Hierarchical Agglomerative Clustering via Relaxed Representatives

Eduardo S. Laber
Departamento de Informática
PUC-RIO
Rio de Janeiro, RJ, Brasil
eduardo.laber1@gmail.com

Miguel dos Anjos Batista
Departamento de Informática
PUC-RIO
Rio de Janeiro, RJ, Brasil
miguel260503@gmail.com

Abstract

Due to their simplicity and interpretability, linkage methods are a central class of hierarchical agglomerative clustering (HAC). In particular, representative-based methods, such as centroid and medoid approaches, are appealing because their representatives provide intuitive summaries of the clusters. However, many representative-based linkage methods suffer from important drawbacks, such as high computational cost and the generation of inversions, which limit their broader applicability.

We introduce **relaxed-representative clustering**, a principled generalization of representative-based linkages that preserves their interpretability while addressing these limitations. This framework yields (i) RMM*, a quadratic-time variant of the cubic-time `minimax` method, and (ii) CRM*, an inversion-free variant of the `centroid` method whose representatives belong to the input set. On the theoretical side, we prove that for every k our methods are $O(\min\{k, \log n\})$-approximation algorithms for the k-center objective.

Empirically, experiments on tabular, biological, and textual datasets show that RMM* consistently outperforms several baselines and achieves clustering quality comparable to `minimax` at a fraction of the computational cost. Meanwhile, CRM* remains competitive with the `centroid` method while producing inversion-free dendrograms and interpretable representatives.

1 Introduction

Hierarchical clustering is an important class of clustering methods, with applications in different fields as evolution studies via phylogenetic trees [28], finance [63] and detection of closely related entities [41, 50]. Given a set $\mathcal{X}$ of n points, a hierarchical clustering is a sequence of n clusterings in which the first clustering has n unitary clusters, each of them corresponding to a point in $\mathcal{X}$, and the ith clustering is obtained from the $(i-1)$th-clustering by replacing two of its clusters with their union. A hierarchical clustering induces a strictly binary tree with n leaves, where each leaf corresponds to a point in $\mathcal{X}$ and each internal node corresponds to a cluster. This tree (a.k.a dendrogram) is useful to visualize potential interesting clustering.

Linkage methods are a family of Hierarchical Agglomerative Clustering (HAC) methods that are often taught in data science/ML courses and are implemented in many machine learning libraries (e.g. `scipy`). At each step, a linkage function is employed to measure the distance between different clusters, and the two with minimum distance are merged.

Representatives, as centroids and medoids, are widely used in clustering algorithms, since they can aid in the interpretation of the resulting clusters [60, 65, 47, 38, 12]. A notable example of this class is the Lloyd k-means algorithm [47]. Several well-known linkage methods, including `Ward`

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

and `centroid`, are also based on representatives [65, 60, 12, 57]. However, some of them, as we exemplify below, present drawbacks that limit their broader applicability.

Our first example is the `centroid` method [60], a popular linkage algorithm that, at each iteration, merges the two clusters whose representatives are closest. Despite its simplicity, it has two well-known drawbacks: its representatives are not necessarily points from the input space, and it may produce *inversions*, both of which undermine the interpretability of the resulting clusters and dendrogram.

Our second example is the `minimax` algorithm [12], a variant of `complete-linkage` that greedily minimizes the distance of a point to its closest representative. For every k, it produces a k-clustering whose maximum diameter is an $O(\log(n-k))$ factor from the minimum possible one [6]. Its major drawback is that no subcubic implementation is known for it.

1.1 Contributions

We address these limitations by introducing the notion of *relaxed representatives*. This concept leads to a new family of HAC algorithms where the representative of a merged cluster is inherited from one of its constituent clusters, rather than being recomputed. This design yields efficient algorithms that remain faithful to the hierarchical structure, avoid costly computations, and retain interpretability.

We make the following contributions:

- **Conceptual novelty.** We introduce the framework of relaxed representatives, which provides a simple but powerful principle for designing hierarchical clustering algorithms.
- **Algorithmic contribution.** We introduce simple linkage methods with different motivations. Relaxed Minimax (`RMM`*) targets compact clusters on moderate-size datasets. In contrast, Closest-Relaxed-Medoids (`CRM`*), is an alternative for the `centroid` method [60] that offers three advantages: it applies to general dissimilarities, does not produce inversions, and scales well on large sparse graph instances.
- **Theoretical guarantees.** We show that the proposed methods achieve bounds on the maximum diameter and k-cost center cost that are linear in k and logarithmic in n, while running in quadratic time. To the best of our knowledge, no other known quadratic-time linkage algorithm has these guarantees.
- **Practical performance.** Our experiments on synthetic and real datasets — including biological and textual data — indicate that `RMM`* is a strong choice for compact hierarchical clustering with high-quality representatives under the k-center objective on datasets of moderate size ($\approx 10^4$ points). They also indicate that `CRM`* is an appealing alternative for those seeking an inversion-free variant of the `centroid` method.

Theoretical Analyses. To formally state our contributions, we introduce some basic notation. Let $(\mathcal{X}, \texttt{dist})$ be a metric space, where $\mathcal{X}$ is a set of n points and `dist` is the distance function. The diameter of a set S is the maximum distance between two of its points. For a clustering $\mathcal{C}$, we define its maximum diameter as the largest cluster diameter in $\mathcal{C}$, and its average diameter as the mean cluster diameter. We write $\texttt{OPT}_{DM}(k)$ and $\texttt{OPT}_{AV}(k)$ for the minimum possible maximum and average diameters, respectively, over all k-clusterings of the data.

We establish the following results for the algorithms proposed here:

- **Complexity.** All algorithms run in quadratic time, which is optimal under the assumption that all distances must be read. Moreover, for datasets represented by sparse graphs with weighted edges `CRM`* admits an $O(m \log n + n \log^2 n)$ time implementation, where n and m are the number of nodes and edges in the graph;
- **Maximum and average diameter.** For every k, their k-clustering has maximum diameter $O(\min\{k, \log n\}) \cdot \texttt{OPT}_{DM}(k)$ [1]. This bound improves the state-of-the-art since no known linkage method achieves it in $O(n^2)$ time, and, notably, it provides an exponential gain (in terms of n) over the known bounds for classical linkage methods (single, complete and average). For `CRM`*, we also show that the average diameter of its k-clustering is $O(k \cdot \texttt{OPT}_{AV}(k))$, for every k.

[1] All the logarithms are base 2

Table 1: Comparison of running times and available bounds on the maximum diameter of a k-clustering

Method	**Bounds depending on k**	**Bounds depending on n**	**Time**
`single`	$\Theta(k)$	$\Theta(n)$	$O(n^2)$
`complete`	$\Omega(k)$; $O(k^{1.30})$	$\Omega(\log n)$; $O(n)$	$O(n^2)$
`average`	$\Omega(k)$ and $O(k^{2.59})$	$\Omega(\log n)$; $O(n)$	$O(n^2)$
`minimax`	$\Theta(k)$	$\Theta(\log n)$	$O(n^3)$
`CRM`*	$O(k)$ (this paper)	$O(\log n)$ (this paper)	$O(n^2)$
`RMM`*	$O(k)$ (this paper)	$O(\log n)$ (this paper)	$O(n^2)$

Table 1 summarizes the best known bounds on the maximum diameter for widely used linkage methods [22, 6, 21, 43]. In terms of efficiency and diameter-based compactness criteria, our algorithms strictly dominate these methods by improving upon each of them in at least one of these criteria while never being worse in the other.

We note that all the above bounds also hold for the popular k-center cost function that measures the maximum distance from a point to its closest representative [31] (see Section 2 for a discussion). That said, we decided to present our theoretical bounds in terms of diameter-related criteria, aiming for a more direct comparison with the available literature.

Importantly, the assumption that the points lie in a metric space is employed solely for the derivation of the approximation bounds, as our proposed methods also run in quadratic time for more general spaces.

Practical Performance. To assess the performance of our proposed methods, we carried out an experimental study in which we compared them with the methods of Table 1, the `centroid` method and methods from [31, 22, 27, 14]. We used both synthetic and real datasets across various domains, such as biology, ratings and text, and considered several clustering criteria. Our main findings are the following:

- `RMM`* consistently produces clusters that are more compact than all methods but `minimax`. While it does not outperform `minimax` it produces close results, and it is around 100 times faster for several datasets of moderate size.
- `CRM`* is competitive with the `centroid` method regarding robustness to noise and Adjusted Rand Index (ARI). Significantly, while our method is free of inversions, `centroid` may produce a large number for some datasets.

Motivation for diameter and k-center cost. The diameter and the k-center cost, the criteria that we employ to analyse our methods, are natural choices to capture compactness that have been widely studied in the literature [31, 22, 17, 5, 11, 29]. Worst-case coverage objectives closely related to k-center also arise in modern embedding-based learning pipelines. For example, [58] formulate batch active learning as a core-set selection problem and employ a greedy k-center method in feature space to select representative labeled examples.

Moreover, they are also used in applications. As an example, in HIV epidemiology, `Cluster Picker` [56] identifies transmission clusters from phylogenetic trees by requiring both a minimum branch support and a maximum pairwise genetic distance threshold. The latter condition directly enforces a bound on the maximum cluster diameter in terms of genetic divergence.

Limitations. While `RMM`* runs in $O(n^2)$ time, it also requires quadratic space in the worst case, which limits its applicability to very large datasets. `CRM`* mitigates this issue since it admits an implementation that operates directly on graph representations, which allows it to scale to larger instances.

1.2 Related Work.

There is a vast body of literature on hierarchical agglomerative clustering methods, including surveys and books dedicated to the topic [53]. In what follows, we review works that use representatives,

establish provable guarantees for well-known linkage methods, or analyze guarantees for the k-center objective.

Methods based on representatives. Linkage methods for HAC based on representatives, as `centroids` [60], `Ward` [66], `medoids` [35], `minimax` [12], `min-sum` [57] or `min-sum-increase` [57] have been widely studied. Some of the available methods are not suitable for general metric spaces (e.g. `centroids` and `Ward`), while others, as `medoids` and `minimax`, require cubic time (no subcubic implementations are known). Our methods are suitable for general metric spaces, run in $O(n^2)$ time and have provable guarantees.

`CRM`* is closely related to the well-known `centroid` method [60] and its variants, as `medoid` [35] because at each step, it merges the clusters that have the closest representatives. However, unlike `medoid` and `centroid` methods, it does not produce inversions (Theorem 3.1), which makes the resulting dendrograms much more interpretable for hierarchical clustering analysis.

For the case where the n input points lie in $\mathbb{R}^d$, Bateni et. al [9] (Theorem 5) presents an $O(n^{1+O(1/c)}d^2)$-time implementation of a variant of the `centroid` method in which, at each iteration, the algorithm merges a pair of clusters whose centroids are within a factor c of the minimum centroid distance, rather than the closest pair. Unlike our approach, their relaxation is applied to the merge criterion, rather than to the representative selection, which leads to different structural properties of the resulting dendrogram, including the presence of inversions.

`RMM`* can be seen as a variant of `minimax` [12], where relaxed representatives are used to reduce the cubic running time of the latter to quadratic time.

Theoretical bounds for k-center objective. There is also a line of works that gives bounds on cohesion criteria of the k-clustering produced by well-known linkage methods [22, 6, 21, 43]. Table 1 shows the main bounds on the maximum diameter in metric spaces. For the case in which `dist` is a metric induced by a norm in $\mathbb{R}^d$ and d is constant, [3, 32] give an $O(\log k)$ upper bound on the maximum diameter of `complete-linkage`. Bounds for other objectives are discussed in Section A.

When we turn to non-linkage methods, [31] proposes a 2-approximation algorithm for flat clustering, which is the best possible approximation under the assumption that $P \neq NP$. For hierarchical clustering [22] shows how to adapt the method from [31] to obtain an 8-approximation algorithm, that is, for every k, the k-clustering built by this algorithm is an 8-approximation. A similar algorithm was independently proposed by [17]. While these methods have stronger theoretical guarantees than ours, they do not yield interpretable dendrograms. Furthermore, our empirical results indicate that `RMM`* outperforms them in terms of the k-center objective.

Scalable Linkage-Methods. Our work, more specifically, the implementations of `CRM`* for sparse graphs, is related to the line of work that develops implementations for classical linkage methods, and variants, for large and sparse graph datasets [23, 24, 25].

2 Preliminaries

A pseudo-code for the class of HAC based on relaxed representatives is shown in Algorithm 2. At each iteration, the two closest clusters in terms of distance `dist` are merged (line 4 and 5), and the representative of one of them becomes the representative of the resulting cluster (line 6). This last line distinguishes linkage methods based on relaxed representatives from standard ones. Note that the pseudo-code uses c_A and c_B to refer to the representatives of A and B, respectively.

Time complexity of linkage methods. The following theorem from [54] will be employed to analyse the time complexity of the algorithms proposed in this paper.

Algorithm 2 Linkage methods based on relaxed representatives.

1: `Active` $\leftarrow$ set of n unitary clusters, each one containing a point of $\mathcal{X}$
2: **For** each cluster C in `Active`, the representative of C is its only point
3: **For** i to $n-1$
4: $\quad(A, B) \leftarrow$ clusters in `Active` for which `dist`(A, B) is minimum
5: $\quad$`Active` $\leftarrow$ `Active` $- \{A\} - \{B\} \cup \{A \cup B\}$
6: $\quad$Use some criterion to decide the representative of $A \cup B$ between c_A and c_B

Theorem 2.1. *[[54]] Let* Link *be a linkage method that uses* dist *as its linkage function. If for all disjoint clusters A, B and C,* $\texttt{dist}(A \cup B, C)$ *can be computed in $O(T)$ time (given* $\texttt{dist}(A, C)$ *and* $\texttt{dist}(B, C)$*), and* dist *satisfies properties (i) and (ii) below, then* Link *can be implemented in $O(n^2 \cdot T)$ time via the* NN-chain *algorithm [52]:*

(i) For every three disjoint clusters A, B and C, $\texttt{dist}(A, B) \leq \min\{\texttt{dist}(A, C), \texttt{dist}(B, C)\}$ *implies* $\min\{\texttt{dist}(A, C), \texttt{dist}(B, C)\} \leq \texttt{dist}(A \cup B, C)$

(ii) for all disjoint clusters A, B, C and D, at any stage of the clustering. the value $\texttt{dist}(A \cup B, C \cup D)$ *does not change if A and B are merged before C and D or vice-versa.*

All the methods we consider here satisfy (ii), so we just need to guarantee that (i) is satisfied.

Additional notations and definitions. In many passages, we mention that two representatives are merged, indicating that the clusters they represent are merged. We say that a cluster C is *active* at some point of the execution of a linkage algorithm if C belongs to the set Active (Algorithm 2).

Recall that we use $\texttt{diam}(S)$ to denote the diameter of a set of points S. Moreover, we use max-diam and avg-diam to denote, respectively, the maximum and the average diameter of the k clusters in a k-clustering $\mathcal{C}$, that is,

$$\texttt{max-diam}(\mathcal{C}) := \max\{\texttt{diam}(C) | C \in \mathcal{C}\} \text{ and } \texttt{avg-diam}(\mathcal{C}) := \frac{1}{k} \sum_{C \in \mathcal{C}} \texttt{diam}(C).$$

Given a point p and a cluster C, we define $\texttt{dmax}(p, C) := \max\{\texttt{dist}(p, x) | x \in C\}$. We say that a point c in a cluster C is its *center* if and only if $\texttt{dmax}(c, C) \leq \texttt{dmax}(c', C)$, for every $c' \in C$.

The cost of a cluster A, denoted by $\texttt{cost}(A)$, is the maximum distance from a point of A to its representative. We only use this notation when the representative is clear from the context. The cost of A and its diameter are related as stated in the following proposition (see Appendix B for a proof):

Proposition 2.2. *For every cluster A, the following holds:* $\frac{1}{2}\texttt{diam}(A) \leq \texttt{cost}(A) \leq \texttt{diam}(A)$.

The k-center cost function. This is a classical objective in the study of flat clustering [31]. Given a k-clustering $\mathcal{C} = (C_1, \ldots, C_k)$, with representatives $c_1, \ldots, c_k$, its k-center cost is defined as the maximum cost among the clusters of $\mathcal{C}$, that is, $\max\{\texttt{cost}(C_i) | C_i \in \mathcal{C}\}$.

Hence, as a consequence of Proposition 2.2, the k-center cost of $\mathcal{C}$ is between $\texttt{max-diam}(\mathcal{C})/2$ and $\texttt{max-diam}(\mathcal{C})$, so that every approximation regarding the maximum diameter is also an approximation to the k-center cost function up to a factor of 2.

Dendrograms and inversions. Linkage methods produce dendrograms, a widely employed tool to analyze hierarchical clustering. A dendrogram is a binary tree that keeps track of the merges performed along the algorithm's execution; that is, if clusters A and B are merged, then a node corresponding to $A \cup B$ becomes the parent of the nodes corresponding to A and B. Importantly, each internal node in the tree is associated with a height, which is the distance between its children, that is, the height of node $A \cup B$ is $\texttt{dist}(A, B)$.

We say that an *inversion* occurs if for some i, with $1 \leq i \leq n - 2$, the height of the node produced by the ith merge is larger than that of the node produced by the $(i + 1)$th merge. Inversions create a serious obstacle for the interpretation of dendrograms, so linkage algorithms that do not produce inversions are highly desirable.

3 Algorithms

3.1 The CRM* algorithm

CRM* can be seen as a member of a class of linkage algorithms, denoted by Closest Relaxed Representatives ($\mathcal{CRR}$), that follow the pseudo-code of Algorithm 2 to build the clusterings and use $\texttt{dist}(c_A, c_B)$ as the distance between clusters A and B, where c_A and c_B are the representatives of A and B, respectively. The motivation of this rule is to prevent clusterings that have clusters with close representatives.

What distinguishes the methods in $\mathcal{CRR}$ is how the representative for $A \cup B$ is selected when A and B are merged. To decide the representative of $A \cup B$, $\texttt{CRM}^*$ maintains a computationally cheap upper bound $\texttt{ucost}$ on the cost of each active cluster. Initially, $\texttt{ucost}(C) = 0$ for each unitary cluster C. After merging A and B, $\texttt{CRM}^*$ set c_A as the representative of $A \cup B$ if $\texttt{ucost}(A) \geq \texttt{ucost}(B)$, and set c_B, otherwise. In the first case, $\texttt{ucost}(A \cup B)$ is set to $\max\{\texttt{ucost}(A), \texttt{dist}(c_A, c_B) + \texttt{ucost}(B)\}$, while in the second is set to $\max\{\texttt{ucost}(B), \texttt{dist}(c_A, c_B) + \texttt{ucost}(A)\}$. This rule aims to obtain compact clusters at a low computational cost, being suitable for a scalable implementation.

After building the dendrogram, $\texttt{CRM}^*$ computes the centers of the $n - 1$ clusters formed along the construction. This last phase can be implemented in $O(n^2)$ time, as we show in Section F,

The algorithms in $\mathcal{CRR}$ are closely related to the `centroid` linkage method [60]. However, unlike this method, algorithms in $\mathcal{CRR}$ do not produce inversions, as stated in the following theorem (proof in Appendix C.1) .

Theorem 3.1. *If $\mathcal{A}$ is a method in $\mathcal{CRR}$, then it does not produce inversions.*

Regarding the time complexity of the algorithms in $\mathcal{CRR}$, we first note that they satisfy property (i) of Theorem 2.1 because

$$\min\{\texttt{dist}(A, C), \texttt{dist}(B, C)\} = \min\{\texttt{dist}(c_A, c_C), \texttt{dist}(c_B, c_C)\} \leq \texttt{dist}(A \cup B, C),$$

where the inequality holds because the representative of $A \cup B$ is either c_A or c_B, so that $\texttt{dist}(A \cup B, C)$ is either $\texttt{dist}(c_A, c_C)$ or $\texttt{dist}(c_B, c_C)$.

Thus, methods in $\mathcal{CRR}$ can be implemented in $O(n^2 T')$ time, where T' is the time to choose representative of $A \cup B$ between c_A and c_B. In the case of $\texttt{CRM}^*$, T' is $O(1)$, so it can be implemented in $O(n^2)$ time.

Scalable Implementation. A relevant property of $\texttt{CRM}^*$ is that it allows a scalable implementation for graph datasets. In these datasets, each vertex corresponds to a point, and each edge is associated with the distance between its two endpoints. Pairs of points that are not neighbors are assumed to have a distance K (e.g. 1) that is larger than any other distance. This assumption is natural in settings where the graph encodes a sparse similarity structure, such as k-nearest neighbor graphs or neighborhood-based similarities (e.g., Jaccard), where non-adjacent vertices are maximally dissimilar by construction.

The implementation of $\texttt{CRM}^*$ for graphs processes the edges in non-decreasing order of the distances between their endpoints. Whenever the current edge connects two vertices that are active representatives of distinct clusters, the corresponding clusters are merged. If at least one endpoint is no longer an active representative, the edge is discarded.

We note that at the end of the procedure, multiple clusters may remain active. We can complete the dendrogram by arbitrarily merging these clusters. This is in line with the procedure's rule of considering the edges by non-decreasing order of distances since, by assumption, all pairs of points without an edge have a distance K, larger than any other distance. Sorting the edges dominates the construction of the dendrogram, incurring $O(m \log m)$ time, where m is the number of edges of the graph.

After building the dendrogram, the procedure computes the centers of the $n - 1$ clusters using the method from Section F.2. This step runs in $O(n \log^2 n + m \log n)$ time and dominates the overall complexity.

In the remainder of this section, we present approximation bounds for algorithms in $\mathcal{CRR}$ and, in particular, for $\texttt{CRM}^*$. We note that Theorems 3.5 only holds for $\texttt{CRM}^*$.

Approximation Bounds. We start with a simple property that bounds the distance between representatives merged by algorithms in $\mathcal{CRR}$. The proof is in Section C.2.

Proposition 3.2. *Let $\mathcal{A}$ be an algorithm in $\mathcal{CRR}$ and let $t \leq n - k$. If two representatives c and c' are merged at iteration t of $\mathcal{A}$, then $\texttt{dist}(c, c') \leq \texttt{OPT}_{DM}(k)$*

Let $\mathcal{A}$ be an algorithm in $\mathcal{CRR}$, and let $\mathcal{C}^* = (C_1^*, \ldots, C_k^*)$ be a k-clustering that will be chosen according to our goal. We denote it by *reference clustering*. We say that a point $p \in C_i^*$ is a *survivor* if it is the last point in C_i^* that survives as a representative during $\mathcal{A}$'s execution, that is, at some iteration it is the only point in C_i^* that is still a representative. Clearly, each cluster in $\mathcal{C}^*$ can have at most one survivor, so the number of survivors is upper bounded by k.

The next lemma bounds the cost of a cluster C built by $\mathcal{A}$ as a function of a reference clustering $\mathcal{C}^*$, and the number of survivors of C. It will be used twice in our analyses.

Lemma 3.3. *Let $\mathcal{C}^* = (C_1^*, \ldots, C_k^*)$ be a reference k-clustering. Moreover, let $\mathcal{A}$ be an algorithm in $\mathcal{CRR}$ and let C be a cluster built by $\mathcal{A}$ in the first $n-k$ iterations. Then,*

$$\texttt{cost}(C) \leq \left(2\sum_{i=1}^{k} \texttt{diam}(\mathcal{C}_i^*)\right) + s(C) \cdot \texttt{OPT}_{DM}(k). \tag{1}$$

where $s(C)$ is number of survivors of C with respect to $\mathcal{C}^*$.

To prove the lemma, we first define a directed tree that keeps track of the merges performed along the execution of $\mathcal{A}$. Each node in the tree corresponds to a point, and there is an edge from u to v if a cluster represented by u is merged with one represented by v, with u becoming the representative of the resulting cluster

For the upper bound on $\texttt{cost}(C)$, let c be the representative of C and let x be the farthest point from c among those in C, so that $\texttt{cost}(C) = \texttt{dist}(c, x)$. In general lines, the proof proceeds by examining the sequence of cluster representatives that appears along the path P in the tree connecting c to x. We carefully compress this sequence so that each cluster in $\mathcal{C}^*$ contributes to it with at most two points. Finally, we use the triangle inequality to bound $\texttt{cost}(C)$ by the sum of the distances between consecutive points in the compressed sequence. If a point p in this sequence is a survivor, we bound the distance between p and its successor using Proposition 3.2. Otherwise, the distance is bounded by the diameter of the cluster in $\mathcal{C}^*$ where p lies.

The next theorem gives upper bounds on $\texttt{max-diam}(\mathcal{C})$ and $\texttt{avg-diam}(\mathcal{C})$.

Theorem 3.4. *Let $\mathcal{A}$ be an algorithm from class $\mathcal{CRR}$ and let $k \geq 2$. The k-clustering $\mathcal{C}$ built by $\mathcal{A}$ satisfies* $\texttt{max-diam}(\mathcal{C}) \leq 6k\texttt{OPT}_{DM}(k)$ *and* $\texttt{avg-diam}(\mathcal{C}) \leq 6k\texttt{OPT}_{AV}(k)$.

To prove the bound on the maximum diameter, we use Lemma 3.3, with the reference clustering being the one with `max-diam` equal to $\texttt{OPT}_{DM}(k)$. In addition, we use the fact that the number of survivors is at most k. For the bound on the average diameter, we proceed similarly, but employing the k-clustering with average diameter $\texttt{OPT}_{AV}(k)$ as the reference clustering.

The following theorem shows that $\texttt{CRM}^*$ has an $O(\log n)$ approximation to the diameter.

Theorem 3.5. *For every k, the k-clustering $\mathcal{C}$ built by* $\texttt{CRM}^*$ *satisfies* $\texttt{max-diam}(\mathcal{C}) \leq O(\log n \texttt{OPT}_{DM}(k))$.

We give a brief overview of the proof. First, we prove that for every cluster C, $\texttt{ucost}(C)$ is an upper bound on $\texttt{cost}(C)$, and as a consequence of Proposition 2.2, an upper bound on $\texttt{diam}(C)/2$.

Then, we show by induction on the number of iterations that the `ucost` of every cluster C built by $\texttt{CRM}^*$ is at most $\texttt{OPT}_{DM}(k)\log(|C)|$. Let A and B be the clusters merged to produce C, with c_A being the representative of C. In this case, $\texttt{ucost}(C) = \texttt{ucost}(A)$ or $\texttt{ucost}(C) = \texttt{ucost}(B) + \texttt{dist}(c_A, c_B)$. To prove the inductive step, we use the inductive hypothesis on A, in the first case, while in the second we use Proposition 3.2 to bound $\texttt{dist}(c_A, c_B)$, together with the inductive hypothesis on the smaller cluster between A and B.

Examples showing the tightness of the bounds for $\texttt{CRM}^*$ can be found in the Appendix D.

3.2 The $\texttt{RMM}^*$ algorithm

The `Relaxed Minimax` ($\texttt{RMM}^*$) follows the pseudo-code of Algorithm 2. To measure the distance between clusters A and B, with representatives c_A and c_B, it employs the linkage function $\texttt{dist}(A, B) := \min\{\texttt{dmax}(c_A, A \cup B), \texttt{dmax}(c_B, A \cup B)\}$. If $\texttt{dmax}(c_A, A \cup B) \leq \texttt{dmax}(c_B, A \cup B)$, c_A becomes the representative of $A \cup B$, otherwise the representative becomes c_B.

As $\texttt{CRM}^*$, after building the dendrogram, $\texttt{RMM}^*$ computes the centers of the $n-1$ clusters formed along the construction in $O(n^2)$ time.

In Appendix E.1 we prove that $\texttt{RMM}^*$ does not produce inversions, while in Appendix E.2 we argue that $\texttt{RMM}^*$ admits an $O(n^2)$ time implementation. To establish this time complexity, we show that

it satisfies property (i) of Theorem 2.1 and that during its execution, it can compute the distance between clusters in $O(1)$ time.

Theorem 3.6, whose proof can be found on the appendix, provide bounds that depend on $\texttt{OPT}_{DM}(k)$. The proof strategy resembles that of [6].

Theorem 3.6. *The k-clustering $\mathcal{C}$ built by* `RMM`* *satisfies* $\texttt{max-diam}(\mathcal{C}) \leq (8k+2)\texttt{OPT}_{DM}(k)$ *and* $\texttt{max-diam}(\mathcal{C}) \leq (4\lceil \log(n-k) \rceil + 2)\texttt{OPT}_{DM}(k)$.

4 Experiments

In this section, we present our main experiments. We employed several datasets across different domains, to evaluate `CRM`* and `RMM`* against `minimax` [12], `centroid`, `complete-linkage`, `average-link`, `single-linkage`, `HDBSCAN`[14], `Dasg`: the 8-approximation hierarchical algorithm for k-center proposed in [22], and `ship-hst`[27], a modern method that embeds the original points in a tree metric (HST) with provable $O(\log n)$ distortion, and then optimally solves the k-center problem for the embedding points. We use the k-center objective (the primary criterion) and ARI (secondary criterion) to assess clustering quality. For the algorithms that do not provide representatives from the input set, we consider the center of a cluster as its representative, so the cost of the clusters produced by these methods becomes well-defined. Additional details and other experiments can be found in Appendix G.

Running time of linkage methods. The lines of Table 2 show the running times of the methods for the datasets in which the quadratic time algorithms took more than 1 second. The reported times include both the computation of the dendrogram and the subsequent determination of the cluster representatives. All methods yield competitive results, except for `minimax`, which is around 100 times slower on some datasets (red numbers), consistent with its $O(n^3)$ time complexity. As discussed in Appendix G.3, the running times reported in this table are stable, exhibiting low variance.

We omit the running times for the `centroid` method from Table 2. Since the `centroid` linkage produces inversions, it cannot be implemented using the nearest-neighbor chain algorithm, making a direct runtime comparison with the other methods uninformative. For the remainder of the paper, `centroid` results are obtained using the `sklearn` implementation.

To study how `CRM`* scales to large graph datasets, we consider vertex-induced subgraphs of the `Orkut` graph [68], obtained by uniformly sampling vertices at varying sizes. The full dataset contains 3.1×10^6 nodes and 1.17×10^8 edges. For each sample, we run the graph versions of `CRM` and `CRM`* and measure the running time. The density m/n^2 of each of these samples is $\approx 10^{-5}$, so the graphs are very sparse. The results are presented in Figure 1, where the curve `CRM` corresponds to the time required to build the dendrogram. More details about the construction of the dataset can be found at the end of Appendix G.1.

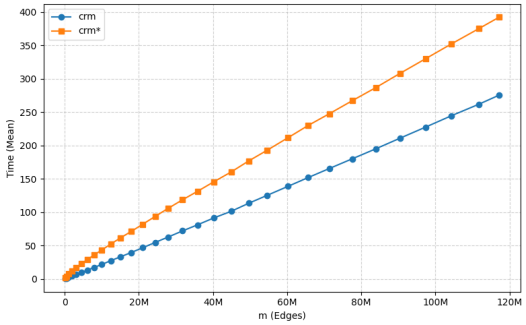


Figure 1: Running time in seconds of `CRM`* and the dendrogram construction (`CRM`) for samples of `Orkut` dataset as a function of the number of edges.

k-center cost. Figure 2 shows, for a subset of datasets selected to reflect different domains and performance profiles, the performance of the algorithms on the k-center cost, as a function of k. The point associated with k (horizontal axis) for a method M represents the ratio between the k-center cost of the k-clustering built by M and that of the k-clustering built by `minimax`. We choose `minimax` as

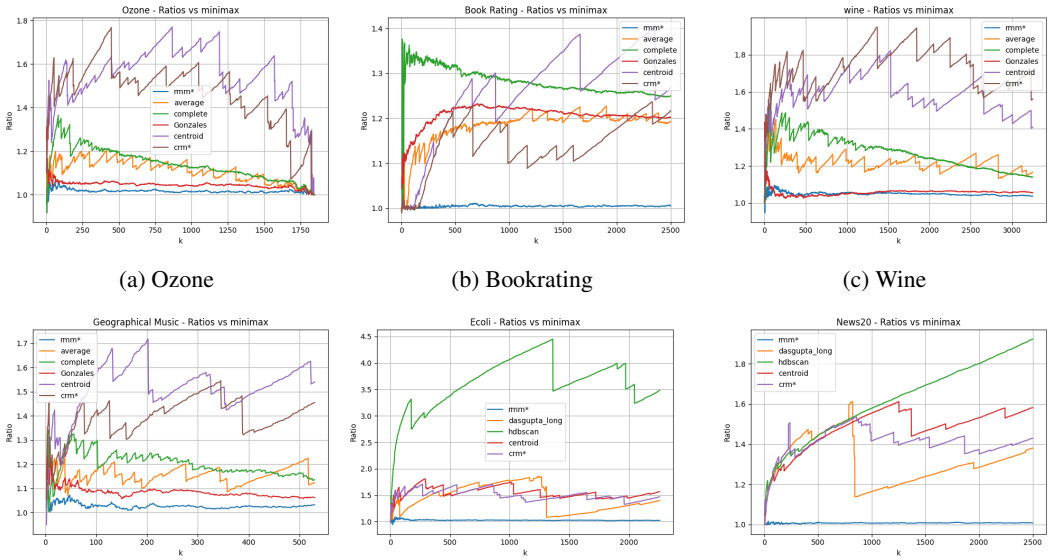
Table 2: Running times across datasets for the evaluated methods.

Dataset	Average	Complete	CRM*	Minimax	RMM*	Single
Allen Brain	5.41	6.46	6.36	64.89	7.85	6.23
Anuran	9.95	11.14	10.58	39.45	14.10	10.73
BookRat.	5.71	5.81	6.83	1075.38	8.45	6.37
Ecoli	4.16	4.58	4.82	13.05	5.93	4.76
MovieLens	8.59	8.63	9.91	1214.91	12.36	9.27
News20	5.67	5.87	6.29	638.48	8.23	6.41
Ozone	2.43	2.90	3.49	5.98	3.83	3.85
Parkinsons	6.63	7.40	8.30	22.05	9.38	8.81
Reuters	5.48	5.83	6.37	41.66	8.03	6.01
Rice	3.09	4.11	5.25	8.12	5.64	4.95
Wine	8.69	9.81	9.89	27.62	12.26	10.17

a reference because it produced the best results for this criterion. In this experiment, we also add the results of `Gonzalez`, a 2-approximation efficient flat-clustering algorithm proposed in [31].

The results of `RMM`* are clearly superior to those of the other subcubic-time hierarchical methods and close to those of `minimax`, which runs in cubic time. Although `Gonzalez` is not constrained to be hierarchical, `RMM`*, perhaps surprisingly, outperforms it on most $(\text{dataset}, k)$ configurations (e.g. 83% when $k \leq 100$).

In Appendix G.4, we provide these graphs for all datasets, and also include `Dasg`, `ship-hst` and `HDBSCAN`. These methods do not achieve competitive performance.



(a) Ozone (b) Bookrating (c) Wine

(d) Geographical Music (e) Ecoli (f) Newsgroups

Figure 2: Ratio between the k-center cost of each method and that of `minimax` for each value of k.

We conducted an additional experiment comparing `CRM`* against the variant of average-link implementation for sparse graphs [25] on the Condensed Matter collaboration network dataset, consisting of 23.133 authors and 93.497 coauthorship edges. From this data, we construct a graph in which vertices correspond to authors and an edge connects two authors that have at least one common coauthor. The resulting graph has 23,133 vertices and 1,166,152 edges. The distance between two authors u and v is defined as the Jaccard distance between their sets of coauthors. As expected, `average-link` produces clusters with a smaller k-center objective. This is consistent with the results reported in Figure 2, where `average-link` also outperforms `CRM`* in terms of compactness.

ARI and other analyses. We evaluated the ARI of the methods on 9 classification datasets, using k as the number of classes. The results are presented in Table 3. We observe that `RMM`* achieves some

Table 3: ARI for classification datasets.

dataset	k	average	complete	RMM*	CRM*	minimax	single	centroid
beans	7	0.32	0.42	**0.55**	0.11	0.50	0.00	0.32
banknote	2	0.08	0.11	0.05	0.03	**0.14**	0.00	0.07
anuram	10	0.00	**0.86**	0.81	0.01	0.81	0.00	0.00
rice	2	0.00	0.18	0.34	**0.56**	0.54	0.00	0.00
mice	8	**0.23**	0.14	0.20	0.21	**0.23**	0.22	0.01
iris	3	**0.72**	0.71	**0.72**	0.69	0.57	0.56	0.56
pendigits	10	0.38	**0.41**	0.30	0.29	**0.41**	0.00	0.07
LetterRec.	26	0.07	0.07	0.1	0.03	**0.11**	0	0.03
BreastCancer	2	0.78	0.51	0.58	0	0.45	0	**0.89**
avg		0.29	0.38	0.41	0.21	**0.42**	0.09	0.22

of the best ARI scores, performing competitively with `minimax` and `complete-linkage`. `CRM*` is competitive with `centroid` and better than `single-linkage`. Although not included in the table, we also computed the ARI scores for `HDBSCAN` and `Dasg`; they perform poorly on these datasets.

In Appendix G.5, we measure robustness to noise via ARI. The result of `RMM*` is similar to those achieved by `minimax`, `average-link` and `complete-linkage`, the best performers. The results of `CRM*` are, again, competitive with the ones obtained by `centroid` and better than those of `single-linkage`.

Finally, we show in Appendix G.6 that `centroid` produces a significant number of inversions for some datasets, and in Section G.7 we present an ablation study regarding the last phase of `RMM*`.

Conclusions. Our main conclusions are the following:

- `RMM*` achieves the lowest k-center cost among the evaluated subcubic-time methods consistently on most datasets and values of k. While it does not outperform `minimax`, the gap is small (e.g., at most 10% in 80% of the cases where $k \leq 100$), and it is significantly faster on several datasets, highlighting a favorable tradeoff between clustering quality and computational efficiency.
- Despite its theoretical guarantees, `CRM*` is less competitive than `RMM*` under the k-center objective. However, it remains an appealing alternative to the centroid method, achieving competitive ARI scores while avoiding inversions.

Acknowledgements. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

The work of the first author is partially supported by CNPq (grants 304272/2026-4 and 408974/2025-7) and the work of the second author was partially financed by Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ), Bolsa Mestrado Nota 10, grant E-26/203.781/2025.

References

[1] Rice (Cammeo and Osmancik). UCI Machine Learning Repository, 2019. DOI: https://doi.org/10.24432/C5MW4Z.

[2] Dry Bean. UCI Machine Learning Repository, 2020. DOI: https://doi.org/10.24432/C50S4B.

[3] Marcel R. Ackermann, Johannes Blömer, Daniel Kuntze, and Christian Sohler. Analysis of agglomerative clustering. *CoRR*, abs/1012.3697, 2010.

[4] E. Alpaydin and Fevzi. Alimoglu. Pen-Based Recognition of Handwritten Digits. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5MG6K.

[5] Daichi Amagata. Fair k-center clustering with outliers. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence*

and Statistics, volume 238 of *Proceedings of Machine Learning Research*, pages 10–18. PMLR, 02–04 May 2024.

[6] Anna Arutyunova, Anna Großwendt, Heiko Röglin, Melanie Schmidt, and Julian Wargalla. Upper and lower bounds for complete linkage in general metric spaces. *Machine Learning*, pages 1–30, 2023.

[7] Mert Balaban, Nima Moshiri, Uyen Mai, Xiaozhou Jia, and Siavash Mirarab. Treecluster: Clustering biological sequences using phylogenetic trees. *PLOS ONE*, 14(8):e0221068, 2019.

[8] Cassotti Matteo Consonni Viviana Ballabio, Davide and Roberto Todeschini. QSAR fish toxicity. UCI Machine Learning Repository, 2019. DOI: https://doi.org/10.24432/C5JG7B.

[9] Mohammadhossein Bateni, Laxman Dhulipala, Willem Fletcher, Kishen N Gowda, D Ellis Hershkowitz, Rajesh Jayaram, and Jakub Lacki. Efficient centroid-linkage clustering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[10] Michael A. Bender and Martín Farach-Colton. The lca problem revisited. In Gaston H. Gonnet and Alfredo Viola, editors, *LATIN 2000: Theoretical Informatics*, pages 88–94, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[11] Sayan Bhattacharya, Martin Costa, Ermiya Farokhnejad, Silvio Lattanzi, and Nikos Parotsidis. Almost optimal fully dynamic k-center clustering with recourse. In *Forty-second International Conference on Machine Learning*, 2025.

[12] J. Bien and R. Tibshirani. Hierarchical clustering with prototypes via minimax linkage". *Journal of the American Statistical Association*, 106:1075–1084, 2011.

[13] Pope D. Brooks, Thomas and Michael Marcolini. Airfoil Self-Noise. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5VW2C.

[14] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[15] Moses Charikar, Vaggos Chatziafratis, and Rad Niazadeh. Hierarchical clustering better than average-linkage. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2291–2304. SIAM, 2019.

[16] Moses Charikar, Vaggos Chatziafratis, Rad Niazadeh, and Grigory Yaroslavtsev. Hierarchical clustering for euclidean data. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 2721–2730. PMLR, 2019.

[17] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. *SIAM J. Comput.*, 33(6):1417–1440, 2004.

[18] Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. Hierarchical clustering: Objective functions and algorithms. *J. ACM*, 66(4):26:1–26:42, 2019.

[19] Nakamura Eduardo Cristo Marco Colonna, Juan and Marcelo Gordo. Anuran Calls (MFCCs). UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C5CC9H.

[20] Cerdeira A. Almeida F. Matos T. Cortez, Paulo and J. Reis. Wine Quality. UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C56S3T.

[21] Sanjoy Dasgupta and Eduardo Sany Laber. New bounds on the cohesion of complete-link and other linkage methods for agglomerative clustering. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 10185–10205. PMLR, 21–27 Jul 2024.

[22] Sanjoy Dasgupta and Philip M. Long. Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences*, 70(4):555–569, 2005. Special Issue on COLT 2002.

[23] Laxman Dhulipala, David Eisenstat, Jakub Łącki, Vahab Mirrokni, and Jessica Shi. Hierarchical agglomerative graph clustering in nearly-linear time. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2676–2686. PMLR, 18–24 Jul 2021.

[24] Laxman Dhulipala, David Eisenstat, Jakub Lacki, Vahab Mirrokni, and Jessica Shi. Hierarchical agglomerative graph clustering in poly-logarithmic depth. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 22925–22940, 2022.

[25] Laxman Dhulipala, Jakub Łącki, Jason Lee, and Vahab Mirrokni. Terahac: Hierarchical agglomerative clustering of trillion-edge graphs. *Proc. ACM Manag. Data*, 1(3), November 2023.

[26] Dimitar Dimitrov, Dénes Türei, Maria Garrido-Rodriguez, Paulina L. Burmedi, Joana S. Nagai, Claire Boys, Roberto O. Ramirez Flores, Hyeongki Kim, Bence Szalai, Igor G. Costa, and Alberto Valdeolivas. Comparison of methods and resources for cell-cell communication inference from single-cell rna-seq data. *Nature Communications*, 13(1):1–13, 2022.

[27] Andrew Draganov, Pascal Weber, Rasmus Skibdahl Melanchton Jørgensen, Anna Beer, Claudia Plant, and Ira Assent. Ultrametric cluster hierarchies: I want 'em all! In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

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

[29] Arnold Filtser, Shaofeng H.-C. Jiang, Yi Li, Anurag Murty Naredla, Ioannis Psarros, Qiaoyuan Yang, and Qin Zhang. Faster approximation algorithms for k-center via data reduction. In *Forty-second International Conference on Machine Learning*, 2025.

[30] R. A. Fisher. Iris. UCI Machine Learning Repository, 1936. DOI: https://doi.org/10.24432/C56C76.

[31] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38(2-3):293–306, 1985.

[32] Anna Großwendt and Heiko Röglin. Improved analysis of complete-linkage clustering. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 656–667. Springer, 2015.

[33] Anna Großwendt, Heiko Röglin, and Melanie Schmidt. Analysis of ward's method. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2939–2957. SIAM, 2019.

[34] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, December 2015.

[35] Dominik Herr, Qi Han, Steffen Lohmann, and Thomas Ertl. Visual clutter reduction through hierarchy-based projection of high-dimensional labeled data. In *Proceedings of Graphics Interface 2016*, GI 2016, pages 109–116. Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine, 2016.

[36] Gardiner Katheleen Higuera, Clara and Krzysztof Cios. Mice Protein Expression. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C50S3Z.

[37] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[38] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York, 1990.

[39] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The uci machine learning repository. `https://archive.ics.uci.edu`, 2024. University of California, Irvine, School of Information and Computer Sciences.

[40] Jon M. Kleinberg and Éva Tardos. *Algorithm design*. Addison-Wesley, 2006.

[41] Ari Kobren, Nicholas Monath, Akshay Krishnamurthy, and Andrew McCallum. A hierarchical algorithm for extreme clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 255–264. ACM, 2017.

[42] Sergei L Kosakovsky Pond, Steven Weaver, Andrew J Leigh Brown, and Joel O Wertheim. Hiv-trace (transmission cluster engine): a tool for large scale molecular epidemiology of hiv-1. *Molecular Biology and Evolution*, 35(7):1812–1819, 2018.

[43] Eduardo S. Laber and Miguel Batista. On the cohesion and separability of average-link for hierarchical agglomerative clustering. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 48710–48739. Curran Associates, Inc., 2024.

[44] Eduardo Sany Laber and Lucas Murtinho. Optimization of inter-group criteria for clustering with minimum size constraints. In *NeurIPS*, 2023.

[45] Scikit learn developers. The 20 newsgroups text dataset. `https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html`, 2017. Accessed: 2025-10-29.

[46] David Lewis. Reuters-21578 Text Categorization Collection. UCI Machine Learning Repository, 1987. DOI: https://doi.org/10.24432/C52G6M.

[47] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.

[48] Volker Lohweg. Banknote Authentication. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C55P57.

[49] Guillaume Marçais and Carl Kingsford. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27(6):764–770, 2011.

[50] Nicholas Monath, Kumar Avinava Dubey, Guru Guruganesh, Manzil Zaheer, Amr Ahmed, Andrew McCallum, Gökhan Mergen, Marc Najork, Mert Terzihan, Bryon Tjanaka, Yuan Wang, and Yuchen Wu. Scalable hierarchical agglomerative clustering. In Feida Zhu, Beng Chin Ooi, and Chunyan Miao, editors, *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 1245–1255. ACM, 2021.

[51] Benjamin Moseley and Joshua R. Wang. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. *J. Mach. Learn. Res.*, 24:1:1–1:36, 2023.

[52] F. Murtagh. *Multidimensional Clustering Algorithms*. Physica-Verlag, Heidelberg, Germany, 1985.

[53] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview, II. *WIREs Data Mining Knowl. Discov.*, 7(6), 2017.

[54] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms, 2011.

[55] Naomi A. O'Leary, Michael W. Wright, J. Rodney Brister, Stacy Ciufo, Diana Haddad, Rich McVeigh, Bhanu Rajput, Barbara Robbertse, Brian Smith-White, Donna Ako-Adjei, et al. Reference sequence (refseq) database at ncbi: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Research*, 44(D1):D733–D745, 2016.

[56] Manon Ragonnet-Cronin, Emma Hodcroft, Stéphane Hué, Emma Fearnhill, Valerie Delpech, Andrew J L Brown, and Samantha J Lycett. Automated analysis of phylogenetic clusters. *Bioinformatics*, 29(14):1709–1710, 2013.

[57] Erich Schubert. Hacam: Hierarchical agglomerative clustering around medoids - and its limitations. In Thomas Seidl, Michael Fromm, and Sandra Obermeier, editors, *LWDA*, volume 2993 of *CEUR Workshop Proceedings*, pages 191–204. CEUR-WS.org, 2021.

[58] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations (ICLR)*, 2018.

[59] David Slate. Letter Recognition. UCI Machine Learning Repository, 1991. DOI: https://doi.org/10.24432/C5ZP40.

[60] R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438, 1958.

[61] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. *Introduction to Data Mining*. Pearson, Boston, MA, 2nd edition, 2018.

[62] Athanasios Tsanas and Max Little. Parkinsons Telemonitoring. UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C5ZS3N.

[63] Michele Tumminello, Fabrizio Lillo, and Rosario N.Mantegna. Correlation, hierarchies, and networks in financial markets. *Journal of Economic Behavior & Organization*, 75(1):40–58, 2010. Transdisciplinary Perspectives on Economic Complexity.

[64] Yuyan Wang and Benjamin Moseley. An objective for hierarchical clustering in euclidean space and its connection to bisecting k-means. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 6307–6314. AAAI Press, 2020.

[65] Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 1963.

[66] Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

[67] WIlliam Wolberg. Breast Cancer Wisconsin (Original). UCI Machine Learning Repository, 1990. DOI: https://doi.org/10.24432/C5HP4Z.

[68] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM)*, pages 745–754. IEEE, 2012.

[69] I-Cheng Yeh. Concrete Compressive Strength. UCI Machine Learning Repository, 2007. DOI: https://doi.org/10.24432/C5PK67.

[70] Zygmunt Zając. goodbooks-10k: Ten thousand books, six million ratings. `https://github.com/zygmuntz/goodbooks-10k`, 2017. GitHub repository; version v1.0 released Sep 19, 2017.

[71] Fan Wei Zhang, Kun and XiaoJing Yuan. Ozone Level Detection. UCI Machine Learning Repository, 2008. DOI: https://doi.org/10.24432/C5NG6W.

[72] Fang Zhou. Geographical Origin of Music. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5VK5D.

A Other bounds for linkage methods

[33] shows that Ward's method, another popular linkage method, is a 2-approximation for the k-means cost function (sum of square errors) when the optimal clusters are well-separated. Bounds on the separability (e.g. the minimum distance between points of different clusters) of some linkage methods can be found in [40, 44, 43].

Many works have analysed the approximation of linkage methods for HAC regarding Dasgupta's cost function and its variants [18, 16, 51, 15]. Dasgupta's cost function assigns a cost to a hierarchical clustering by weighting each pair of points by their similarity and the height at which they are first separated in the dendrogram, thereby encouraging clusterings that merge highly similar points near the leaves. While insightful, they have some shortcomings as discussed in [43]: they are less interpretable than traditional cost functions that measure compactness and separability; they do not reveal the quality of the clusters generated for a specific value of k and for the relevant case in which the points lie in a metric space every hierarchical clustering has $1/2$ approximation for the maximization of the variant studied in [64].

B Proof of Proposition 2.2

Proof. Consider $x, y \in A$ such that $\texttt{diam}(A) = \texttt{dist}(x, y)$. Let z be the farthest point from c_A. Clearly, $\texttt{cost}(A) = \texttt{dist}(c_A, z) \leq \texttt{diam}(A)$. On the other hand, it follows from the triangle inequality that $\texttt{dist}(x, y) \leq \texttt{dist}(x, c_A) + \texttt{dist}(c_A, y)$, so that $\texttt{diam}(A)/2 = \texttt{dist}(x, y)/2 \leq \max\{\texttt{dist}(c_A, x), \texttt{dist}(c_A, y)\} \leq \texttt{cost}(A)$. □

C Additional material and proofs of Section 3.1

C.1 Proof of Theorem 3.1

Proof. Let A_i and B_i be the clusters merged at iteration i of $\mathcal{A}$. It is enough to prove that $\texttt{dist}(A_i, B_i) \leq \texttt{dist}(A_{i+1}, B_{i+1})$. We consider two cases:

Case 1) $A_{i+1} \neq A_i \cup B_i$ and $B_{i+1} \neq A_i \cup B_i$. Then, B_{i+1} and A_{i+1} are active clusters at the beginning of iteration i. Since A_i and B_i are merged, it follows that $\texttt{dist}(A_i, B_i) \leq \texttt{dist}(A_{i+1}, B_{i+1})$.

Case 2) Either $A_{i+1} = A_i \cup B_i$ or $B_{i+1} = A_i \cup B_i$. We assume w.l.o.g. that $A_{i+1} = A_i \cup B_i$. Then, $\texttt{dist}(A_{i+1}, B_{i+1}) = \texttt{dist}(A_i \cup B_i, B_{i+1})$. Since $\mathcal{A}$ belongs to $\mathcal{CRR}$, the representative of A_{i+1} is either that of A_i or that of B_i. Assume w.l.o.g, that the representative of A_{i+1} is the same of A_i. Thus, $\texttt{dist}(A_{i+1}, B_{i+1}) = \texttt{dist}(A_i, B_{i+1})$. Since B_{i+1} is an active cluster at the beginning of iteration i, we have that $\texttt{dist}(A_i, B_i) \leq \texttt{dist}(A_i, B_{i+1}) = \texttt{dist}(A_{i+1}, B_{i+1})$. □

C.2 Proof of Proposition 3.2

Proof. Let $\mathcal{C}^*$ be a k-clustering with maximum diameter $\texttt{OPT}_{DM}(k)$. At the beginning of the iteration t of $\mathcal{A}$ there are $n - t + 1 \geq k + 1$ representatives. Hence, two of them, say x and x', lie in the same cluster of $\mathcal{C}^*$. Since $\mathcal{A}$ in $\mathcal{CRR}$ $\texttt{dist}(c, c') \leq \texttt{dist}(x, x') \leq \texttt{OPT}_{DM}(k)$. □

C.3 Proof of Lemma 3.3

To prove this lemma, we need to introduce some additional concepts.

Let $\mathcal{A}$ be an algorithm in $\mathcal{CRR}$. We define a tree T that keeps track of the merges along $\mathcal{A}$'s execution. The tree T is built as follows: if a representative u is merged with a representative v, during $\mathcal{A}$ execution, and u remains as the representative of the new cluster, then u becomes the parent of v (a directed edge (u, v) is created).

Given a reference clustering $\mathcal{C}^*$ and a path $P = p_0, \ldots, p_M$ in T, we recursively define the *compressed* path P' associated with $(P, \mathcal{C}^*)$. Let $P_i = p_i, \ldots, p_M$ be the suffix of P starting at p_i. If P has only one node/point, then $P' := P$. Otherwise, let $C^* \in \mathcal{C}^*$ be the cluster that contains p_0. We have two cases:

- p_0 is the only point in P that lies in C^*. In this case, P' is given by p_0 followed by the compressed path associated with $(P_1, \mathcal{C}^*)$;
- there is a point in P, different from p_0, that lies in C^*. In this case, let $p_i \in P$ be the farthest point from p_0 (in terms of the number of edges) that belongs to C^*. Then, P' is given by p_0 followed by the compressed path associated with $(P_i, \mathcal{C}^*)$.

The key property of compressed paths is highlighted in the following proposition.

Proposition C.1. *Let P' be the compressed path associated with $(P, \mathcal{C}^*)$. Then, each cluster in $\mathcal{C}^*$ contributes to P' with at most two points. Moreover, if a cluster in $\mathcal{C}^*$ has two points in P', then they are consecutive.*

Example 1. Let $\mathcal{C}^* = (C_1^*, C_2^*, C_3^*)$ be a reference clustering and let $P = p_1, p_2, p_3, p_4, p_5, p_6, p_7$ be a path in the tree T associated with the execution of $\mathcal{A}$, where $\{p_1, p_3\} \subseteq C_1^*$, $\{p_2, p_4, p_6\} \subseteq C_2^*$ and $\{p_5, p_7\} \subseteq C_3^*$. In this case, $P' = p_1, p_3, p_4, p_6, p_7$.

Given a compressed path P', we classify each of its points but the last one as follows:

- A point p'_i is of type (F)irst if p'_i and its successor p'_{i+1} belong to the same cluster in $\mathcal{C}^*$;
- A point p'_i is of type (S)urvivor if it is not of type F and it is a survivor;
- A point p'_i is of type (R)emaining if it is neither of type F nor of type S.

In what follows, we abuse notation by using S, F, and R to represent the sets of points in a compressed path that, respectively, correspond to these types.

In the compressed path of our example, $p_1, p_4 \in F$ and $p_3, p_6 \in R$. Note that p_3 and p_6 cannot be of type S since at the beginning of every iteration that p_3 (resp. p_6) is a representative, p_1 (resp. p_4) is also a representative.

Now we are ready to prove Lemma 3.3.

Proof. We first prove the bound on $\texttt{cost}(C)$, and then we show how to modify the proof to obtain the upper bound on $\texttt{ucost}(C)$. Let u be the representative of cluster C and let v be such that $\texttt{dist}(u, v) = \texttt{cost}(C)$. Let P be a path in T that starts at u and ends at v. Moreover, let $P' = p'_0, \ldots, p'_M$ be the compressed path associated with $(P, \mathcal{C}^*)$.

From the triangle inequality and the definitions of sets F, S and R, we have

$$\texttt{cost}(C) = \texttt{dist}(u, v) \leq \sum_{i=0}^{|P'|-1} \texttt{dist}(p'_i, p'_{i+1}) = \sum_{p'_i \in F \cup S \cup R} \texttt{dist}(p'_i, p'_{i+1}) \tag{2}$$

The definition of type F implies that $p'_i \in F$ and its successor p'_{i+1} in P' are in the same cluster of $\mathcal{C}^*$, so that $\texttt{dist}(p'_i, p'_{i+1}) \leq \texttt{diam}(C_\ell^*)$ for some cluster C_ℓ^* in $\mathcal{C}^*$. Since no cluster in $\mathcal{C}^*$ has more than two points in P' (Proposition C.1) we get that

$$\sum_{p'_i \in F} \texttt{dist}(p'_i, p'_{i+1}) \leq \sum_{i=1}^{k} \texttt{diam}(C_i^*). \tag{3}$$

Moreover, we have that

$$\sum_{p'_i \in R} \texttt{dist}(p'_i, p'_{i+1}) \leq \sum_{i=1}^{k} \texttt{diam}(C_i^*). \tag{4}$$

To see that this inequality holds, consider a point p'_i of type R and let C_ℓ^* be the cluster that contains p'_i. The point p'_{i+1}, the successor of p'_i in P', is a successor of p'_i in the path P, otherwise p'_i would be of type F. Hence, p'_i and p'_{i+1} are merged by the algorithm. When this happens, there is still another point p in C_ℓ^* that is a representative, otherwise p'_i would be a survivor and, then, either of type F or S. This implies that $\texttt{dist}(p'_i, p'_{i+1}) \leq \texttt{dist}(p, p'_i) \leq \texttt{diam}(C_\ell^*)$. Moreover, no cluster in $\mathcal{C}^*$ has more than one point of type R in path P'. These facts together justify inequality (4).

Finally, we argue that

$$\sum_{p'_i \in S} \texttt{dist}(p'_i, p'_{i+1}) \leq s(C)\texttt{OPT}_{DM}(k) \tag{5}$$

Indeed, note that if p'_i belongs to S then its successor p'_{i+1} in P' is also its successor in P. Then, they are merged by $\mathcal{A}$ and Proposition 3.2 guarantees that $\texttt{dist}(p'_i, p'_{i+1}) \leq \texttt{OPT}_{DM}(k)$. Moreover, $|S| \leq s(C)$.

Combining (2)-(5), we establish the upper bound on $\texttt{cost}(C)$.

□

C.4 Proof of Theorem 3.4

Proof. Let $\mathcal{C} = (C_1, \ldots, C_k)$ and let $\mathcal{C}^*$ be a k-clustering whose maximum diameter is $\texttt{OPT}_{DM}(k)$. Moreover, let C be the cluster with maximum diameter among those in $\mathcal{C}$.

It follows from Lemma 3.3 that

$$\texttt{max-diam}(\mathcal{C}) = \texttt{diam}(C) \leq 2\texttt{cost}(C) \leq 4\sum_{i=1}^{k} \texttt{diam}(C^*_i) + 2s(C)\texttt{OPT}_{DM}(k) \leq 6k\texttt{OPT}_{DM}(k).$$

Now, let $\mathcal{C}^*$ be the k-clustering with average diameter $\texttt{OPT}_{AV}(k)$. It follows from Lemma 3.3 that for all j

$$\texttt{diam}(C_j) \leq 2\texttt{cost}(C_j) \leq 2s(C_j) \cdot \texttt{OPT}_{DM}(k) + 4\sum_{i=1}^{k} \texttt{diam}(C^*_i)$$

Thus,

$$\texttt{avg-diam}(C) = \frac{1}{k}\sum_{j=1}^{k} \texttt{diam}(C_j) \leq \frac{2}{k}\left(\sum_{j=1}^{k} s(C_j) \cdot \texttt{OPT}_{DM}(k)\right) + 4 \cdot \sum_{i=1}^{k} \texttt{diam}(C^*_i) \leq 6k\texttt{OPT}_{AV}(k),$$

where the last inequality holds because the number of survivors is at most k. □

C.5 Proof of Theorem 3.5

The proofs of this section consider that for each cluster A, $\texttt{cost}(A) = \texttt{dmax}(c_A, A)$, where c_A is the representative of A employed during the dendrogram construction (when the merges are performed), rather than the center c^*_A of A that is obtained at a post-processing step via the algorithm of Section F.

The following proposition shows that `ucost` is an upper bound on `cost`, and it is useful for the proof of Theorem 3.5.

Proposition C.2. *Every cluster C built by* $\texttt{CRM}^*$ *satisfies* $\texttt{cost}(C) \leq \texttt{ucost}(C)$.

Proof. The proof is by induction on the number of iterations of $\texttt{CRM}^*$. At the beginning the result holds because every cluster has `ucost` and `cost` equal to 0. We assume by induction that the bounds hold for every active cluster at the beginning of iteration j

Let A and B be the clusters merged at iteration j. Assume w.l.o.g. that the representative of $C = A \cup B$ is c_A. Then, $\texttt{ucost}(C) = \max\{\texttt{ucost}(A), \texttt{ucost}(B) + \texttt{dist}(c_A, c_B)\}$ and

$$\texttt{cost}(C) = \max\{\texttt{cost}(A), \texttt{dmax}(c_A, B)\}$$

.

If $\texttt{cost}(C) = \texttt{cost}(A)$ the result holds because by induction $\texttt{cost}(A) \leq \texttt{ucost}(A) \leq \texttt{ucost}(C)$.

If $\texttt{cost}(C) = \texttt{dmax}(c_A, B)$, then

$\texttt{cost}(C) = \texttt{dmax}(c_A, B) \leq \texttt{dist}(c_A, c_B) + \texttt{cost}(B) \leq \texttt{ucost}(B) + \texttt{dist}(c_A, c_B) \leq \texttt{ucost}(C)$

□

Now, we are ready to complete the proof.

Proof. We show by induction on the number of iterations that $\texttt{ucost}(C) \leq \texttt{OPT}_{DM}(k)\log(|C)|$, for every cluster C built by $\texttt{CRM}^*$. This is enough to establish the result because $\texttt{diam}(C) \leq 2\texttt{cost}(C) \leq 2\texttt{ucost}(C)$.

At the beginning, the result holds because every cluster has size 1 and ucost 0.

Let us consider that A and B are merged at iteration j to produce cluster C. We assume w.l.o.g that the representative of C is the same of A, that is, c_A.

Note that $\texttt{ucost}(C) = \max\{\texttt{ucost}(A), \texttt{ucost}(B) + \texttt{dist}(c_A, c_B)\}$. We split the proof into two cases:

Case 1) $\texttt{ucost}(C) = \texttt{ucost}(A)$, In this case, the induction hypothesis assures that $\texttt{ucost}(A) \leq \texttt{OPT}_{DM}(k)\log(|A|) < \texttt{OPT}_{DM}(k)\log(|C|)$.

Case 2) $\texttt{ucost}(C) = \texttt{ucost}(B) + \texttt{dist}(c_A, c_B)$. In this case,

$$\texttt{ucost}(C) = \texttt{dist}(c_A, c_B) + \texttt{ucost}(B) \leq \texttt{OPT}_{DM}(k) + \texttt{ucost}(B), \tag{6}$$

where the inequality follows from Proposition 3.2.

We consider two scenarios:

(i) $|B| \leq |C|/2$. In this case, the induction hypothesis assures that $\texttt{ucost}(B) \leq \texttt{OPT}_{DM}(k)\log(|B|) \leq \texttt{OPT}_{DM}(k)\log(|C|/2)$. By replacing this inequality in 6, we get that $\texttt{ucost}(C) \leq \texttt{OPT}_{DM}(k) + \texttt{OPT}_{DM}(k)\log(|C|/2) \leq \texttt{OPT}_{DM}(k)\log(|C|)$.

(ii) $|B| > |C|/2$. This implies that $|A| \leq |C|/2$, so the choice of the algorithm and the induction hypothesis assure that $\texttt{ucost}(B) \leq \texttt{ucost}(A) \leq \texttt{OPT}_{DM}(k)\log(|A|) \leq \texttt{OPT}_{DM}(k)\log(|C|/2)$. Again, by replacing this inequality in 6, we establish the induction. □

D Lower Bounds

The following example shows that the bound on avg-diam given by Theorem 3.4 is tight. Consider a set $\mathcal{X} = A \cup A'$ of $2k$ points, where $A = \{a_1, \ldots, a_k\}$ and $A' = \{a'_1, \ldots, a'_k\}$. We have $\texttt{dist}(a_i, a'_i) = (1-\epsilon)$, for $i = 1, \ldots, k$, and all other distances are equal to 1. $\mathcal{A}$ builds k clusters, where the ith cluster is $\{a_i, a'_i\}$. The sum of the diameters is $k(1-\epsilon)$. On the other hand, the k-clustering $(A, a'_1, a'_2, \ldots, a'_{k-2}, \{a'_{k-1}, a'_k\})$ has sum of diameters 2.

Theorem D.1. *There is an instance I for which every algorithm in $\mathcal{CRR}$ builds a k-clustering $\mathcal{C}$, with* $\texttt{avg-diam}(\mathcal{C}) \geq \frac{k-\epsilon}{2}\texttt{OPT}_{AV}(k)$.

We note that this example also shows an $\Omega(k)$ lower bound for complete-linkage, average-link and single-linkage.

Now, we show that the $O(\log n)$ and $O(k)$ bounds on max-diam given by Theorem 3.5 and 3.4, respectively, are tight up to constant factors. For that, we define a sequence of instances $I_2, \ldots, I_k$, where the instance I_j is built from instance I_{j-1}.

Each instance I_j is associated with a graph G_j that has weights on edges and color on nodes. Each node of G_j corresponds to a point of I_j and the distance between two points u and v in I_j is given by the weight of the shortest path between u and v in G_j.

The graph G_2 has 4 nodes v_1, v_2, v_3 and v_4, the first two have color 1 and the last two have color 2. Points having the same color are connected by an edge with weight 1. Let $\alpha_2 = 1 - \epsilon$, for a positive ϵ very close to 0. We create edges (v_1, v_3) and (v_2, v_4), both with weights α_2.

The graph G_j is built from (I_{j-1}, G_{j-1}) as follows. Let $\mathcal{C} = (C_1, \ldots, C_{j-1})$ be the $(j-1)$-clustering built by $\texttt{CRM}^*$ for instance I_{j-1}. Let H_i be the subgraph of G_{j-1} induced by the nodes associated with the points in C_i. Let H_j be a graph isomorphic to H_1. The color of a node u' in H_j is equal to the color of its corresponding node in H_1. Moreover, let H'_i be a graph isomorphic to H_i for $i = 1, \ldots, j$. In this case the color of a node u' in H'_i is 1+color(u), where u is the node corresponding to u' in H_i. Recall that $\alpha_2 = (1-\epsilon)$ and let $\alpha_{i+1} = (1+\alpha_i)/2$, for $i > 2$. Note that for every ℓ, $\alpha_\ell < 1$.

The graph G_j is then obtained by: (i) adding the subgraphs $H_j, H'_1, \ldots, H'_j$ to G_{j-1}; (ii) creating j edges with weight α_j, where the ith edge connects the node in H_i with color $j-1$ (we will show that it does exist) with the node of H'_i that has color j and (iii) adding an edge of weight 1 between any two nodes that have the same color but do not have an edge yet.

Lemma D.3, presented below, shows that I_j is an instance in which $\texttt{CRM}^*$ builds a j-clustering whose maximum diameter is a factor of $(j-1)(1-\epsilon)$ from the optimal. The proof assumes a particular implementation of $\texttt{CRM}^*$ that has the following property: when two clusters A and B, with $\texttt{ucost}(A) = \texttt{ucost}(B)$, are merged, $\texttt{CRM}^*$ break ties choosing the point associated with the highest color as the representative of $A \cup B$.

In what follows, we use $\texttt{CRM}^*(I)$ to denote the execution of $\texttt{CRM}^*$ on instance I. We need an auxiliary result.

Proposition D.2. *Let w be a positive number. Moreover, let $\mathcal{X}$ and $\mathcal{X}'$ be sets of points and* dist *be a distance function that satisfy* $\texttt{dist}(x, x') > w$*, for every $x \in \mathcal{X}$ and $x' \in \mathcal{X}'$.*

Let ℓ be the first iteration of $\texttt{CRM}^*(\mathcal{X}, \texttt{dist})$*, in which the distance between any pair of representatives is larger than w. Moreover, let C be a cluster that is active at the beginning of iteration ℓ and let c be its representative.*

If ℓ' is the first iteration in which representatives at a distance larger than w are merged by $\texttt{CRM}^*(\mathcal{X} \cup \mathcal{X}', \texttt{dist})$*, then C is an active cluster at the beginning of iteration ℓ' and c is its representative.*

Proof. Since every point in $\mathcal{X}$ is at a distance larger than w from every point in $\mathcal{X}'$ and $\texttt{CRM}^*$ always merges the closest representatives, all the merges executed by $\texttt{CRM}^*(\mathcal{X})$ in the first $\ell - 1$ iteration will be executed by $\texttt{CRM}^*(\mathcal{X} \cup \mathcal{X}')$ in the first $\ell' - 1$ iterations. Thus, the cluster C, with representative c, that is active at the beginning of the iteration ℓ of $\texttt{CRM}^*(\mathcal{X})$ will be also active at iteration ℓ' of $\texttt{CRM}^*(\mathcal{X} \cup \mathcal{X}')$. □

Lemma D.3. *For instance I_j, there is a j-clustering whose maximum diameter is 1 and $\texttt{CRM}^*$ builds a j-clustering $\mathcal{C}^j$ such that $\texttt{max-diam}(\mathcal{C}^j) \geq (j-1)(1-\epsilon)$.*

Proof. Let C^*_i be the points with color i in instance I_j and let $\mathcal{C}^* = (C^*_1, \ldots, C^*_j)$. By design, points with the same color are at a distance at most 1, so that $\texttt{max-diam}(\mathcal{C}^*) = 1$.

Let $\mathcal{C}^j$ be the j-clustering built by $\texttt{CRM}^*$ for instance I_j. In order to give a lower bound on the diameter of the k-clustering obtained by $\texttt{CRM}^*$, we first prove that the construction of $\mathcal{C}^j$ satisfies the following properties:

- (a) the distance between points that lie in different clusters in $\mathcal{C}^j$ is at least 1;
- (b) every cluster in $\mathcal{C}^j$ has a point of color 1 and its representative has color j;
- (c) the maximum distance between representatives that are merged along the construction of $\mathcal{C}^j$ is α_j

We prove by induction. For $j = 2$, $\texttt{CRM}^*$ builds the clustering $(\{v_1, v_3\}, \{v_2, v_4\})$ that satisfies the desired properties.

We assume the result holds for $j-1$ and prove that it also holds for j. Let $C_1, \ldots, C_{j-1}$ be the $(j-1)$-clustering obtained by $\texttt{CRM}^*(I_{j-1})$. Let $S = \{H_1, \ldots, H_j, H'_1, \ldots, H'_j\}$, be the set of $2j$ subgraphs that are used to build the graph G_j from (I_{j-1}, G_{j-1}). Moreover, let C_j be the set of points associated with the nodes in H_j and let C'_i be the set of points associated with the nodes in H'_i. Recall that I_j is comprised by the set of points that lie in the clusters of S.

At the beginning of iteration $\ell = (|I_{j-1}| - j + 2)$ of $\texttt{CRM}^*(I_{j-1})$, we have the clusters $C_1, \ldots, C_{j-1}$ and, by induction, in the $(|I_{j-1}| - j + 1)$ first iterations of $\texttt{CRM}^*(I_{j-1})$ representatives at a distance larger than α_{j-1} are not merged. Moreover, at the beginning of iteration ℓ, the induction assures that the distance between any two representatives is at least 1, which is larger than α_{j-1}.

Let $\mathcal{X}' = C_j \cup C'_1 \cdots \cup C'_j$. By design, $\texttt{dist}(x, x') \geq \alpha_{j-1}$ for $x \in I_{j-1}$ and $x' \in \mathcal{X}'$. Moreover, let ℓ' be the first iteration in which two representatives at a distance larger than α_{j-1} are merged by $\texttt{CRM}^*(I_j)$. Therefore, it follows from Proposition D.2 that the clusters $C_1, \ldots, C_{j-1}$ are active at the

beginning of iteration ℓ' of $\texttt{CRM}^*(I_j)$. Moreover, it follows by induction and by Proposition D.2 that their representatives are points with color $j-1$.

Similarly, we can conclude that $C_j, C'_1, \ldots, C'_j$ are active at the beginning of iteration ℓ' of $\texttt{CRM}^*(I_j)$ due to the isomorphism employed to build the graphs $H_j, H'_1, \ldots, H'_j$.

Thus, at the beginning of iteration ℓ', the clusters of set S are active. Then, in the next j iterations, the clusters C_ℓ and C'_ℓ, for $\ell = 1, \ldots, j$, are merged because, by design, their representatives are the closest ones, being at distance α_j. Thus, items (a) and (c) are established. We turn to item (b). Let c_i and c'_i be the representatives of C_i and C'_i, respectively. The isomorphism between the underlying graphs of C_i and C'_i implies that $\texttt{ucost}(C_i) = \texttt{ucost}(C'_i)$. Moreover, c_i and c'_i have colors $j-1$ and j, respectively. Thus, our assumption regarding $\texttt{CRM}^*$, ensures that the representative of the cluster $C_i \cup C'_i$ is a point with color j. Furthermore, $C_i \cup C'_i$ has a point with color 1 because, by induction, C_i has a point of color 1. Hence, we also establish item (b).

Consider the cluster $C_i \in C^j$ and points u and v in C_i such that color(u)=1 and color(v)=j. By design, we have only edges between nodes with colors differing by at most 1. Thus, any path between u and v has at least $j-1$ edges. Since each edge has weight at least $1-\epsilon$, we have that $\texttt{diam}(C_i) \geq \texttt{dist}(u, v) \geq (j-1)(1-\epsilon)$. □

The above result implies that for every k, $\texttt{CRM}^*$ builds a k-clustering $\mathcal{C}$ with $\texttt{max-diam}(\mathcal{C}) \in \Omega(k)\texttt{OPT}_{DM}(k)$ for instance I_k.

Note that $|I_2| = 4$ and $|I_j| = \left(2 + \frac{2}{j-1}\right)|I_{j-1}|$, so that $|I_k| = k2^{k-1}$. Thus, we also have that, for the instance I_k, $\texttt{max-diam}(\mathcal{C}) \in \Omega(\log n)\texttt{OPT}_{DM}(k)$.

Theorem D.4. *There is an instance for which* $\texttt{CRM}^*$ *builds a* k*-clustering* $\mathcal{C}$ *that satisfies* $\texttt{max-diam}(\mathcal{C}) \in \Omega(\log n) \cdot \texttt{OPT}_{DM}(k)$. *Furthermore, for every integer* $k \leq \log n$*, there is an instance in which* $\texttt{CRM}^*$ *builds a* k*-clustering that satisfies* $\texttt{max-diam}(\mathcal{C}) \in \Omega(k)\texttt{OPT}_{DM}(k)$.

E Proofs of Section 3.2

E.1 $\texttt{RMM}^*$ does not produce inversions

Theorem E.1. *The dendrogram produced by* $\texttt{RMM}^*$ *has no inversions.*

Proof. Let A_i and B_i be the clusters merged at iteration i of $\texttt{RMM}^*$. It is enough to prove that $\texttt{dist}(A_i, B_i) \leq \texttt{dist}(A_{i+1}, B_{i+1})$. We consider two cases:

Case 1) $A_{i+1} \neq A_i \cup B_i$ and $B_{i+1} \neq A_i \cup B_i$. Then, B_{i+1} and A_{i+1} are active clusters at the beginning of iteration i. Since A_i and B_i are merged, it follows that $\texttt{dist}(A_i, B_i) \leq \texttt{dist}(A_{i+1}, B_{i+1})$.

Case 2) Either $A_{i+1} = A_i \cup B_i$ or $B_{i+1} = A_i \cup B_i$. We assume w.l.o.g. that $A_{i+1} = A_i \cup B_i$. Then, $\texttt{dist}(A_{i+1}, B_{i+1}) = \texttt{dist}(A_i \cup B_i, B_{i+1})$. Due to $\texttt{RMM}^*$'s choice, the representative of A_{i+1} is either that of A_i or that of B_i. Assume w.l.o.g, that the representative of A_{i+1} is the same of A_i, say c_{A_i}. Thus,

$$\texttt{dist}(A_{i+1}, B_{i+1}) = \min\{\texttt{dmax}(c_{A_i}, A_i \cup B_i \cup B_{i+1}), \texttt{dmax}(c_{B_{i+1}}, A_i \cup B_i \cup B_{i+1})\} \geq$$
$$\min\{\texttt{dmax}(c_{A_i}, A_i \cup B_{i+1}), \texttt{dmax}(c_{B_{i+1}}, A_i \cup B_{i+1})\} = \texttt{dist}(A_i, B_{i+1}) \geq \texttt{dist}(A_i, B_i),$$

where the last inequality holds because B_{i+1} is an active cluster at the beginning of iteration i. □

E.2 An $O(n^2)$ time implementation for $\texttt{RMM}^*$

To achieve an $O(n^2)$ time complexity, $\texttt{RMM}^*$ stores $\texttt{dmax}(c_X, Y)$ for every two (possibly equal) clusters X and Y, where c_X is the representative of X.

These values can be updated in $O(1)$ time after A and B are merged. In fact, let $c_{A\cup B}$ be the representative of $A \cup B$. Then, $\texttt{dmax}(c_{A\cup B}, C)$ is equal to $\texttt{dmax}(c_A, C)$ if $c_{A\cup B} = c_A$ or equal to $\texttt{dmax}(c_B, C)$, otherwise. Moreover,

$$\texttt{dmax}(c_C, A \cup B) = \max\Big(\texttt{dmax}(c_C, A), \texttt{dmax}(c_C, B)\Big)$$

Now, we argue that $\texttt{dist}(A \cup B, C)$ can be computed in $O(1)$ time, for every cluster C, after merging A and B.

We have that

$$\texttt{dist}(A \cup B, C) := \min\{\texttt{dmax}(c_{A\cup B}, A \cup B \cup C), \texttt{dmax}(c_C, A \cup B \cup C)\},$$

Let us assume w.l.o.g. that the representative of $A \cup B$ is c_A. In this case, $\texttt{dmax}(c_{A\cup B}, A \cup B \cup C) = \max\{\texttt{dmax}(c_A, A), \texttt{dmax}(c_A, B), \texttt{dmax}(c_A, C)\}$ and $\texttt{dmax}(c_C, A \cup B \cup C) = \max\{\texttt{dmax}(c_C, A), \texttt{dmax}(c_C, B), \texttt{dmax}(c_C, C)\}$.

It remains to show that property (i) of Theorem 2.1 holds.

Proposition E.2. *For every three disjoint clusters A, B and H,*

$$\texttt{dist}(A \cup B, H) \geq \min\big(\texttt{dist}(A, H), \texttt{dist}(B, H)\big)$$

Proof. We first consider that c_A is the representative of $(A\cup B)\cup H$. Then, it is also the representative of $A \cup B$. Thus,

$$\max\big(\texttt{cost}(A), \texttt{dmax}(c_A, B)\big) \leq \max\big(\texttt{cost}(B), \texttt{dmax}(c_B, A)\big)$$

and

$$\texttt{cost}(A \cup B) = \max\big(\texttt{cost}(A), \texttt{dmax}(c_A, B)\big)$$

It follows that

$$\begin{aligned}
\texttt{dist}(A \cup B, H) = \\
\max\big(\texttt{cost}(A \cup B), \texttt{dmax}(c_A, H)\big) = \\
\max\Big(\texttt{cost}(A), \texttt{dmax}(c_A, B), \texttt{dmax}(c_A, H)\Big) \geq \\
\max\big(\texttt{cost}(A), \texttt{dmax}(c_A, H)\big) \geq \\
\min\Big(\max\big(\texttt{cost}(A), \texttt{dmax}(c_A, H)\big), \max\big(\texttt{cost}(H), \texttt{dmax}(c_H, A)\big)\Big) \geq \\
\texttt{dist}(A, H) \geq \\
\min\big(\texttt{dist}(A, H), \texttt{dist}(B, H)\big)
\end{aligned}$$

The case in which c_B is the representative of $(A \cup B) \cup H$ is analogous.

Now, assume c_H is the representative of $A \cup B \cup H$. We have that

$$\begin{aligned}
\texttt{dist}(A \cup B, H) = \\
\max\big(\texttt{cost}(H), \texttt{dmax}(c_H, A \cup B)\big) = \\
\max\big(\texttt{cost}(H), \texttt{dmax}(c_H, A), \texttt{dmax}(c_H, B)\big) \geq \\
\max\big(\texttt{cost}(H), \texttt{dmax}(c_H, A)\big) \geq \\
\min\Big(\texttt{dmax}(c_H, A \cup H\Big), \min\Big(\max\big(\texttt{cost}(H), \texttt{dmax}(c_H, A)\big), \max\big(\texttt{cost}(A), \texttt{dmax}(c_A, H)\big)\Big) \geq \\
\texttt{dist}(A, H) \geq \\
\min\big(\texttt{dist}(A, H), \texttt{dist}(B, H)\big)
\end{aligned}$$

□

E.3 Proof of Theorem 3.6

The proofs of this section consider that for each cluster A, $\texttt{cost}(A) = \texttt{dmax}(c_A, A)$, where c_A is the representative of A employed during the dendrogram construction (when the merges are performed), rather than the center c_A^* of A that is obtained at a post-processing step via the algorithm of Section F.

We start with a simple proposition.

Proposition E.3. *For every clusters A and B, with representatives c_A and c_B,*

$$\texttt{dist}(A, B) \leq \texttt{dist}(c_A, c_B) + \max\{\texttt{cost}(A), \texttt{cost}(B)\}$$

Proof. We have that

$$\begin{aligned}
\texttt{dist}(A, B) = & \quad (7)\\
\min\{\texttt{dmax}(c_A, A \cup B), \texttt{dmax}(c_B, A \cup B)\} \leq & \quad (8)\\
\texttt{dmax}(c_A, A \cup B) = & \quad (9)\\
\max\{\texttt{cost}(A), \texttt{dmax}(c_A, B)\} \leq & \quad (10)\\
\max\{\texttt{cost}(A), \texttt{dist}(c_A, c_B) + \texttt{dmax}(c_B, B)\} = & \quad (11)\\
\max\{\texttt{cost}(A), \texttt{dist}(c_A, c_B) + \texttt{cost}(B)\} \leq & \quad (12)\\
\texttt{dist}(c_A, c_B) + \max\{\texttt{cost}(A), \texttt{cost}(B)\} & \quad (13)
\end{aligned}$$

□

Proof of Theorem 3.6. Let $\mathcal{C}^*$ be a k-clustering with maximum diameter $\texttt{OPT}_{DM}(k)$. For a cluster C let $\texttt{cov}(C, \mathcal{C}^*)$ be the set of clusters in $\mathcal{C}^*$ that C intersects.

For $0 \leq j \leq 2k$, let i_j be the first iteration in which the RMM* obtains a cluster C with $\texttt{cost}(C) > 2j\texttt{OPT}_{DM}(k)$. Thus, at the beginning of iteration i_j the cost of each cluster is at most $2j\texttt{OPT}_{DM}(k)$ and at the end of this iteration, there is a cluster with cost larger than $2j\texttt{OPT}_{DM}(k)$. We use phase j to denote the set of iterations $i_j, i_j + 1, \ldots, i_{j+1} - 1$.

Linear Bound on k. We show that $\texttt{max-diam}(\mathcal{C}) \leq (8k+2)\texttt{OPT}_{DM}(k)$. Pick some cluster $C_i^* \in \mathcal{C}^*$. At the beginning of iteration i_{j+1} there is at most 1 cluster with representative in C_i^* and cost smaller than or equal to $(2j+1)\texttt{OPT}_{DM}(k)$. In fact, assume there are 2 clusters, say C and C' with these properties. Let A and B be the clusters merged at iteration i_{j+1}. It follows from the choice of the algorithm and Proposition E.3 that

$$\texttt{cost}(A \cup B) = \texttt{dist}(A, B) \leq \texttt{dist}(C, C') \leq$$
$$\texttt{OPT}_{DM}(k) + \max\{\texttt{cost}(C), \texttt{cost}(C')\} \leq (2j+2)\texttt{OPT}_{DM}(k),$$

which contradicts the definition of i_{j+1}. If a cluster has cost not larger than $(2j+1)\texttt{OPT}_{DM}(k)$ at the beginning of iteration i_{j+1} we say that its representative is *marked* on phase j.

At the beginning of iteration i_{j+1} (end of phase j), consider a cluster C' whose representative c' was not marked at phase j. Moreover, let C be the active cluster that has representative c' at the beginning of phase j. Since C and C' share the same representative, we have $C \subseteq C'$. We will argue that $|\texttt{cov}(C', \mathcal{C}^*)| \geq |\texttt{cov}(C, \mathcal{C}^*)| + 1$. Note that $|\texttt{cov}(C')| \geq |\texttt{cov}(C)|$ since $C \subseteq C'$. For the sake of a contradiction, we assume that $|\texttt{cov}(C')| = |\texttt{cov}(C)|$. Since $C \subseteq C'$, we must have $\texttt{cov}(C') = \texttt{cov}(C)$. Let x' be a point in C' such that $\texttt{cost}(C') = \texttt{dist}(c', x')$. Furthermore, let x be a point in C that belongs to the same cluster that x' lies in $\mathcal{C}^*$. It follows that $\texttt{cost}(C') = \texttt{dist}(c', x') \leq \texttt{dist}(c', x) + \texttt{dist}(x, x') \leq 2j\texttt{OPT}_{DM}(k) + \texttt{OPT}_{DM}(k)$, which contradicts the fact that c' was not marked in phase j.

We consider two cases:

Case 1) RMM finishes by the beginning of phase $2k$.

Let C be the cluster with maximum diameter among the k clusters that RMM outputs. The definition of a phase $2k$ implies that $\texttt{cost}(C) \leq 4k\texttt{OPT}_{DM}(k)$.

Case 2) RMM does not finish by the beginning of phase $2k$.

In this case, at the beginning of phase $2k$ there are at least $k+1$ active clusters, so that there are two representatives, say c and c', that lie in the same cluster C_i^* of $\mathcal{C}^*$.

At each phase, either c or c' is not marked because at most one point in C_i^* can be marked per phase. Since there are $2k$ phases, either c or c' is not marked in k phases. We assume w.l.o.g that c is not marked in k phases. Thus, $|\texttt{cov}(C, \mathcal{C}^*)| = k$, where C is the cluster represented by c at the beginning of phase $2k$. Thus, we can conclude that the distance between any two points u and v from the input set is at most $(8k+2)\texttt{OPT}_{DM}(k)$. In fact, let u' (resp. v') be a point that belongs to both C and to the cluster in which u (resp. v) lies in in $\mathcal{C}^*$. We have that $\texttt{dist}(u,v) \leq \texttt{dist}(u,u') + \texttt{dist}(u',c) + \texttt{dist}(c,v') + \texttt{dist}(v',v) \leq (8k+2)\texttt{OPT}_{DM}(k)$.

Logarithmic Bound. We also make use of phases but now we redefine i_j to be the first iteration in which the algorithm obtains a cluster C with $\texttt{cost}(C) > j\texttt{OPT}_{DM}(k)$. Let $n_0 = n$ and n_j be the number of clusters at the beginning of phase j, for $j \geq 1$. We argue that $n_{j+1} \leq (n_j - k)/2 + k$. In fact, let S be the set of clusters at the beginning of phase j that do not merge with any other cluster by the beginning of phase $j+1$. If S has more than k clusters then two of them, say C and C', have representatives in the same cluster $C_\ell^* \in \mathcal{C}^*$. Let A and B be the clusters merged at iteration i_{j+1}. It follows from the choice of the algorithm and inequalities (9)-(13) that

$$\texttt{cost}(A \cup B) = \texttt{dist}(A,B) \leq \texttt{dist}(C,C') \leq$$
$$\texttt{OPT}_{DM}(k) + \max\{\texttt{cost}(C), \texttt{cost}(C')\} \leq (j+1)\texttt{OPT}_{DM}(k),$$

which contradicts the definition of i_{j+1}. Since $|S| \leq k$, then at least $n_j - k$ clusters are merged, so $n_{j+1} \leq (n_j - k)/2 + k = (n_j + k)/2$.

From this formula, we get that $n_j \leq (n-k)/2^j + k$. Thus, the number of clusters at the beginning of phase $\lceil \log(n-k) \rceil$ is at most $k+1$ and each of these clusters has $\texttt{cost}$ at most $\lceil \log(n-k) \rceil \texttt{OPT}_{DM}(k)$. Thus, by running an additional iteration, the algorithm obtains k clusters, each of them with cost at most $(2\lceil \log(n-k) \rceil + 1)\texttt{OPT}_{DM}(k)$ □

F Computing centers

For the algorithms presented below, the input $\mathcal{D}$ is a dendrogram (tree) with $2n-1$ nodes $X_1, \ldots, X_{2n-1}$ representing the merges of a linkage method. The first n nodes are singletons that correspond to the n points in the input $\mathcal{X}$. The node X_i, for $i > n$, has two children $X_{\ell(i)}$ and $X_{r(i)}$ corresponding to the clusters that are merged to produce X_i.

F.1 Computing centers in $O(n^2)$ time when the input is a matrix

Algorithm 4 outputs the centers of the clusters $X_1, \ldots, X_{2n-1}$. The approach consists of computing $\texttt{dmax}(p, X_i)$ for each cluster X_i and each x in X_i. These values are stored in the variable $\texttt{MaxDist}$.

Time Complexity. The overall complexity of the algorithm is

$$O\left(\sum_{i=1}^{n-1} |X_{\ell(i)}| \cdot |X_{r(i)}|\right). \tag{14}$$

We note that each pair of points (x,y) contributes to at most one product $|X_{\ell(i)}| \cdot |X_{r(i)}|$ in the sum (14). In fact, if they contribute to $|X_{\ell(i)}| \cdot |X_{r(i)}|$, then they will lie in X_i after the merge between $X_{\ell(i)}$ and $Y_{r(i)}$, preventing them from contribute to another product. Thus, the sum in (14) is $O(n^2)$.

Having $\texttt{dmax}(x, X_i)$ for every $x \in X_i$, the center for X_i can be computed in $O(n)$ time by a simple scan. Thus, the algorithm runs in quadratic time.

F.2 Computing the centers for a graph dataset in $O(m \log n + n \log^2 n)$ time

Here, we assume w.l.o.g that the children of X_i in the dendrogram $\mathcal{D}$ satisfy $|X_{\ell(i)}| \geq |X_{r(i)}|$. This ensures that we always merge the smaller cluster into the larger one.

We associate each edge $e = uv$ of the graph with a *stamp*, defined as the minimum integer j such that both u and v belong to X_j. This stamp corresponds to the Lowest Common Ancestor (LCA)

Algorithm 4 Computing centers in $O(n^2)$ time.

Require: $\mathcal{D}$: the dendrogram associated with a linkage method; distance function `dist`
Ensure: the centers of the clusters X_i, $i = 1 \ldots, 2n-1$.

For $i = 1, \ldots, n$
  $\texttt{MaxDist}[x, X_i] \leftarrow 0$
  Center(X_i) $\leftarrow$ the single point in X_i.
For $i = n+1, \ldots, 2n-1$
  For each $x \in X_{\ell(i)}$
    $\texttt{MaxDist}[x, X_i] \leftarrow \texttt{MaxDist}[x, X_{\ell(i)}]$
    For each $y \in X_{r(i)}$
      $\texttt{MaxDist}[x, X_i] \leftarrow \max\{\texttt{MaxDist}[x, X_i], \texttt{dist}(x, y)\}$
  For each $x \in X_{r(i)}$
    $\texttt{MaxDist}[x, X_i] \leftarrow \texttt{MaxDist}[x, X_{r(i)}]$
    For each $y \in X_{\ell(i)}$
      $\texttt{MaxDist}[x, X_i] \leftarrow \max\{\texttt{MaxDist}[x, X_i], \texttt{dist}(x, y)\}$
  Center(X_i) $\leftarrow arg\min\{\texttt{MaxDist}[z, X_i] | z \in X_i\}$

of u and v in $\mathcal{D}$. All stamps can be computed in $O(m \log n + n \log n)$ time by preprocessing $\mathcal{D}$ to answer LCA queries [10]. Our presentation is in terms of the weights of the edges. The weight of an edge uv corresponds to the distance between the points associated with nodes u and v.

Algorithm 5 details the procedure. It takes $\mathcal{D}$ as input and performs $n-1$ iterations. In the iteration associated with cluster X_i, it computes the center of X_i using the data structure H_{X_i} and global state arrays. The structure H_{X_i} is a Binary Search Tree (BST) that maintains the pair $(W_{max}[v], v)$ for each candidate $v \in X_i$; where $W_{max}[v]$ is the maximum weight of an edge incindent to v, among those that have been considered by the algorithm so far. The field W_{max} is the search key of the BST. The global array $\deg[\cdot]$ tracks the number of neighbors of each node within the current cluster.

Initially, the structures are set up for the leaf clusters $X_1, \ldots, X_n$. Then, the main loop executes three key blocks for each internal node i:

The first block (**Merge Step**) initializes H_{X_i} by inheriting the structure from the larger child $H_{X_{\ell(i)}}$ and inserting elements from the smaller child $X_{r(i)}$. Due to the assumption that $|X_{\ell(i)}| \geq |X_{r(i)}|$, the size of the cluster containing a node v at least doubles each time v is moved from a smaller child to a parent structure. Consequently, each node is re-inserted at most $O(\log n)$ times. Since each insertion takes $O(\log n)$, the total time complexity for this block is $O(n \log^2 n)$.

The second block (**Edge Processing**) updates H_{X_i} and the global arrays by processing all edges with stamp i. For each such edge, we update the global degree and the maximum weight for the endpoints in $O(\log n)$ time. Since each edge is considered exactly once throughout the entire algorithm, the total contribution of this block is $O(m \log n)$.

The third block (**Center Selection**) identifies the center of X_i. The candidate center is the node z with the smallest key in H_{X_i} that is connected to every other node in the cluster (i.e., $\deg[z] = |X_i| - 1$). To find this node, the algorithm repeatedly removes the element with the minimum key from H_{X_i} until the condition is met. A node removed from H_{X_i} in this step is permanently discarded and never re-inserted. Thus, each node is removed at most once, making the total contribution of this block $O(n \log n)$.

G Experimental Study

In this section, we provide additional details about our experiments. Our codes are available in the supplementary material (.zip file).

All experiments were performed on a virtualized server environment equipped with an Intel Xeon (Cascadelake) processor (10 vCPUs) and 48 GB of RAM, running Ubuntu 22.04.5 LTS.

G.1 Datasets

We used classical tabular datasets available in UCI [39], and also from specific domains, namely, text, movies/books rating, and biology. For the tabular datasets, we normalized the values in each column

Algorithm 5 COMPUTING CENTERS USING GLOBAL STATE ARRAYS

Require: $\mathcal{D}$: dendrogram, $G = (V, E, W)$: weighted graph
Ensure: The center of clusters $X_1, \ldots, X_{2n-1}$.

Global Structures:
    Arrays $\deg[1 \ldots n]$ and $W_{max}[1 \ldots n]$ initialized to 0.
    Binary Trees (Sets) $H_{X_1}, \ldots, H_{X_{2n-1}}$.
Initialization:
For $u = 1, \ldots, n$
    $H_{X_u} \leftarrow \{(0, u)\}$
    $|X_u| \leftarrow 1$
End For
Main Loop:
For $i = n + 1, \ldots, 2n - 1$
    $|X_i| \leftarrow |X_{\ell(i)}| + |X_{r(i)}|$

    Merge Step:
        if $|X_{\ell(i)}| < |X_{r(i)}|$ swap them so that $X_{\ell(i)}$ becomes the largest
        $H_{X_i} \leftarrow H_{X_{\ell(i)}}$ {Inherit structure from $H_{X_{\ell(i)}}$}
        For every node $v \in H_{X_{r(i)}}$
            Insert $(W_{max}[v], v)$ into H_{X_i}
        End For

    Process Edges:
        For every edge $e = (u, v)$ with stamp i and weight w
            Call UpdateNode(H_{X_i}, u, w)
            Call UpdateNode(H_{X_i}, v, w)
        End For

    Find Center:
        $center(X_i) \leftarrow$ undefined
        While $H_{X_i} \neq \emptyset$
            Let (key, z) be the element with minimum key in H_{X_i}
            If $\deg[z] = |X_i| - 1$
                $center(X_i) \leftarrow z$; **Break**
            Else
                Remove (key, z) from H_{X_i}
        End While
    If $center(X_i)$ is undefined $\rightarrow$ pick arbitrary node in X_i
End For

Procedure UpdateNode(H, u, w):
    If $u \in H$
        Remove $(W_{max}[u], u)$ from H
        $W_{max}[u] \leftarrow \max\{W_{max}[u], w\}$ {Update global max edge}
        $\deg[u] \leftarrow \deg[u] + 1$ {Update global degree}
        Insert $(W_{max}[u], u)$ into H

so that they lie within the interval [0, 1]. After that, we measured the distance between two points using Euclidean distance.

To evaluate the applicability of our methods in a biological context, where compact clusters are highly desirable [7, 42], we considered two datasets, one of them (Ecoli) is a collection of complete genomes of *Escherichia coli*, downloaded from the NCBI RefSeq database [55]. Each genome was processed using Jellyfish [49], which efficiently computes the frequency of all k-mers occurring in a DNA sequence. This step transforms each genome into a high-dimensional vector representation based on its k-mer profile, providing a natural embedding for subsequent clustering analyses. With these $\mathbb{R}^d$ representations, we followed the same analyses applied to the tabular datasets.

The other dataset comprises single-cell RNA-seq profiles from the Allen Brain [26]. We preprocessed the matrix with standard scRNA-seq steps: cell and gene filtering (min. 1 count), library-size normalization to 10000 counts per cell, and $\log(1 + x)$ transformation. We then identified the top highly variable genes and standardized their expression values to zero mean and unit variance, capping extreme values at 10. For our experiments, we drew a fixed random subset of 5000 cells and used the resulting expression matrix (top 3000 HVGs) as input for clustering. Again, With these $\mathbb{R}^d$ representations, we followed the same analyses applied to the tabular datasets.

To assess the behavior of our methods on textual data, we considered two corpora: the 20 Newsgroups [45] and the Reuters-21578 [46] collections. In both cases, documents were preprocessed into sparse bag-of-words representations, normalized using the TF-IDF scheme with ℓ_2 scaling, and embedded into a low-dimensional latent space using truncated SVD. We then computed pairwise Euclidean distances between document embeddings, yielding dense distance matrices used as input for the clustering experiments.

We further evaluated our methods on two ratings datasets: MovieLens-1M [34] and a book-ratings [70] collection. In both cases, user–item rating matrices were constructed and normalized to mitigate scale differences across users and items. Missing ratings were imputed as the average of the item, and the resulting matrices were projected into a low-dimensional continuous space using principal component analysis (PCA). These continuous representations were then treated as points in $\mathbb{R}^d$ and analyzed with the same clustering procedures applied to the other datasets.

Table 4 summarizes the datasets used, where n denotes the number of data points and d the number of attributes per point.

Table 4: Datasets: n is the number of points and d is the number of attributes.

Dataset	n	d	Source
Airfoil	1501	5	[13]
Anuran	7193	22	[19]
Banknote	1371	5	[48]
Concrete	1028	8	[69]
Geographical Music	1057	116	[72]
Ozone	3694	72	[71]
Parkinsons	5874	19	[62]
Qsarfish	906	6	[8]
Rice	3809	7	[1]
Wine	4898	11	[20]
Reuters	5000	300	[46]
News20	5000	300	[45]
Book Rating	5000	300	[70]
Movie Lens	6040	300	[34]
Allen Brain	5000	3000	[26]
Ecoli	4514	511	[55]
Beans	13611	16	[2]
Mice	552	77	[36]
Iris	150	4	[30]
PenDigits	5000	16	[4]
BreastCancer	683	9	[67]
LetterRecognition	5000	16	[59]

Orkut Dataset. For the scalability experiment, we used the `orkut` dataset, we define edge distances by setting $\texttt{dist}(u, v) = \log\big(1 + \min\{d(u), d(v)\}\big)$, where $d(x)$ denotes the degree of vertex x. This assignment of distances is a simple heuristic intended to assign more importance (smaller distances) to friendships between users with relatively few connections, reflecting the social-network nature of Orkut. Importantly, the primary goal of this dataset is to assess the scalability of `CRM` and `CRM`*, and the particular choice of edge distances is secondary to this goal.

G.2 Evaluated Methods

In order to assess the quality of our methods `RMM`* and `CRM`*, we compared them with the following ones:

- `minimax`. This linkage method uses $\min\{\texttt{dmax}(x, A \cup B) | x \in A \cup B\}$ as the distance between clusters A and B. It can be seen as a greedy approach to minimizing the k-center cost function, and it runs in $O(n^3)$ time.
- `centroid`. The distance between two clusters A and B is given by the distance of their centroids. The distance is calculated via the Lance-Williams formula.
- `complete-linkage`. It uses the maximum distance from a point in A to a point in B as the distance between two clusters A and B. This rule greedily tries to minimize the diameter, which is closely related to the k-center cost function.
- `single-linkage`. It uses the minimum distance from a point in A to a point in B as the distance between two clusters A and B.
- `average-link`. It uses the average distance between points in A and B, that is,

  $$\frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} \texttt{dist}(a, b)$$

  as the distance between two clusters A and B.
- `Gonzalez`. Differing from the above algorithms, this method does not produce HAC. For a given value of k, this algorithm obtains in $O(kn)$ time a k-clustering, whose k-center cost function is at most twice the optimum [31].
- `Dasg`. This method produces a hierarchical clustering that guarantees a $8-$approximation to the optimal k-center cost function for all values of k simultaneously [22].
- `HDBSCAN`. This method is a hierarchical density-based clustering method that builds a hierarchy of dense regions and extracts stable clusters while automatically identifying noise and outliers.
- `ship-hst`. This is a modern method [27] that embeds the original points in a tree metric (HST) with provable $O(\log n)$ distortion, and then optimally solves the k-center for the embedding points.

All linkage methods but `centroid` were implemented using the NN-Chain framework [52, 54]. The key difference between `minimax` and the others implemented via NN-Chain lies in the update of `dist`. After merging clusters A and B, `minimax` requires $O(n^2)$ time to compute $\texttt{dist}(A \cup B, C)$ for every cluster C, whereas the other algorithms perform this step in $O(n)$ time.

`ship-hst` is obtained by running the code provided in [27] (`https://github.com/pasiweber/SHiP-framework/`) with the options `treetype`=HST and hierarchy=0 (k-centers).

Some of the clustering criteria that we evaluate depend on representatives. For those that do not compute representatives from the input set, we use the center of a cluster C as its representative. We note that `centroid` does not compute a representative from the input set, so we adopted this same strategy.

G.3 Running Time

Table 2 (in the main body) reports the average running times of all evaluated methods, measured over datasets whose total execution exceeded one second. The timing includes both the dendrogram

construction and the computation of cluster representatives. Here, we provide a more detailed breakdown.

Execution time was measured from the moment the distance matrix became available until the completion of all clusters and their corresponding representatives. Each method was executed tree times per dataset.

Table 5 shows the average ($\pm$ standard deviation) time required for dendrogram computation, while Table 6 presents the corresponding results for the entire computation (dendrograms+representatives). The observed standard deviations were small, confirming that the reported averages are stable and representative.

Table 5: Average time in seconds $\pm$ standard deviation (1 decimal) for each dataset and method to compute dendrogram.

Dataset	Average	Complete	CRM*	Minimax	RMM*	Single
Allen Brain	3.3 ± 0.1	4.4 ± 1.0	4.1 ± 0.1	64.9 ± 0.7	5.7 ± 0.1	3.5 ± 0.1
Anuran	6.5 ± 1.0	7.1 ± 0.2	7.1 ± 0.1	39.5 ± 0.8	10.0 ± 0.1	6.0 ± 0.2
BookRatings	3.4 ± 0.0	3.9 ± 0.1	4.0 ± 0.1	1075 ± 48.8	5.7 ± 0.1	3.5 ± 0.1
Ecoli	2.7 ± 0.1	3.1 ± 0.2	3.3 ± 0.2	13.1 ± 0.2	4.4 ± 0.2	3.0 ± 0.2
MovieLens	4.8 ± 0.1	6.0 ± 0.1	5.8 ± 0.1	1214 ± 27.7	8.4 ± 0.1	5.2 ± 0.1
News20	3.5 ± 0.0	3.9 ± 0.1	4.2 ± 0.0	638 ± 35.6	5.8 ± 0.1	3.6 ± 0.1
Ozone	1.3 ± 0.0	1.6 ± 0.0	1.6 ± 0.0	6.0 ± 0.1	2.0 ± 0.0	1.8 ± 0.2
Parkinsons	4.0 ± 0.1	4.7 ± 0.0	5.5 ± 1.0	22.1 ± 1.2	6.6 ± 0.1	5.8 ± 1.1
Reuters	3.6 ± 0.3	3.9 ± 0.2	4.4 ± 0.2	41.7 ± 0.5	6.0 ± 0.2	3.7 ± 0.1
Rice	1.9 ± 0.1	2.2 ± 0.1	2.4 ± 0.1	8.1 ± 0.2	3.1 ± 0.1	2.2 ± 0.1
Wine	5.7 ± 0.1	7.0 ± 1.2	7.0 ± 0.2	27.6 ± 1.3	9.4 ± 0.1	6.8 ± 1.2

Table 6: Average time in seconds $\pm$ standard deviation for each dataset and method to compute dendrogram and representatives (when needed).

Base	Average	Complete	CRM*	Minimax	RMM*	Single
AllenBrain	5.4 ± 0.3	6.5 ± 1.3	6.4 ± 0.3	64.9 ± 0.7	7.9 ± 0.2	6.2 ± 0.2
Anuram	9.9 ± 1.6	11.1 ± 1.0	10.6 ± 0.4	39.5 ± 0.8	14.1 ± 1.0	10.7 ± 1.0
BookRating	5.7 ± 0.0	5.8 ± 0.1	6.8 ± 0.1	1075 ± 48.8	8.5 ± 0.1	6.4 ± 0.1
Ecoli	4.2 ± 0.2	4.6 ± 0.2	4.8 ± 0.3	13.1 ± 0.2	5.9 ± 0.2	4.8 ± 0.3
MovieLens	8.6 ± 0.4	8.6 ± 0.3	9.9 ± 0.4	1214 ± 27.7	12.4 ± 0.4	9.3 ± 0.5
News20	5.7 ± 0.1	5.9 ± 0.1	6.3 ± 0.1	638 ± 35.6	8.2 ± 0.2	6.4 ± 0.2
Ozone	2.4 ± 0.0	2.9 ± 0.1	3.5 ± 0.1	6.0 ± 0.1	3.8 ± 0.1	3.9 ± 0.3
Parkinsons	6.6 ± 0.3	7.4 ± 0.3	8.3 ± 1.0	22.1 ± 1.2	9.4 ± 0.3	8.8 ± 1.3
Reuters	5.5 ± 0.4	5.8 ± 0.2	6.4 ± 0.2	41.7 ± 0.5	8.0 ± 0.1	6.0 ± 0.1
Rice	3.1 ± 0.2	4.1 ± 0.1	5.3 ± 0.1	8.1 ± 0.2	5.6 ± 0.2	5.0 ± 0.2
Wine	8.7 ± 0.3	9.8 ± 1.3	9.9 ± 0.3	27.6 ± 1.3	12.3 ± 0.2	10.2 ± 1.3

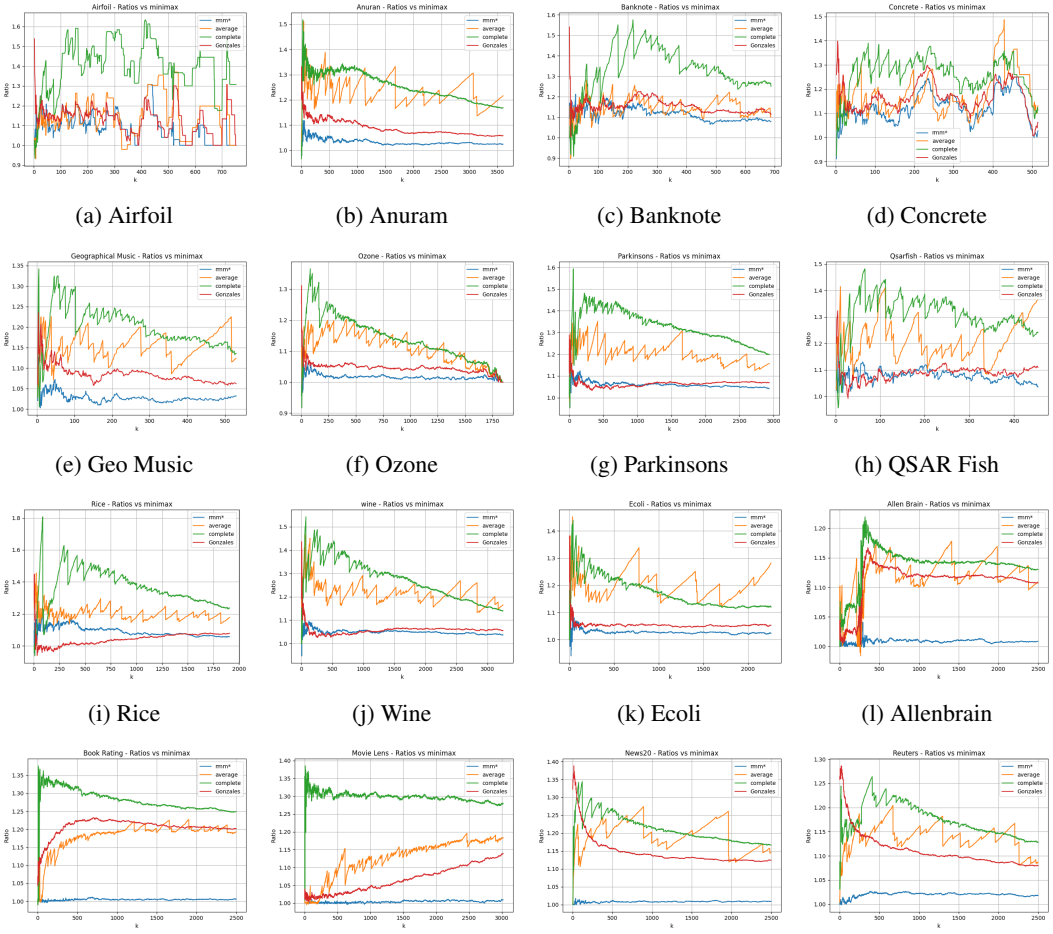
The quadratic-time methods exhibit similar behavior. While `minimax` is relatively competitive on some datasets—running two to three times slower than the quadratic-time methods—the difference becomes substantial (around 100×) on high-dimensional datasets such as MovieLens, Book Ratings, Reuters, and News20.

G.4 k-center objective

Due to space constraints, in the main body of the paper we presented only four datasets in the comparison of the methods with respect to the k-center cost. Here, we provide the corresponding plots and analysis for all datasets.

For clarity of visualization, in Figure 3, we first show only the top-performing methods and, then, in Figure 4, we repeat RMM* and include the remaining methods. As in the body, the results are normalized by that of `minimax`. For `ship-hst`, due to the inherent randomness of the algorithm and the lack of a seed-fixing mechanism in the implementation provided by [27], the reported values

represent the average of five separate executions. Moreover, for some datasets (with large d), we were unable to run `ship-hst` due to memory requirements.



(a) Airfoil (b) Anuram (c) Banknote (d) Concrete

(e) Geo Music (f) Ozone (g) Parkinsons (h) QSAR Fish

(i) Rice (j) Wine (k) Ecoli (l) Allenbrain

(m) Book Ratings (n) MovieLens (o) Newsgroups (p) Reuters

Figure 3: Ratio between the k-center cost of the best performing methods and that of `minimax` for each value of k, for multiple datasets

G.5 ARI for Synthetic Data

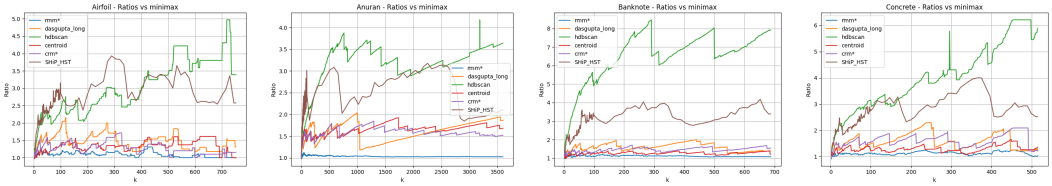
To evaluate the robustness of the clustering algorithms to noise, we generated synthetic datasets composed of well-separated Gaussian clusters contaminated with uniformly distributed noise. Each instance consists of k clusters in $\mathbb{R}^d$, constructed as follows. We first sample k cluster centers $\mu_1, \ldots, \mu_k$ independently from the uniform distribution over the hypercube $[-10, 10]^d$. For each center μ_i, we then generate n data points independently from the Gaussian distribution $\mathcal{N}(\mu_i, \sigma^2 I_d)$, where I_d is the d-dimensional identity matrix and σ controls the cluster spread.

To adapt σ to the geometry of the instance, we set

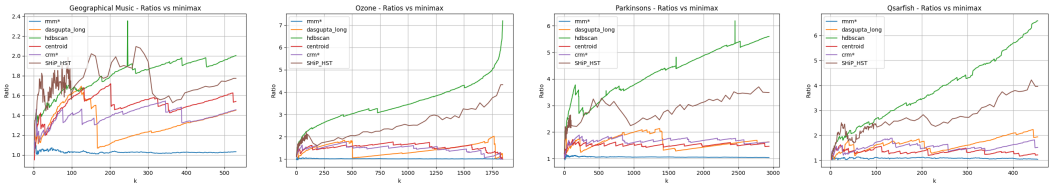
$$\sigma = \alpha \cdot \min_{i \neq j} \|\mu_i - \mu_j\|,$$

for a fixed constant $\alpha \in (0, 1)$. This ensures that clusters remain well-separated while allowing controlled overlap depending on the value of α.

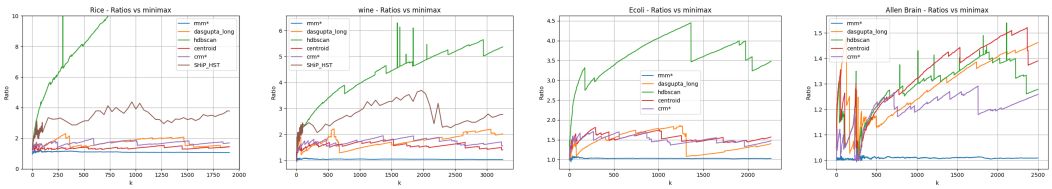
Noise is introduced by adding m points drawn independently from the uniform distribution over $[-10, 10]^d$. The final dataset therefore contains $kn + m$ points, and the level of contamination is quantified by the noise ratio $m/(kn)$. This construction allows us to systematically test the performance of clustering algorithms under increasing noise levels.



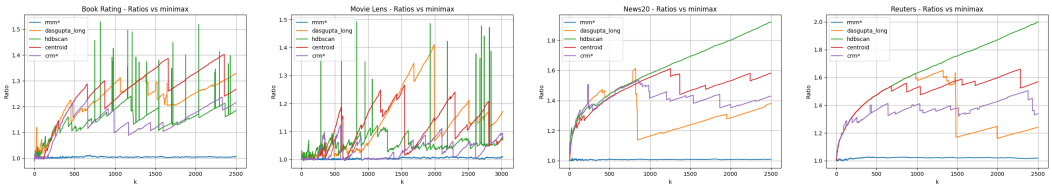
(a) Airfoil (b) Anuram (c) Banknote (d) Concrete



(e) Geo Music (f) Ozone (g) Parkinsons (h) QSAR Fish



(i) Rice (j) Wine (k) Ecoli (l) Allenbrain



(m) Book Ratings (n) MovieLens (o) Newsgroups (p) Reuters

Figure 4: Ratio between the k-center cost of each method and that of `minimax` for each value of k, for multiple datasets

To assess clustering quality, we report the Adjusted Rand Index (ARI) [37]. It is computed only over the non-noise points, with the ground-truth clustering defined by assigning two points to the same cluster if and only if they were generated from the same Gaussian center.

After generating datasets under varying noise levels and dimensions, we compared all hierarchical methods using ARI. Figures 5 - 10 show the ARI achieved for several values of d. In all plots, the x-axis corresponds to the number of noise points m, while the y-axis reports the ARI.

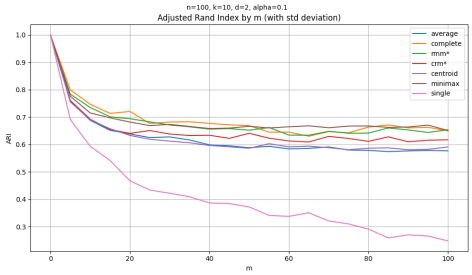


Figure 5: ARI for d = 2

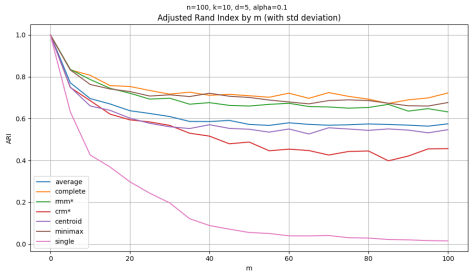


Figure 6: ARI for d = 5

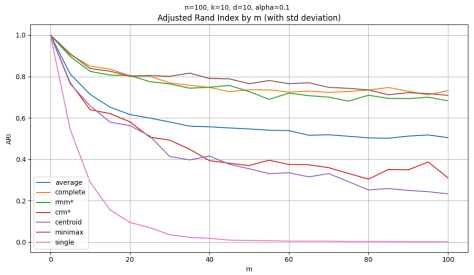


Figure 7: ARI for d = 10

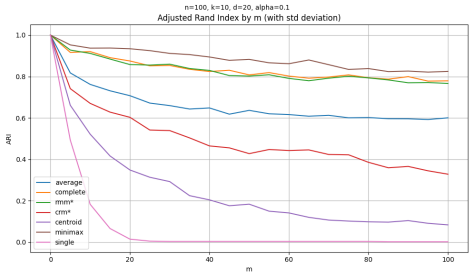


Figure 8: ARI for d = 20

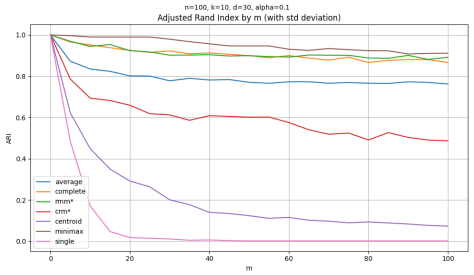


Figure 9: ARI for d = 30

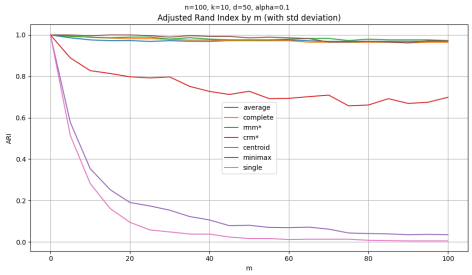


Figure 10: ARI for d = 50

The results show that RMM* behaves very similarly to complete-linkage, a method well known for its robustness to noise [61]. This similarity is encouraging, as it suggests that RMM* inherits the desirable stability of complete-linkage while maintaining stronger theoretical guarantees. Hence, RMM* not only provides provable approximation bounds and compact clusters but also offers consistent empirical robustness across noisy datasets.

These experiments also suggest that CRM* is competitive with `centroid`, and more robust to noise than `single-linkage`.

G.6 Inversions of centroid algorithm

Table 7 shows the number of inversions produced by the `centroid` algorithm for the datasets under evaluation. We observe that for some datasets the percentage of inversions is higher than 20%.

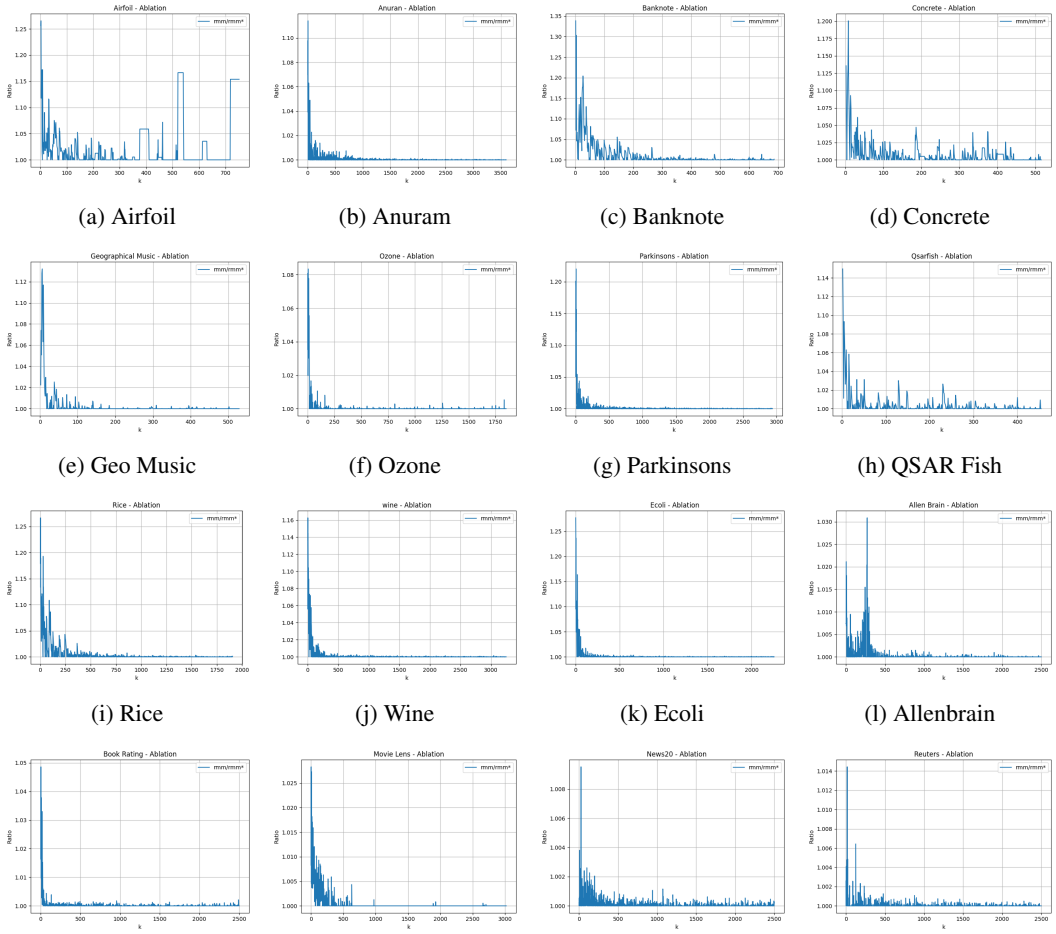
Table 7: Number of inversions of `centroid` method per dataset

Dataset	n	Number of inversions	Inversions ratio (%)
Airfoil	1501	10	0.7
Anuram	7194	1577	21.9
Banknote	1372	58	4.2
Concrete	1029	27	2.6
Geographical Music	1058	269	25.5
Ozone	3695	473	12.8
Parkinsons	5875	642	10.9
Qsarfish	907	49	5.4
Rice	3810	228	6.0
Wine	6497	623	9.6
Ecoli	4514	775	17.2
AllenBrain	5000	1195	23.9
BookRatings	5000	371	7.4
MovieLens	6040	275	4.6
News20	5000	672	13.5
Reuters	5000	997	20.0

G.7 Ablation Study

In this ablation study, we evaluate how replacing the representatives obtained during the construction of the dendrogram with cluster centers computed by Algorithm 4 affects the k-center objective.

The plots in Figure 11 show a consistent improvement in the k-center objective when cluster centers are used. This gain is particularly significant for small values of k. These findings empirically justify the use of Algorithm 4 to compute cluster centers after the dendrogram is built by RMM*.



(a) Airfoil (b) Anuram (c) Banknote (d) Concrete

(e) Geo Music (f) Ozone (g) Parkinsons (h) QSAR Fish

(i) Rice (j) Wine (k) Ecoli (l) Allenbrain

(m) Book Ratings (n) MovieLens (o) Newsgroups (p) Reuters

Figure 11: Ratio between the k-center cost of RMM(a version of RMM* that does not compute the centers) and that of RMM* across varying values of k for multiple datasets.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: There is a section in the introduction.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification:

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We add the relevant information in the appendix, and we send our code in the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide our code in a .zip file (supplementary material)

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We detail the number of clusters employed in our ARI experiments and how we preprocessed the datasets.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provided graphs containing all relevant information to assess the strength of our results, and we provide standard deviations associated with our running time experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We specify it in the Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer:[Yes]

   Justification:

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper presents algorithmic contributions to hierarchical clustering, focusing on improving computational efficiency and theoretical understanding rather than on specific applications. Any societal impact would therefore be indirect and depend on the deployment context.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We include the documentation for our code in the .zip file attached as supplementary material.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.