we propose our algorithm in Algorithm 11.

Algorithm 11: APPCOSTMETRIC(G, ε)

1 for each $j \in \{0, \ldots, r-1\}$, invoke CLIQUE-TREE-TRAVERSAL to obtain $\hat{c}_j$

2 output $\widehat{\text{cost}}(G) = \frac{n(n-1)}{2} + \frac{\varepsilon}{2}\sum_{j=0}^{r-1}(1+\varepsilon)^j \cdot (\hat{c}_j^2 - \hat{c}_j)$

Theorem 1.7. *Let G be an n-point graph in metric space, where each edge weight represents* distance *between two vertices, and $0 < \varepsilon < 1$ be a parameter. Algorithm 11 outputs an estimate $\widehat{\text{cost}}(G)$ of single-linkage clustering cost* $\text{cost}(G)$ *in metric space, such that with probability at least* $3/4$*,*

$$(1-\varepsilon)\text{cost}(G) \leq \widehat{\text{cost}}(G) \leq (1+\varepsilon)\text{cost}(G).$$

The query complexity and running time of the algorithm are $\tilde{O}(n/\varepsilon^7)$ in expectation.

B.3 Analysis of Algorithm 11

According to [CS09], CLIQUE-TREE-TRAVERSAL has the following performance guarantee.

Lemma B.2. *Let $U_{rep}^{(j)}$ be defined as in [CS09] (a set of* representative vertices *in the whole graph. This set is obtained by full* CLIQUE-TREE-TRAVERSAL*, and with maximum cardinality). For any given ε, $j \in [1, r]$, there exists an algorithm that computes in time $\tilde{O}(n/\varepsilon^6)$ and outputs a value $\hat{c}_j$ such that*

$$\left|\hat{c}_j - c_j\right| \leq c_j - c_{j+1} + \frac{3\varepsilon}{8r}\left|U_{rep}^{(j)}\right|$$

Proof. From [CS09], we know that $c_{j+1} - K \leq \mathbf{E}[\hat{c}_j] \leq c_j$, where K is the number of connected components in G_{j+1} where there exists a vertex p, such that starting from p will stop with more than $\frac{4r}{\varepsilon}$ representative vertices. By the definition of K, we have that $K \cdot \frac{4r}{\varepsilon} \leq |U_{rep}^{(j)}|$. Thus,

$$c_{j+1} - \frac{\varepsilon}{4r}|U_{rep}^{(j)}| \leq \mathbf{E}[\hat{c}_j] \leq c_j$$

Besides, $\Pr[|\hat{c_j} - \mathbf{E}[\hat{c}_j]| \geq \frac{\varepsilon}{8r}|U_{rep}^{(j)}|] \leq \frac{1}{16}$. Thus, with probability more than $15/16$, we have

$$c_{j+1} - \frac{3\varepsilon}{8r}|U_{rep}^{(i)}| = c_{j+1} - \frac{\varepsilon}{4r}|U_{rep}^{(j)}| - \frac{\varepsilon}{8r}|U_{rep}^{(j)}| \leq \hat{c}_j \leq c_j + \frac{\varepsilon}{8r}|U_{rep}^{(j)}|$$

Therefore, $\hat{c}_j - c_j \leq \frac{\varepsilon}{8r}|U_{rep}^{(j)}|$ and $c_j - \hat{c}_j \leq c_j - c_{j+1} + \frac{3\varepsilon}{8r}|U_{rep}^{(j)}|$, which completes the proof.

□

Once we bound the error of estimate $\hat{c}_j$ by $\left|U_{rep}^{(j)}\right|$, we can bound $\left|U_{rep}^{(j)}\right|$ by $\mathrm{cost}(G)$, and this will help us prove the approximation ratio of estimating clustering cost.

Lemma B.3. *Assume that all edge weights are in* $[1, 4n^2/\varepsilon]$. *It holds that for any* $1 \leq j \leq r$,

$$\mathrm{cost}(G) \geq \frac{\varepsilon(1+\varepsilon)^j}{16} \cdot |U_{rep}^{(j)}| \cdot c_j$$

Proof. When the number of clusters developed in single-linkage is exactly c_j, the cost of them is cost_{c_j}. As subgraphs $G_1, \ldots, G_r$ are built in increasing edge weights, the vertices in the connected components in G_j exactly build these single-linkage clusters. Suppose for a certain connected component $S \in G_j$, there are s vertices in total, and x of them are *representative vertices*. As the pairwise distance between *representative vertices* is at least $\varepsilon(1+\varepsilon)^j$, and all the weights are at least 1, then the cost of MST in S is at least $\varepsilon(1+\varepsilon)^j \cdot (x-1) + 1 \cdot (s-x)$. Since $x \leq s$, then $\mathrm{cost}_S(\mathrm{MST}) \geq \varepsilon \cdot (1+\varepsilon)^j(x-1)$. Therefore, by summing up cost for each component $S \in G_j$, we have

$$\mathrm{cost}_{c_j} \geq \sum_{S \in G_j} \mathrm{cost}_S(\mathrm{MST}) \geq \sum_{S \in G_j} \varepsilon \cdot (1+\varepsilon)^j(x-1) = \varepsilon(1+\varepsilon)^j \left(\left|U_{rep}^{(j)}\right| - c_j\right)$$

If we have fewer clusters than c_j, we need to use more expensive edges to connect two clusters; then we will have more cost there. That is to say, for all $1 \leq k \leq c_j$, $\mathrm{cost}_k \geq \mathrm{cost}_{c_j}$. Therefore,

$$\mathrm{cost}(G) = \sum_{k=1}^{n} \mathrm{cost}_k \geq \sum_{k=1}^{c_j} \mathrm{cost}_k \geq \mathrm{cost}_{c_j} \cdot c_j \geq \varepsilon(1+\varepsilon)^j \left(\left|U_{rep}^{(j)}\right| - c_j\right) \cdot c_j$$

Case I. When $\left|U_{rep}^{(j)}\right| \geq 2c_j$, we have $\left|U_{rep}^{(j)}\right| - c_j \geq \frac{1}{2}\left|U_{rep}^{(j)}\right|$, and so $\mathrm{cost}(G) \geq \varepsilon(1+\varepsilon)^j \left|U_{rep}^{(j)}\right| c_j/2$.

Case II. When $\left|U_{rep}^{(j)}\right| < 2c_j$. From Lemma B.1, we have

$$\mathrm{cost}(G) \geq \frac{n(n-1)}{2(1+\varepsilon)} + \frac{\varepsilon}{2(1+\varepsilon)} \sum_{j=0}^{r-1} (1+\varepsilon)^j \cdot (c_j^2 - c_j) \geq \frac{\varepsilon}{2(1+\varepsilon)} (1+\varepsilon)^j \cdot (c_j^2 - c_j) \geq \frac{\varepsilon}{4}(1+\varepsilon)^j \cdot (c_j^2 - c_j).$$

If $c_j = 1$, from [CS09] we know that $\mathrm{cost}(\mathrm{MST}) \geq \varepsilon \cdot (1+\varepsilon)^j \cdot \left|U_{rep}^{(j)}\right|/4$, and as $\mathrm{cost}(G) \geq \mathrm{cost}_1 = \mathrm{cost}(\mathrm{MST})$, the lemma is true. Otherwise, $c_j \geq 2$, then $c_j^2 - c_j \geq \frac{c_j^2}{2}$, and so $\mathrm{cost}(G) \geq \varepsilon(1+\varepsilon)^j c_j^2/8 > \varepsilon(1+\varepsilon)^j \left|U_{rep}^{(j)}\right| c_j/16$.

This completes the proof of the lemma. □

Similarly, we have the following lemma.

Lemma B.4. *It holds that for any $j \in [1,r]$,*

$$\mathrm{cost}(G) \geq \frac{\varepsilon^2(1+\varepsilon)^j}{8r} \cdot |U_{rep}^{(j)}|^2$$

Proof. First of all, suppose $\varepsilon(1+\varepsilon)^j > 1$. Recall that for any two vertices $u, b \in U_{rep}^{(j)}$, their distance is at least $d(u,v) \geq \varepsilon(1+\varepsilon)^j$. Then, for any i such that $(1+\varepsilon)^i \leq \varepsilon(1+\varepsilon)^j$, the subgraph G^i contains all the edges with weight at most $(1+\varepsilon)^i$. Thus, every vertex in the $U_{rep}^{(j)}$ falls in a separate connected component. Therefore, $c_i \geq |U_{rep}^{(j)}|$ and

$$\begin{aligned}
\mathrm{cost}(G) &\geq \frac{\varepsilon}{2} \sum_{i:(1+\varepsilon)^i \leq \varepsilon(1+\varepsilon)^j} (1+\varepsilon)^i \cdot (c_i^2 - c_i) \\
&\geq \frac{\varepsilon}{4} \sum_{i:(1+\varepsilon)^i \leq \varepsilon(1+\varepsilon)^j} (1+\varepsilon)^i \cdot c_i^2 \\
&\geq \frac{\varepsilon}{4} \cdot |U_{rep}^{(j)}|^2 \cdot [(1+\varepsilon)^0 + (1+\varepsilon)^1 + \cdots + \varepsilon(1+\varepsilon)^j] \\
&\geq \frac{\varepsilon}{8} \cdot |U_{rep}^{(j)}|^2 \cdot \varepsilon(1+\varepsilon)^j \\
&\geq \frac{\varepsilon^2(1+\varepsilon)^j}{8r} \cdot |U_{rep}^{(j)}|^2
\end{aligned}$$

Besides, if $\varepsilon(1+\varepsilon)^j \leq 1$, we have $\frac{\varepsilon^2(1+\varepsilon)^j}{8r}|U_{rep}^{(j)}|^2 \leq \frac{\varepsilon}{8r}n^2 \leq \mathrm{cost}(G)$, as $|U_{rep}^{(j)}| \leq n$ and $\mathrm{cost}(G) \geq \frac{n(n-1)}{2} \geq \frac{\varepsilon}{8r}n^2$. This completes the proof of the lemma.

□

Given the above analysis, we are ready to prove the guarantee of estimating clustering cost in metric space.

Proof of Theorem 1.7. From Lemma B.2 we know that $\hat{c}_j \leq c_j + \frac{\varepsilon}{8r}|U_{rep}^{(j)}|$, and then

$$|\hat{c}_j^2 - c_j^2| = |\hat{c}_j - c_j| \cdot |\hat{c}_j + c_j| \leq |\hat{c}_j - c_j| \cdot (2c_j + \frac{\varepsilon}{8r}|U_{rep}^{(j)}|)$$

Let $\widehat{\mathrm{cost}}(G) = \frac{n(n-1)}{2} + \frac{\varepsilon}{2}\sum_{j=0}^{r-1}(1+\varepsilon)^j \cdot (\hat{c}_j^2 - \hat{c}_j)$, and $\mathrm{cost}(G') = \frac{n(n-1)}{2} + \frac{\varepsilon}{2}\sum_{j=0}^{r-1}(1+\varepsilon)^j \cdot (c_j^2 - c_j)$. Then we have,

$$\begin{aligned}
\left|\widehat{\mathrm{cost}}(G) - \mathrm{cost}(G')\right| &\leq \frac{\varepsilon}{2}\sum_{j=0}^{r-1}(1+\varepsilon)^j \cdot |(\hat{c}_j^2 - \hat{c}_j) - (c_j^2 - c_j)| \\
&\leq \frac{\varepsilon}{2}\sum_{j=0}^{r-1}(1+\varepsilon)^j \cdot (\left|\hat{c}_j^2 - c_j^2\right| + \left|\hat{c}_j - c_j\right|) \\
&\leq \frac{\varepsilon}{2}\sum_{j=0}^{r-1}(1+\varepsilon)^j \cdot \left|\hat{c}_j - c_j\right| \cdot \left(2c_j + \frac{\varepsilon}{8r}|U_{rep}^{(j)}| + 1\right) \qquad \text{(by the bound of } \left|\hat{c}_j^2 - c_j^2\right|) \\
&\leq \frac{\varepsilon}{2}\sum_{j=0}^{r-1}(1+\varepsilon)^j \cdot \left(c_j - c_{j+1} + \frac{3\varepsilon}{8r}|U_{rep}^{(j)}|\right) \cdot \left(3c_j + \frac{\varepsilon}{8r}|U_{rep}^{(j)}|\right) \\
&\qquad \text{(by the bound of } \left|\hat{c}_j - c_j\right|, \text{ and } 1 \leq c_j)
\end{aligned}$$

$$
\begin{aligned}
&\leq \varepsilon \sum_{j=0}^{r-1}(1+\varepsilon)^j \cdot \left(\frac{3}{2}c_j^2 - \frac{3}{2}c_j c_{j+1} - \frac{\varepsilon}{16r}|U_{rep}^{(j)}| \cdot c_{j+1} + \frac{10\varepsilon}{16r}|U_{rep}^{(j)}| \cdot c_j + \frac{3\varepsilon^2}{128r^2} \cdot |U_{rep}^{(j)}|^2\right) \\
&\leq \varepsilon \sum_{j=0}^{r-1}(1+\varepsilon)^j \cdot \left(\frac{3}{2}c_j^2 - \frac{3}{2}c_j c_{j+1} - \frac{\varepsilon}{16r}|U_{rep}^{(j)}| \cdot c_{j+1}\right) + \varepsilon \sum_{j=0}^{r-1}(\frac{10}{r}\text{cost}(G) + \frac{3}{16r}\text{cost}(G)) \\
&\qquad\qquad \text{(by Lemma B.3 and Lemma B.4)} \\
&\leq \varepsilon \sum_{j=0}^{r-1}(1+\varepsilon)^j \cdot \left(\frac{3}{2}c_j^2 - \frac{3}{2}c_j c_{j+1}\right) + \varepsilon \cdot r \cdot (\frac{10}{r} + \frac{3}{16r})\text{cost}(G) \\
&\leq (*) + 11\varepsilon \cdot \text{cost}(G)
\end{aligned}
$$

Where $(*) = \varepsilon \sum_{j=0}^{r-1}(1+\varepsilon)^j \cdot \left(\frac{3}{2}c_j^2 - \frac{3}{2}c_j c_{j+1}\right)$. For the first term in the summation, by the definition of $\text{cost}(G')$, we have that $\varepsilon \sum_{j=0}^{r-1}(1+\varepsilon)^j \cdot \frac{3}{2}c_j^2 = 3(\text{cost}(G) - \frac{n(n-1)}{2} + \frac{\varepsilon}{2}\sum_{j=0}^{r-1} c_j)$. For the second term in the summation,

$$
\varepsilon \sum_{j=0}^{r-1}(1+\varepsilon)^j \cdot \frac{3}{2}c_j \cdot c_{j+1} \geq \frac{3}{1+\varepsilon} \cdot \frac{\varepsilon}{2}\sum_{j=0}^{r-1}(1+\varepsilon)^{j+1}c_{j+1}^2 = \frac{3}{1+\varepsilon} \cdot \frac{\varepsilon}{2}(\sum_{j=0}^{r-1}(1+\varepsilon)^j c_j^2 - (1+\varepsilon)^0 c_0^2 + (1+\varepsilon)^r c_r^2)
$$

Denote the weight of the minimum spanning tree in the graph G' as $\text{cost}(\text{MST})$. Note that $\text{cost}_1 = \text{cost}(\text{MST}) = n - W + \varepsilon \cdot \sum_{j=0}^{r-1}(1+\varepsilon)^j c_j$ according to [CS09]. As $(1+\varepsilon)^0 c_0^2 = n^2$ and $(1+\varepsilon)^r c_r^2 = W$, we have that

$$
\begin{aligned}
(*) &\leq 3\left(\text{cost}(G') - \frac{n(n-1)}{2} + \frac{\varepsilon}{2}\sum_{j=0}^{r-1} c_j\right) - \frac{3}{1+\varepsilon} \cdot \frac{\varepsilon}{2}\left(\sum_{j=0}^{r-1}(1+\varepsilon)^j c_j^2 - n^2 + W\right) \\
&\leq 3\left(\text{cost}(G') - \frac{n(n-1)}{2} + \frac{\varepsilon}{2}\sum_{j=0}^{r-1} c_j\right) - \frac{3}{1+\varepsilon}\left(\text{cost}(G') - \frac{n(n-1)}{2} + \frac{\varepsilon}{2}\sum_{j=0}^{r-1} c_j - \frac{\varepsilon}{2}n^2 + \frac{\varepsilon}{2}W\right) \\
&\qquad\qquad \text{(by the definition of } \text{cost}(G')) \\
&\leq 3\left(\text{cost}(G') - \frac{n(n-1)}{2} + \frac{\varepsilon}{2}\sum_{j=0}^{r-1} c_j\right) - 3(1-\varepsilon)\left(\text{cost}(G') - \frac{n(n-1)}{2} + \frac{\varepsilon}{2}\sum_{j=0}^{r-1} c_j - \frac{\varepsilon}{2}n^2\right) \\
&\qquad\qquad \text{(since } \tfrac{1}{1+\varepsilon} \geq 1-\varepsilon \text{ and } \varepsilon < 1) \\
&= 3\varepsilon \cdot \text{cost}(G') - 3\varepsilon\frac{n(n-1)}{2} + \frac{3\varepsilon^2}{2}\sum_{j=0}^{r-1} c_j + \frac{3\varepsilon}{2}n^2 \qquad \text{(since } 1-\varepsilon \leq 1) \\
&\leq 3\varepsilon \cdot \text{cost}(G') + \frac{3\varepsilon}{2}(\text{cost}_1 - n + W) + \frac{3\varepsilon}{2}n^2 \qquad \text{(by the definition of } \text{cost}_1) \\
&\leq 3\varepsilon \cdot \text{cost}(G') + \frac{3\varepsilon}{2} \cdot 2\text{cost}_1 + \frac{3\varepsilon}{2} \cdot 4\text{cost}(G') \\
&\qquad\qquad \text{(since } \text{cost}_1 = \text{cost}(\text{MST}) \geq W \text{ and } n^2 \leq 2n(n-1) \leq 4\text{cost}(G') \text{ for } n \geq 2) \\
&\leq 12\varepsilon \cdot \text{cost}(G') \qquad \text{(since } \text{cost}_1 \leq \text{cost}(G'))
\end{aligned}
$$

Therefore, $\left|\widehat{\text{cost}}(G) - \text{cost}(G')\right| \leq 12\varepsilon \cdot \text{cost}(G') + 11\varepsilon \cdot \text{cost}(G') = 23\varepsilon \cdot \text{cost}(G')$. From Lemma B.1, we have that $|\text{cost}(G') - \text{cost}(G)| \leq \varepsilon \cdot \text{cost}(G)$, and so

$$
\left|\widehat{\text{cost}}(G) - \text{cost}(G)\right| \leq \left|\widehat{\text{cost}}(G) - \text{cost}(G')\right| + \left|\text{cost}(G') - \text{cost}(G)\right| \leq 23\varepsilon \cdot \text{cost}(G') + \varepsilon \cdot \text{cost}(G) \leq 48\varepsilon \cdot \text{cost}(G)
$$

Replacing ε with $\varepsilon/48$, we get a $(1+\varepsilon)$ estimate of $\mathrm{cost}(G)$.

Running time analysis. since each invocation on CLIQUE-TREE-TRAVERSAL takes $\tilde{O}(n/\varepsilon^6)$ time, and we invoke it for $r = O(\log(n/\varepsilon)/\varepsilon)$ times, the total running time is $r \cdot \tilde{O}(n/\varepsilon^6) = \tilde{O}(n/\varepsilon^7)$. □

C Sublinear Algorithms in Metric Space: Similarity Case

We now give sublinear algorithms for SLC cost when the edge weight in a metric graph represents similarity relationship. We obtain the following result.

Theorem C.1. *Let G be an n-point graph in metric space, where each edge weight represents* similarity *between two vertices, and $0 < \varepsilon < 1$ be a parameter. Algorithm 12 outputs an estimate $\widehat{\mathrm{cost}^{(s)}}(G)$ of single-linkage clustering cost $\mathrm{cost}^{(s)}(G)$ in metric space, such that with probability at least $3/4$,*

$$(1-\varepsilon)\mathrm{cost}^{(s)}(G) \le \widehat{\mathrm{cost}^{(s)}}(G) \le (1+\varepsilon)\mathrm{cost}^{(s)}(G).$$

The query complexity and running time of the algorithm are $\tilde{O}(n/\varepsilon^7)$ in expectation.

C.1 Cost Formula for $\mathrm{cost}^{(s)}(G)$

Let $\ell_j = (1+\varepsilon)^j$ and $G_j^{(s)}$ be the subgraph of G spanned by all edges with weight at least ℓ_j, and let $c_j^{(s)}$ be the number of connected components in $G_j^{(s)}$. Note that $1 = c_0^{(s)} \le c_1^{(s)} \le \cdots \le c_{r+1}^{(s)} = n$. Let $w_1^{(s)} \ge w_2^{(s)} \ge \cdots \ge w_{n-1}^{(s)}$ be the sorted weights on the *maximum* spanning tree. Assume that any edge weight is $(1+\varepsilon)^j$ in the graph G' for some $0 \le j \le r$, and $(1+\varepsilon)^r = W$. Let n_j be the number of edges with weight j on the maximum spanning tree, and we have $\sum_{j<\ell} n_j = c_\ell^{(s)} - 1$, and thus $n_j = c_{j+1}^{(s)} - c_j^{(s)}$. Then the cost function can be derived in the following theorem.

Lemma C.1. *Let G be an n-point graph in metric space such that all pairwise distances are in the interval $[1, W]$, where $W = (1+\varepsilon)^r$. Let $\ell_j = (1+\varepsilon)^j$. For any $0 \le j \le r$, we let $G_j^{(s)}$ denote the subgraph of G spanned by all edges with weights* at least *ℓ_j, and let $c_j^{(s)}$ denote the number of connected components in $G_j^{(s)}$. Then we have*

$$\mathrm{cost}^{(s)}(G) \le \frac{n(n-1)}{2} + \varepsilon \sum_{j=1}^{r} (1+\varepsilon)^{j-1} \frac{(c_j^{(s)} + n - 1)(n - c_j^{(s)})}{2} \le (1+\varepsilon)\mathrm{cost}^{(s)}(G)$$

Proof. We let G' be the weighted graph obtained by rounding every edge weight in G to the nearest power of $(1+\varepsilon)$. For example, for any pair (u,v), if the distance is $(1+\varepsilon)^j < d(u,v) \le (1+\varepsilon)^{j+1}$, then we round $d(u,v)$ to be $(1+\varepsilon)^{j+1}$. After rounding, the i-th largest edge weight on the MaxST of G' becomes w'_i. Since we only increased the edge weights by at most a factor of $(1+\varepsilon)$, the cost of clustering is also increased, by at most a factor of $(1+\varepsilon)$. That is,

$$\mathrm{cost}^{(s)}(G) \le \mathrm{cost}^{(s)}(G') \le (1+\varepsilon)\mathrm{cost}^{(s)}(G).$$

Note that after rounding, the threshold graphs in G' are not changed, and thus each $c_j^{(s)}$ has the same value with respect to G. Thus, the clustering cost of G' can be written as,

$$\mathrm{cost}^{(s)}(G') = \sum_{i=1}^{n-1} (n-i) \cdot w_i^{(s)} \qquad \text{(by definition)}$$

$$= \sum_{i=1}^{n_r} (n-i)\cdot \ell_r + \sum_{i=n_r+1}^{n_r+n_{r-1}} (n-i)\cdot \ell_{r-1} + \cdots + \sum_{i=n_r+\cdots+n_1+1}^{n_r+\cdots+n_0} (n-i)\cdot \ell_0$$

(reorganizing the sum by grouping the terms according to edge weights)

$$= \sum_{i=n-c_{r+1}^{(s)}+1}^{n-c_r^{(s)}} (n-i)\cdot \ell_r + \sum_{i=n-c_r^{(s)}+1}^{n-c_{r-1}^{(s)}} (n-i)\cdot \ell_{r-1} + \cdots + \sum_{i=n-c_1^{(s)}+1}^{n-c_0^{(s)}} (n-i)\cdot \ell_0$$

(since $n_j = c_{j+1}^{(s)} - c_j^{(s)}$)

$$= \sum_{i=n-c_{r+1}^{(s)}+1}^{n-c_0^{(s)}} (n-i)\cdot \ell_r - \sum_{i=n-c_r^{(s)}+1}^{n-c_0^{(s)}} (n-i)\cdot \ell_r + \sum_{i=n-c_r^{(s)}+1}^{n-c_0^{(s)}} (n-i)\cdot \ell_{r-1} - \sum_{i=n-c_{r-1}^{(s)}+1}^{n-c_0^{(s)}} (n-i)\cdot \ell_{r-1} + \ldots$$

$$+ \sum_{i=n-c_2^{(s)}+1}^{n-c_0^{(s)}} (n-i)\cdot \ell_1 - \sum_{i=n-c_1^{(s)}+1}^{n-c_0^{(s)}} (n-i)\cdot \ell_1 + \sum_{i=n-c_1^{(s)}+1}^{n-c_0^{(s)}} (n-i)\cdot \ell_0$$

$$= \ell_r \cdot \sum_{i=1}^{n-1} (n-i) - \sum_{i=n-c_r^{(s)}+1}^{n-1} (n-i)\cdot(\ell_r - \ell_{r-1}) - \cdots - \sum_{i=n-c_1^{(s)}+1}^{n-1} (n-i)\cdot(\ell_1 - \ell_0)$$

(since $c_0^{(s)} = 1$ and $c_{r+1}^{(s)} = n$)

$$= (1+\varepsilon)^r \frac{n(n-1)}{2} - \varepsilon(1+\varepsilon)^{r-1}\cdot\frac{c_r^{(s)}(c_r^{(s)}-1)}{2} - \cdots - \varepsilon(1+\varepsilon)^0\cdot\frac{c_1^{(s)}(c_1^{(s)}-1)}{2}$$

$$= \frac{n(n-1)}{2} + \varepsilon\sum_{j=1}^{r}(1+\varepsilon)^{j-1}\cdot\frac{n(n-1)}{2} - \varepsilon\sum_{j=1}^{r}(1+\varepsilon)^{j-1}\cdot\frac{c_j^{(s)}(c_j^{(s)}-1)}{2}$$

(since $(1+\varepsilon)^r = 1+\varepsilon\sum_{j=1}^{r}(1+\varepsilon)^{j-1}$)

$$= \frac{n(n-1)}{2} + \varepsilon\sum_{j=1}^{r}(1+\varepsilon)^{j-1}\frac{(c_j^{(s)}+n-1)(n-c_j^{(s)})}{2}$$

This completes the proof of the lemma. □

The cost of the maximum spanning tree of graph G' can be derived in a similar approach.

$$\begin{aligned}
\text{cost(MaxST)} &= \sum_{j=0}^{r}(1+\varepsilon)^j\cdot n_j \\
&= \sum_{j=0}^{r}(1+\varepsilon)^j(c_{j+1}^{(s)} - c_j^{(s)}) \\
&= \sum_{j'=1}^{r+1}(1+\varepsilon)^{j'-1}\cdot c_{j'}^{(s)} - \sum_{j=0}^{r}(1+\varepsilon)^j\cdot c_j^{(s)} && \text{(substituting } j' = j+1 \text{ in the first sum)} \\
&= \sum_{j=0}^{r}(1+\varepsilon)^{j-1}\cdot c_j^{(s)} + (1+\varepsilon)^r\cdot c_{r+1}^{(s)} - (1+\varepsilon)^{-1}\cdot c_0^{(s)} - \sum_{j=0}^{r}(1+\varepsilon)^j\cdot c_j^{(s)} \\
&= (1+\varepsilon)^r\cdot n - \frac{1}{1+\varepsilon} - \varepsilon\sum_{j=0}^{r}(1+\varepsilon)^{j-1}\cdot c_j^{(s)} && \text{(since } c_{r+1}^{(s)} = n, c_0^{(s)} = 1)
\end{aligned}$$

$$= Wn - 1 - \varepsilon \sum_{j=1}^{r} (1+\varepsilon)^{j-1} \cdot c_j^{(s)} \tag{6}$$

C.2 Estimating $c_j^{(s)}$

For a pair of vertices p and q, if $d(p,q) < \varepsilon(1+\varepsilon)^{j-1}$, then they share the same neighborhood inside the subgraph $G_j^{(s)}$. Because for any neighbor v of p, we have $d(p,v) \geq (1+\varepsilon)^j$, then $d(q,v) \geq d(p,v) - d(p,q) > (1+\varepsilon)^j - \varepsilon(1+\varepsilon)^{j-1} = (1+\varepsilon)^{j-1}$, which means that v is also a neighbor of q. We modify the original algorithm CLIQUE-TREE-TRAVERSAL to obtain CLIQUE-TREE-TRAVERSAL-SIMILARITY, and the only difference is that the representative set V_{rep} regarding to subgraph $G_j^{(s)}$ has pairwise distance at least $\varepsilon(1+\varepsilon)^{j-1}$.

Analogously to the metric space representing distance, we have the following lemma.

Lemma C.2. *Let $U_{rep}^{(j)}$ be a set of* representative vertices *in the whole graph. This set is obtained by full* CLIQUE-TREE-TRAVERSAL-SIMILARITY*, and with maximum cardinality. For any given $\varepsilon,\ j \in [1,r]$, there exists an algorithm that computes in time $\tilde{O}(n/\varepsilon^6)$ and outputs a value $\hat{c}_j^{(s)}$ such that*

$$\left|\hat{c}_j^{(s)} - c_j^{(s)}\right| \leq c_j^{(s)} - c_{j-1}^{(s)} + \frac{3\varepsilon}{8r}\left|U_{rep}^{(j)}\right|$$

Proof. Let $V_p^{(j)}$ be the set of vertices in the connected component of vertex p in the graph $G_j^{(s)}$, and V_{exp} be the set of visited vertices when calling subroutine CLIQUE-TREE-TRAVERSAL-SIMILARITY and starting traversal from p. We have that $V_p^{(j)} \subseteq V_{exp} \subseteq V_p^{(j-1)}$.

Therefore, $c_{j-1}^{(s)} - \frac{\varepsilon}{4r}|U_{rep}^{(j)}| \leq \mathbf{E}[\hat{c}_j^{(s)}] \leq c_j^{(s)}$ and $\Pr[|\hat{c}_j^{(s)} - \mathbf{E}[\hat{c}_j^{(s)}]| \geq \frac{\varepsilon}{8r}|U_{rep}^{(j)}|] \leq \frac{1}{16}$. Thus, with probability more than $15/16$, we have

$$c_{j-1}^{(s)} - \frac{3\varepsilon}{8r}|U_{rep}^{(i)}| = c_{j-1}^{(s)} - \frac{\varepsilon}{4r}|U_{rep}^{(j)}| - \frac{\varepsilon}{8r}|U_{rep}^{(j)}| \leq \hat{c}_j^{(s)} \leq c_j^{(s)} + \frac{\varepsilon}{8r}|U_{rep}^{(j)}|$$

Therefore, $\hat{c}_j^{(s)} - c_j^{(s)} \leq \frac{\varepsilon}{8r}|U_{rep}^{(j)}|$ and $c_j^{(s)} - \hat{c}_j^{(s)} \leq c_j^{(s)} - c_{j-1}^{(s)} + \frac{3\varepsilon}{8r}|U_{rep}^{(j)}|$, which completes the proof. □

C.3 Estimating $\mathrm{cost}^{(s)}(G)$

Algorithm 12: APPCOSTMETRICSIM(G,ε)

1 for each $j \in \{1,\ldots,r\}$, invoke CLIQUE-TREE-TRAVERSAL-SIMILARITY to obtain $\hat{c}_j^{(s)}$

2 output $\widehat{\mathrm{cost}^{(s)}}(G) = \frac{n(n-1)}{2} + \frac{\varepsilon}{2}\sum_{j=1}^{r}(1+\varepsilon)^{j-1} \cdot (\hat{c}_j^{(s)} + n - 1)(n - \hat{c}_j^{(s)})$

Theorem C.1. *Let G be an n-point graph in metric space, where each edge weight represents* similarity *between two vertices, and $0 < \varepsilon < 1$ be a parameter. Algorithm 12 outputs an estimate $\widehat{\mathrm{cost}^{(s)}}(G)$ of single-linkage clustering cost $\mathrm{cost}^{(s)}(G)$ in metric space, such that with probability at least $3/4$,*

$$(1-\varepsilon)\mathrm{cost}^{(s)}(G) \leq \widehat{\mathrm{cost}^{(s)}}(G) \leq (1+\varepsilon)\mathrm{cost}^{(s)}(G).$$

The query complexity and running time of the algorithm are $\tilde{O}(n/\varepsilon^7)$ in expectation.

Proof of Theorem C.1. By Lemma C.2, $\left|\hat{c}_j^{(s)} - c_j^{(s)}\right| \le c_j^{(s)} - c_{j-1}^{(s)} + \frac{3\varepsilon}{8r}\left|U_{rep}^{(j)}\right|$, and we denote this additive error as a. Denote $c_j^{(s)} + n - 1$ as A_j and $n - c_j^{(s)}$ as D_j. Then for each $1 \le j \le r$,

$$\begin{aligned}
&A_j - a \le \widehat{A}_j \le A_j + a \\
&D_j - a \le \widehat{D}_j \le D_j + a \\
&(A_j - a)(D_j - a) \le \widehat{A}_j \cdot \widehat{D}_j \le (A_j + a)(D_j + a)
\end{aligned}$$

Therefore, $|\widehat{A}_j \cdot \widehat{D}_j - A_j \cdot D_j| \le a(A_j + D_j) + a^2$. Since $a = c_j^{(s)} - c_{j-1}^{(s)} + \frac{3\varepsilon}{8r}\left|U_{rep}^{(j)}\right| \le n + \frac{3\varepsilon}{8r} \cdot n \le 2n - 1 = A_j + D_j$, then $|\widehat{A}_j \cdot \widehat{D}_j - A_j \cdot D_j| \le 2a(A_j + D_j) \le (c_j^{(s)} - c_{j-1}^{(s)} + \frac{3\varepsilon}{8r}\left|U_{rep}^{(j)}\right|) \cdot 4n$.

Thus, the error of the estimate $\widehat{\text{cost}^{(s)}}(G)$ is,

$$\begin{aligned}
|\widehat{\text{cost}^{(s)}}(G) - \text{cost}^{(s)}(G')| &\le \frac{\varepsilon}{2}\sum_{j=1}^{r}(1+\varepsilon)^{j-1}|\widehat{A}_j \cdot \widehat{D}_j - A_j \cdot D_j| \\
&\le \frac{\varepsilon}{2}\sum_{j=1}^{r}(1+\varepsilon)^{j-1}(c_j^{(s)} - c_{j-1}^{(s)} + \frac{3\varepsilon}{8r}|U_{rep}^{(j)}|) \cdot 4n
\end{aligned}$$

On the one hand, since all the representative vertices inside $U_{rep}^{(j)}$ have pairwise distances at least $\varepsilon(1+\varepsilon)^{j-1}$, the cost of the maximum spanning tree of G is at least $\text{cost}(\text{MaxST}) \ge \varepsilon(1+\varepsilon)^{j-1}(|U_{rep}^{(j)}| - 1) \ge \frac{\varepsilon}{2}(1+\varepsilon)^{j-1}|U_{rep}^{(j)}|$, if $|U_{rep}^{(j)}| \ge 2$.

On the other hand, we have

$$\begin{aligned}
\sum_{j=1}^{r}(1+\varepsilon)^{j-1}(c_j^{(s)} - c_{j-1}^{(s)}) &= \sum_{j=1}^{r}(1+\varepsilon)^{j-1}c_j^{(s)} - (1+\varepsilon)\sum_{j=1}^{r}(1+\varepsilon)^{j-2}c_{j-1}^{(s)} \\
&= \sum_{j=1}^{r}(1+\varepsilon)^{j-1}c_j^{(s)} - (1+\varepsilon)\sum_{j'=0}^{r-1}(1+\varepsilon)^{j'-1}c_{j'}^{(s)} \\
&\qquad \text{(substituting } j' = j-1 \text{ in the second sum)} \\
&= \sum_{j=1}^{r}(1+\varepsilon)^{j-1}c_j^{(s)} - (1+\varepsilon)\left(\sum_{j=1}^{r}(1+\varepsilon)^{j-1}c_j^{(s)} + (1+\varepsilon)^{-1}c_0^{(s)} - (1+\varepsilon)^{r-1}c_r^{(s)}\right) \\
&= -\varepsilon\sum_{j=1}^{r}(1+\varepsilon)^{j-1}c_j^{(s)} - 1 + (1+\varepsilon)^r c_r^{(s)} \qquad (\text{since } c_0^{(s)} = 1) \\
&= \text{cost}(\text{MaxST}) - Wn + 1 - 1 + Wc_r^{(s)} \qquad (\text{by Eq. (6) and since } (1+\varepsilon)^r = W) \\
&\le \text{cost}(\text{MaxST}) \qquad (\text{since } c_r^{(s)} \le n)
\end{aligned}$$

Therefore, the error of $\widehat{\text{cost}^{(s)}}(G)$ is

$$|\widehat{\text{cost}^{(s)}}(G) - \text{cost}^{(s)}(G')| \le \frac{\varepsilon}{2}\text{cost}(\text{MaxST}) \cdot 4n + \sum_{j=1}^{r}\frac{3\varepsilon}{8r}\text{cost}(\text{MaxST}) \cdot 4n \qquad (\text{by the above analysis})$$

$$= \frac{7\varepsilon}{2}\text{cost}(\text{MaxST}) \cdot n$$

Furthermore, the lower bound of $\text{cost}^{(s)}(G')$ is,

$$\begin{aligned}
\text{cost}^{(s)}(G') &= \frac{n(n-1)}{2} + \varepsilon \sum_{j=1}^{r} (1+\varepsilon)^{j-1} \frac{(c_j^{(s)} + n - 1)(n - c_j^{(s)})}{2} && \text{(by Lemma C.1)} \\
&\geq \frac{n(n-1)}{2} + \frac{n}{2} \cdot \varepsilon \sum_{j=1}^{r} (1+\varepsilon)^{j-1} (n - c_j^{(s)}) && \text{(since } c_j^{(s)} \geq 1\text{)} \\
&= \frac{n(n-1)}{2} + \frac{n}{2} \cdot \left(\varepsilon \cdot \frac{(1+\varepsilon)^0((1+\varepsilon)^r - 1)}{\varepsilon} n - \varepsilon \sum_{j=1}^{r} (1+\varepsilon)^{j-1} c_j^{(s)} \right) \\
&= \frac{n(n-1)}{2} + \frac{n}{2} \cdot (Wn - n - \varepsilon \sum_{j=1}^{r} (1+\varepsilon)^{j-1} c_j^{(s)}) && \text{(since } (1+\varepsilon)^r = W\text{)} \\
&= \frac{n(n-1)}{2} + \frac{n}{2} \cdot (\text{cost}(\text{MaxST}) + 1 - n) && \text{(by Eq. (6))} \\
&= \frac{n}{2} \cdot \text{cost}(\text{MaxST})
\end{aligned}$$

Thus, $|\widehat{\text{cost}^{(s)}}(G) - \text{cost}^{(s)}(G')| \leq 7\varepsilon \cdot \text{cost}^{(s)}(G')$. Since $|\text{cost}^{(s)}(G') - \text{cost}^{(s)}(G)| \leq \varepsilon \cdot \text{cost}^{(s)}(G)$, we have that

$$\begin{aligned}
|\widehat{\text{cost}^{(s)}}(G) - \text{cost}^{(s)}(G)| &\leq |\widehat{\text{cost}^{(s)}}(G) - \text{cost}^{(s)}(G')| + |\text{cost}^{(s)}(G') - \text{cost}^{(s)}(G)| \\
&\leq 7\varepsilon \cdot \text{cost}^{(s)}(G') + \varepsilon \cdot \text{cost}^{(s)}(G) \\
&\leq 16\varepsilon \cdot \text{cost}^{(s)}(G)
\end{aligned}$$

Replacing ε with $\varepsilon/16$, we get a $(1+\varepsilon)$ estimate of $\text{cost}^{(s)}(G)$.

Running time analysis. since each invocation on CLIQUE-TREE-TRAVERSAL-SIMILARITY takes $\tilde{O}(n/\varepsilon^6)$ time, and we invoke it for $r = O(\log(n/\varepsilon)/\varepsilon)$ times, the total running time is $r \cdot \tilde{O}(n/\varepsilon^6) = \tilde{O}(n/\varepsilon^7)$. □

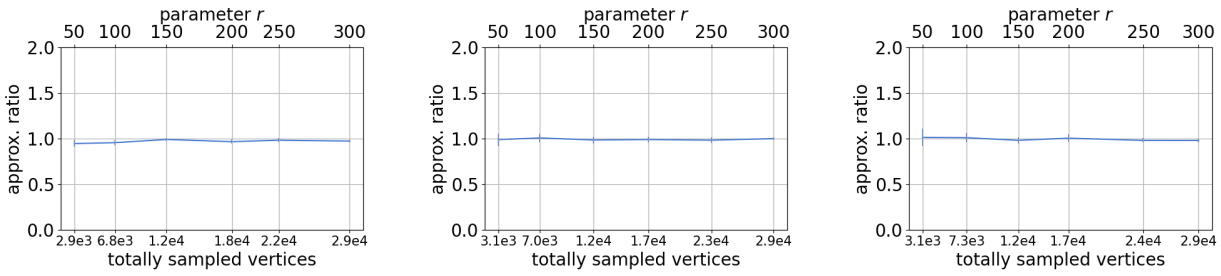
D More on Experiments

D.1 Experiments in Distance Graphs

For road networks, Fig. 3, Fig. 4 and Fig. 5 report the approximation ratio for estimating clustering cost $\text{cost}(G)$ among these datasets; Fig. 7, Fig. 8 and Fig. 9 compare the exact and estimated profile values for different choices of the sample size r. In addition, Fig. 6 presents the approximation ratio for estimating both $\text{cost}(G)$ and profiles on localization datasets.

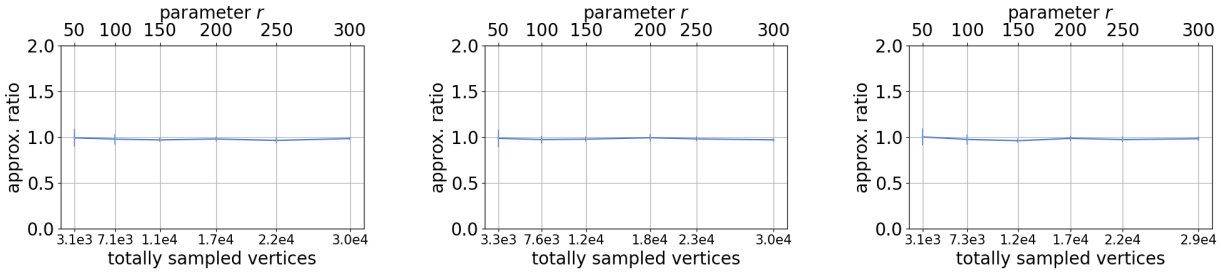
D.2 Experiments in Similarity Graphs

When setting parameters, we choose smaller constants than theory suggests, since real-world datasets often admit good performance with more modest settings; in practice, one could also average results across multiple runs to reduce variance, relaxing the need for per-run accuracy guarantees.



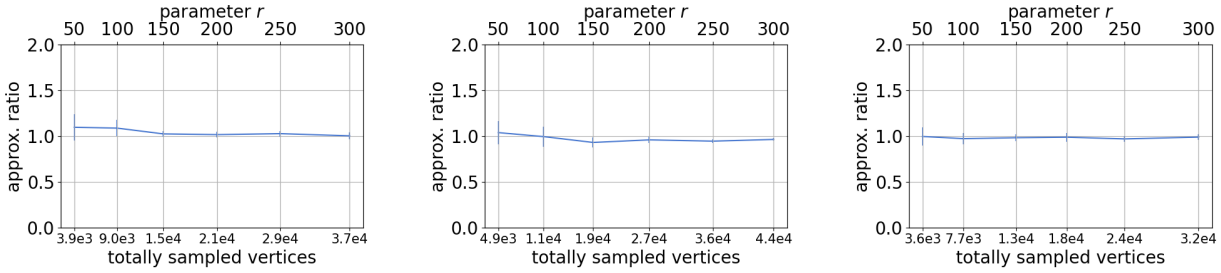
(a) Luxembourg (b) Belgium (c) Netherlands

Figure 3: Approximation ratio in distance graphs: road networks



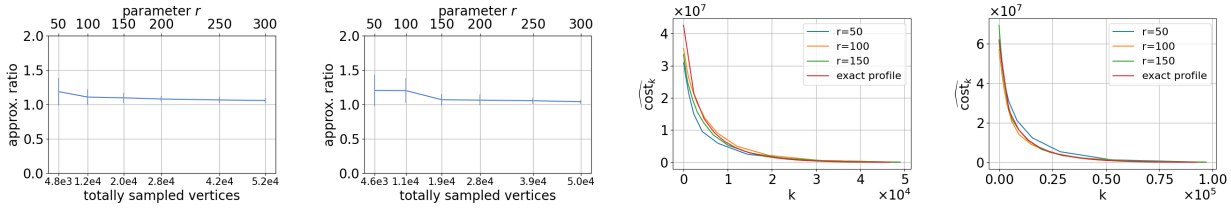
(a) Italy (b) Great Britain (c) Germany

Figure 4: Approximation ratio in distance graphs: road networks



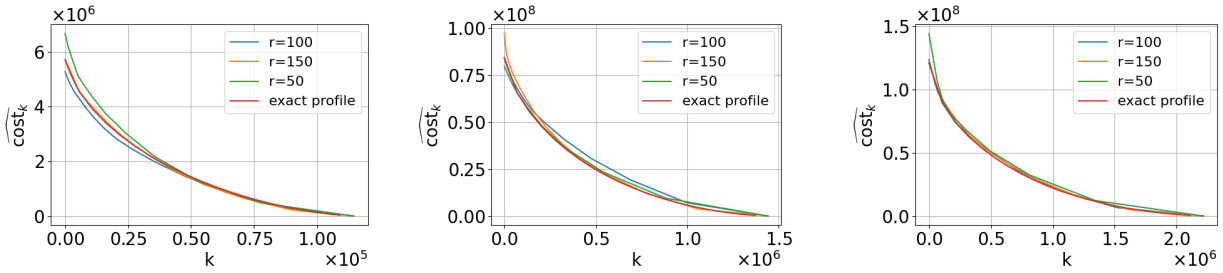
(a) Asia (b) USA (c) Europe

Figure 5: Approximation ratio in distance graphs: road networks



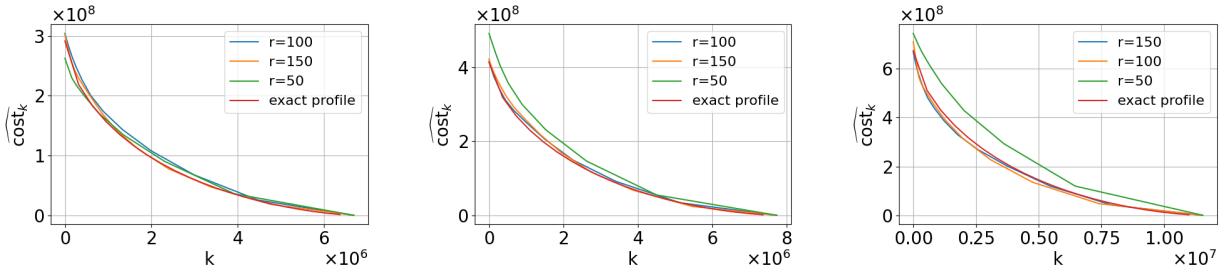
(a) Accuracy of Brightkite (b) Accuracy of Gowalla (c) Profiles of Brightkite (d) Profiles of Gowalla

Figure 6: Approximation ratio and profiles for distance graphs in localization based datasets



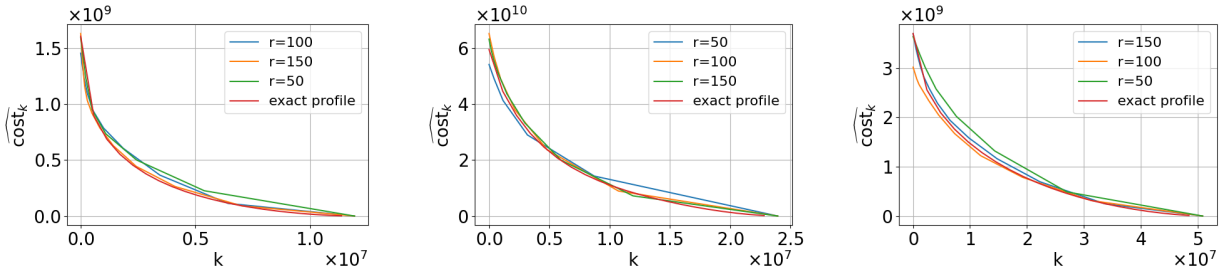
(a) Luxembourg (b) Belgium (c) Netherlands

Figure 7: Profiles for distance case in road networks



(a) Italy (b) Great Britain (c) Germany

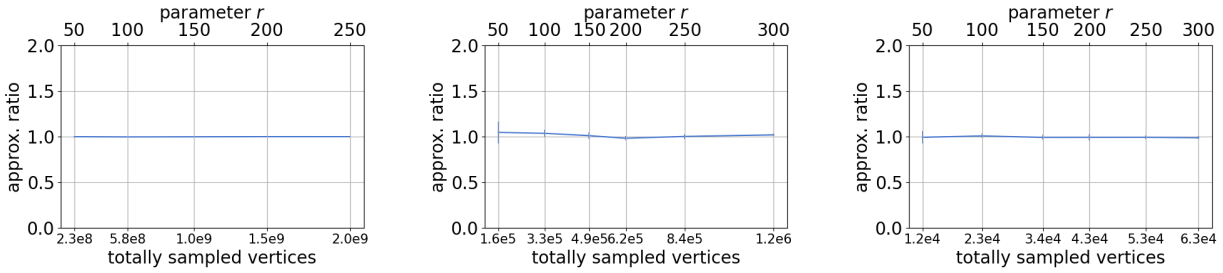
Figure 8: Profiles for distance case in road networks



(a) Asia (b) USA (c) Europe

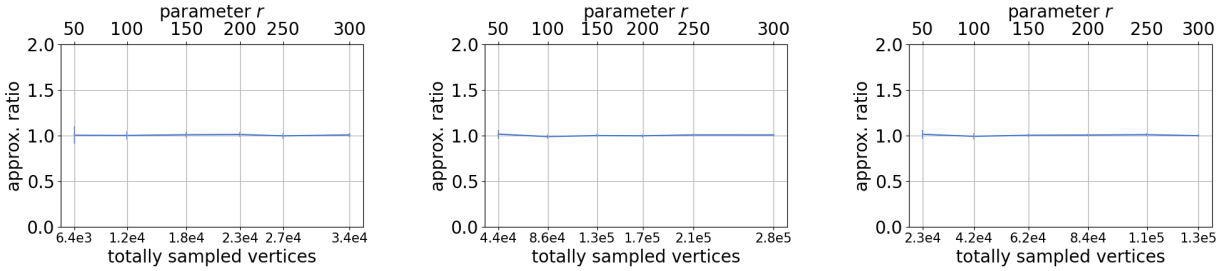
Figure 9: Profiles for distance case in road networks

Specifically, in Algorithm 8, we estimate two types of values: we use Algorithm 1 to estimate $\hat{c}_j$, and use Algorithm 7 to estimate $\overline{D}_j = n - \bar{c}_j$. These two algorithms use different sample sizes r. For the first, we set r to be the input sample size of the code. For the second, we use a larger sample size r'. From theory, $r = O(\frac{1}{\varepsilon^2})$ and $r' = \frac{W}{\varepsilon^2}$, thus $r' = O(r \cdot W)$. In practice, we set $r' = \max\{r \cdot W / \log n, r\}$. Dividing $r \cdot W$ by $\log n$ balances the running time of the two algorithms, and there are at most $O(\log n)$ intervals overall. For the intervals, since $r = O(\frac{1}{\varepsilon^2})$, we set $\varepsilon = \frac{1}{\sqrt{r}}$ and define interval values as in Definition 7.8.



(a) Spotify (b) History (c) Business

Figure 10: Approximation ratio in similarity graphs

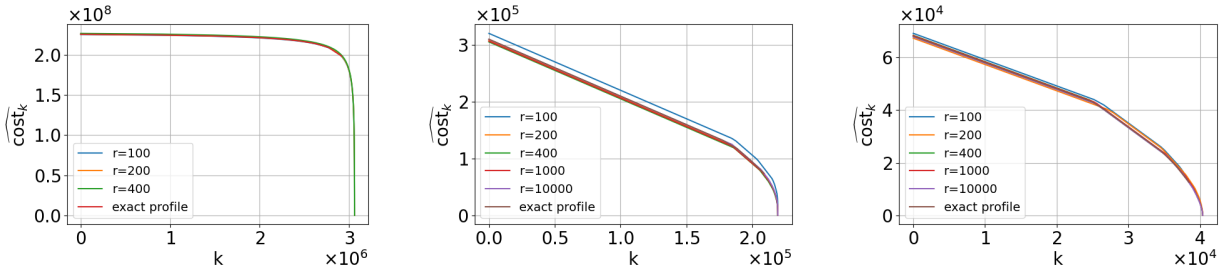


(a) CS (MAG) (b) Geology (c) CS (DBLP)

Figure 11: Approximation ratio in similarity graphs

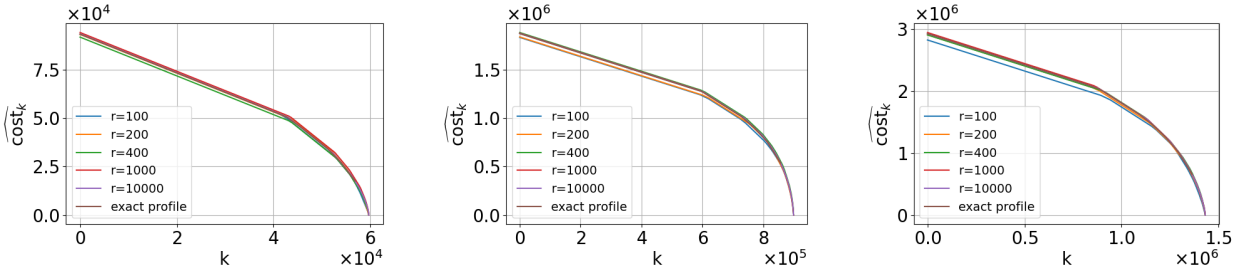
Fig. 10 and Fig. 11 show the approximation ratio for estimating the clustering cost $\text{cost}^{(s)}(G)$. Larger sample size r results in better average approximation ratio and smaller deviation: as r increases, r' also increases, improving the estimates of each $\hat{c}_j$ and $\widehat{D}_m$; moreover, with ε tied to r, the intervals become finer, which improves accuracy for c_j and D_j for $1 \le j \le W$.

We also compute profiles for similarity datasets, as shown in Fig. 12 and Fig. 13, which demonstrate that our algorithm estimates profiles accurately. Fig. 1c presents normalized profiles for similarity datasets: The Spotify curve differs markedly from the co-authorship graphs, indicating that the profile method distinguishes different types of graphs. A higher curve suggests more collaboration; for instance, the Spotify dataset exhibits more 'collaboration' between songs. Note that both *Computer Science* and *DBLP* are co-authorship graphs in the computer science field, but *DBLP* is more complete and thus reveals more structure, including greater collaboration among authors.



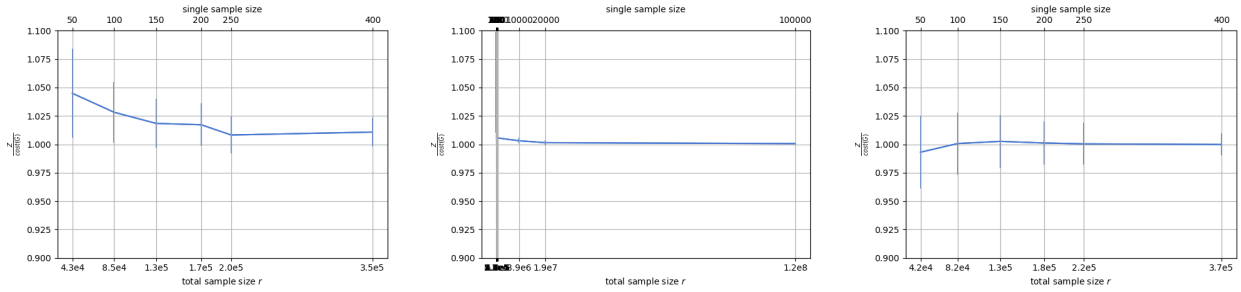
(a) Spotify (b) History (c) Business

Figure 12: Profiles for similarity graphs



(a) CS (MAG) (b) Geology (c) CS (DBLP)

Figure 13: Profiles for similarity graphs



(a) Zoom in on y-axis (b) Extremely large r (c) Set the last interval$= [0,1]$

Figure 14: Bias of co-citation datasets, take Geology as an example

D.3 Discussion on Bias in the Algorithm

Fig. 14a shows that, when the sample size r is small, the approximation ratio begins above 1. But when r becomes very large, Fig. 14b shows that this bias diminishes and the curve converges to 1. This behavior may stem from a characteristic of the similarity datasets, as indicated in Table 3: in the last interval $[0, \frac{\varepsilon n}{W}]$, more D_i values lie below the interval average. Consequently, when r is small, the last interval is too sparse, and D_i values within it are consistently over-estimated, producing the observed bias. Setting the last interval to the smaller range $[0,1]$ removes this effect, as shown in Fig. 14c: the average ratio concentrates around 1,

with deviation on both sides, meaning some experiments under-estimate the cost while others over-estimate it.

Table 3: In the last interval, D_i's distribution

dataset	W	$\#c_j < \frac{\varepsilon n}{W}$	$\#c_j < \frac{\varepsilon n}{W}/2$
History	606	430	329
Geology	192	133	119
CS (MAG)	36	22	19