FAST LINKAGE ALGORITHM FOR k-MEDIANS HIERARCHICAL CLUSTERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Hierarchical linkage methods are widely used due to their simplicity, interpretability, and ability to produce full clusterings across all values of k. The k-medians objective is a fundamental measure of cluster compactness, capturing average distances to representative points. Despite its importance, existing hierarchical linkage methods that align with this objective are either computationally expensive or lack scalability.

We introduce two simple hierarchical linkage algorithms, `r-msi` and `f-msi`, which, to the best of our knowledge, are the first linkage methods aligned with the k-medians objective to achieve subcubic time complexity. Our methods can be seen as efficient variations of `msi`, a linkage method that runs in cubic time. `r-msi` runs in $O(n^2 \log n)$ time while `f-msi` runs in $O(n^2)$ time and produces inversion-free dendrograms. Notably, `f-msi` admits an implementation for datasets represented as weighted graphs that runs in $O((m + n\sqrt{m}) \log m)$ time, where n and m are the number of vertices and edges of the graph, as well as a subquadratic-time implementation in which the merge rule is γ-approximated.

Experiments on several datasets from different domains show that our algorithms present strong empirical performance, achieving clustering quality close to `msi` while being up to 75 times faster. Moreover, `f-msi` scales to graph datasets with over 10^7 edges.

1 INTRODUCTION

Given a dataset $\mathcal{X}$ of n points, a hierarchical clustering is a sequence of n successive partitions. This sequence can be viewed as a process that starts with n singleton clusters—each point in $\mathcal{X}$ forms its own cluster—and at each step two existing clusters are merged, yielding a new partition with one fewer cluster.

This iterative procedure induces a full binary tree with n leaves: each leaf corresponds to a point of $\mathcal{X}$, while every internal node represents a cluster created during the merging process. The resulting tree, known as a *dendrogram*, offers a convenient visual representation of how clusters form across different scales and helps reveal potentially meaningful structure in the data.

Hierarchical clustering is commonly used for exploratory data analysis, with applications in several fields, as finance and biology (Eisen et al., 1998; Tumminello et al., 2010). Linkage methods such as *Ward*, *single linkage*, *complete linkage*, and *average linkage* (Murtagh and Contreras, 2017) are popular due to their simplicity and interpretability. At each iteration, these methods use a linkage function (a function that maps a pair of clusters into a real number) to decide which clusters must be merged.

Existing linkage functions are largely heuristic and are designed to favor compact and/or well-separated clusters. Natural notions of compactness include minimizing the average distance or average squared distance of points to cluster representatives, which are generally less sensitive to outliers than diameter-based criteria.

Arguably, Ward's method (Ward, 1963) is the usual choice for building hierarchical clusterings aimed at producing compact clusters under a squared-distance criterion. Rather than focusing on maximum

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

distances, Ward favors clusters with small average squared dispersion around their representative. This method can be viewed as a hierarchical analogue of the k-means objective, since it greedily merges the pair of clusters that yields the smallest increase in within-cluster Sum of Squared Error. Ward runs in $O(n^2)$ time, which makes it suitable for datasets of moderate size. However, it inherits key limitations of k-means-type methods, among them: (i) Although it provides a centroid for each cluster, this centroid is often not a point from the input set, which is not desirable in terms of interpretability; (ii) Ward's criterion is intrinsically linked to squared Euclidean geometry, so it is not directly applicable to other relevant metrics (e.g., cosine, Jaccard) or to datasets whose points do not lie in $\mathbb{R}^d$ and (iii) it can be quite sensitive to outliers

These limitations motivate the use of representative-based methods that operate directly on medoids rather than centroids, and that are compatible with arbitrary metrics. One prominent example is Partitioning Around Medoids (PAM) method (Kaufman and Rousseeuw, 1990).

The Minimum Sum Increasing (Schubert, 2021), `msi` for short, is a hierarchical analogue of PAM: at each iteration, it greedily merges the pair of clusters whose union produces the smallest increase in the k-medians objective (sum of the distances of the points to their closest representatives). However, no subcubic implementation is known for this method, which makes it significantly slow for datasets of moderate size. Moreover, `msi` may present inversions (see Section 2), which compromises the interpretability of the resulting dendrogram, since merges can occur at smaller distance levels after higher ones.

Motivated by this scenario, we design fast and simple linkage methods with strong practical performance for the k-median objective.

1.1 Our Contributions

Our contributions are as follows:

- **Algorithmic.** We first introduce `r-msi`, an efficient variant of `msi` that runs in $O(n^2 \log n)$ time. While the linkage distance of `msi` depends on the medoid of $A \cup B$, the linkage distance of `r-msi` depends only on the medoids of A and B, allowing for more efficient computation. Building on `r-msi` we develop `f-msi`, our main algorithm. It employs a surrogate function that remains aligned with the k-medians objective while being compatible with the NN-Chain framework (Murtagh, 1985). This yields quadratic running time, inversion-free dendrograms, and a scalable implementation for datasets represented as weighted graphs that runs in $O((m + n\sqrt{m}) \log(m + n))$ time, where n and m are, respectively, the number of vertices and edges of the graph. In addition, using the techniques developed by Abboud et al. (2019), we show that `f-msi` admits a subquadratic variant for points in $\mathbb{R}^d$ that merges at each iteration two clusters whose distance is within a factor of γ of the minimum `f-msi` distance between any pair of clusters.
- **Empirical.** We compared our algorithms against `complete-linkage`, `average-link`, `msi`, `PAM`, `TeraHAC`(Dhulipala et al., 2023) and the method of Cohen-Addad et al. (2021) on several datasets across different domains, including textual, biology and rating. Our methods outperform all other hierarchical methods in terms of k-medians cost but `msi` (the cubic-time method), for most datasets and values of k. Although they do not surpass `msi`, their performance is close while being up to 75 times faster on some datasets of moderate size. In terms of scalability, we show that our graph-oriented implementation scales to graph datasets with over 10^7 edges, while the subquadratic variant produces clusterings with k-medians costs close to those of the exact version and can process datasets with two million points in a few minutes.

Perhaps surprisingly, the simple linkage rule of `f-msi`, namely, the distance between the cluster representatives weighted by the size of the smaller cluster, is sufficient to achieve this combination of efficiency, scalability, and empirical performance.

Limitations. From a theoretical perspective, we do not know whether our methods admit strong worst-case approximation guarantees. It is worth noting that widely used linkage approaches lack strong worst-case approximations. For example, `Ward`'s method has a polynomial lower bound for the k-means objective (Großwendt et al., 2019).

1.2 Related Work.

There is extensive research on hierarchical agglomerative clustering, with several surveys and monographs covering the topic in depth (Murtagh and Contreras, 2017; Ran et al., 2023). Below, we highlight practical studies that focus on k-medians objective and scalable linkage methods.

Practical methods for the k-median objective. The Minimum Sum Increasing (`msi`), discussed above, was evaluated against the linkage methods `min-sum` (Schubert, 2021), `medoids` (Herr et al., 2016), and `minimax`(Bien and Tibshirani, 2011), achieving substantially lower k-medians costs (Schubert, 2021). Except for `medoids`, which obtained the worst results, no subcubic implementations are known for these methods. In contrast, our approaches achieve k-medians costs close to those of `msi` in subcubic time by enforcing representative inheritance and, in the case of `f-msi`, using a surrogate linkage compatible with the NN-chain framework.

Cohen-Addad et al. (2021) proposes an algorithm for hierarchical clustering targeting the k-medians objective on Euclidean spaces, and shows how to parallelise it efficiently. The algorithm runs in nearly linear time when the input is a set of n points in $\mathbb{R}^d$, and it has provable polylogarithmic approximation guarantees. The algorithms we present here are more general since they are not restricted to points lying in $\mathbb{R}^d$. Moreover, our methods exhibit substantially lower k-median cost in our experiments.

Scalable linkage methods. The implementation of `f-msi` for sparse graphs, is related to the line of work that develops implementations for classical linkage methods, and variants, for large datasets (Abboud et al., 2019; Dhulipala et al., 2021; 2022; 2023). For graphs datasets, Dhulipala et al. (2021) obtains a $\tilde{O}(m + n\sqrt{m})$ running time implementation of `average-link` and an $\tilde{O}(m)$ running time implementation for a class of linkage methods that satisfy the reducibility condition and are *triangle-based*. Their framework does not directly apply to `f-msi`.

Abboud et al. (2019) considers variants of classical linkage methods that, instead of merging the two most similar clusters at each iteration, merge two clusters whose similarity is within a $(1 + \epsilon)$ factor of the maximum one. This approach is also investigated in Dhulipala et al. (2022; 2023) for sparse graphs datasets. Our subquadratic version of `f-msi` is an adaptation of the variant of `Ward`'s method discussed in Abboud et al. (2019). However, the simplicity of `f-msi` allows us to shave off a logarithmic factor from the time complexity achieved for `Ward`'s variant.

2 Preliminaries

The input for our clustering methods is a set $\mathcal{X}$ of n points and a distance function `dist` that maps each pair of points into a non-negative number.

For a point p and a cluster C, we define $\texttt{dsum}(p, C) := \sum_{x \in C} \texttt{dist}(p, x)$. We use r_C to denote the representative of C, that is, the point in C that represents it. The *medoid* m_C of a cluster C is the representative that yields the minimum cost, that is, $m_C := \arg\min_{c \in C} \texttt{dsum}(c, C)$.

A k-clustering is a clustering with k clusters. The k-medians cost of a clustering $\mathcal{C} = (C_1, \ldots, C_k)$ is defined as

$$\texttt{cost}(\mathcal{C}) = \sum_{i=1}^{k} \texttt{dsum}(m_{C_i}, C_i),$$

A k-clustering is optimal for instance $(\mathcal{X}, \texttt{dist})$ if it achieves the minimum cost among all possible k-clusterings.

Linkage Methods. A pseudo-code for linkage methods based on representatives is shown in Algorithm 2. At each iteration, the two closest clusters in terms of the linkage distance $\texttt{dist}_L$ are merged (lines 4 and 5), and the representative of the new cluster $A \cup B$ is computed (line 6). In our discussion, we say that a cluster is *active* if it belongs to the set `Active` and that a representative is active if it is the representative of an active cluster.

Inversions. For a linkage method $\mathcal{A}$, with linkage distance $\texttt{dist}_L$, let A_i and B_i be the clusters merged at the ith iteration of its execution. We say that $\mathcal{A}$ produces an inversion if $\texttt{dist}_L(A_j, B_j) > \texttt{dist}_L(A_{j+1}, B_{j+1})$, for some j.

Algorithm 2 Linkage Algorithms based on 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: $(A, B) \leftarrow$ clusters in `Active` for which $\texttt{dist}_L(A, B)$ is minimum
5: Remove clusters A and B from `Active` and then add $A \cup B$ to `Active`
6: Compute a representative for $A \cup B$

3 THE r−msi ALGORITHM

r−msi follows the pseudo-code presented in Algorithm 2. It uses the linkage function

$$\texttt{dist}_r(A, B) := \tag{1}$$

$$\min\{\texttt{dsum}(m_A, A \cup B), \texttt{dsum}(m_B, A \cup B)\} - \texttt{dsum}(m_B, B) - \texttt{dsum}(m_A, A) = \tag{2}$$

$$\min\{\texttt{dsum}(m_A, B) - \texttt{dsum}(m_B, B), \texttt{dsum}(m_B, A) - \texttt{dsum}(m_A, A)\}, \tag{3}$$

and the representative of $A \cup B$ (line 6) is its medoid

To explain the motivation of our linkage distance, we first consider the Minimum Sum Increasing (msi), a linkage method presented in Schubert (2021) that employs a very natural greedy rule that at each step merges the two clusters that yield the minimum increasing on the k-medians cost. Its linkage distance $\texttt{dist}_{\texttt{msi}}$ is given by

$$\texttt{dist}_{\texttt{msi}}(A, B) := \texttt{dsum}(m_{A\cup B}, A \cup B) - \texttt{dsum}(m_A, A) - \texttt{dsum}(m_B, B).$$

The problem with this function is that it needs to recompute the medoid for each pair of active clusters, which is expensive. By using the best candidate between the medoids of A and B (equation 1) as the representative of $A \cup B$, we obtain r−msi distance function.

Time Complexity. r−msi can be implemented in $O(n^2 \log n)$ time. In a nutshell, the implementation follows the approach of Day and Edelsbrunner (1984). For each active cluster C, it maintains a binary heap P_C containing one entry for each active cluster C' that was created before C. The key associated with C' in the heap P_C is $\texttt{dist}_r(C, C')$. To determine the next pair of clusters to merge, it selects the minimum key across all heaps and updates them in $O(n \log n)$ time (See Appendix A).

Theoretical Analyses. We present two theoretical analyses of r−msi, both assuming that the points lie in a metric space. Proofs can be found in Appendix C.

We first show that r−msi achieves a constant approximation for instances satisfying the natural weak δ-center separation. This separability condition was used in Ackermann et al. (2010) to analyse Ward linkage algorithm. The definition for δ-center separation presented below uses points in a metric space rather than in $\mathbb{R}^d$.

Definition 3.1. An instance $(\mathcal{X}, \texttt{dist})$ satisfies the weak δ-center separation with respect to some target clustering $C_1, \ldots, C_k$ if there exist centers $c_1, \ldots, c_k \in \mathcal{X}$ such that for each cluster C_j, $j \in [k]$, and for all $i \neq j$, $\texttt{dist}(c_j, c_i) \geq \delta \cdot \max_{x \in C_j} \texttt{dist}(x, c_j)$

Theorem 3.2. *Let $\mathcal{C}^*$ be an optimal k-clustering for the instance $(\mathcal{X}, \texttt{dist})$. If $(\mathcal{X}, \texttt{dist})$ satisfies the weak $(4+\epsilon)$-center separation with respect to $\mathcal{C}^*$, then the cost of the k-clustering produced by* r−msi *is at most* $8\texttt{cost}(\mathcal{C}^*)$.

The next theorem relates msi and r−msi by showing that, if r−msi's merge rule is applied at a given iteration, the resulting clustering is within a constant factor of the clustering obtained by applying the msi rule at the same iteration.

Theorem 3.3. *Given a set of points $\mathcal{X}$ that lie in a metric space, let $\mathcal{C}$ be a partition of $\mathcal{X}$ into $k+1$ clusters. Moreover, let $\mathcal{C}^m$ and $\mathcal{C}^r$ be the k-clusterings that are obtained by merging two clusters of $\mathcal{C}$ according to* msi *and* r−msi *rule, respectively. Then,* $\texttt{cost}(\mathcal{C}^r) \leq 5\texttt{cost}(\mathcal{C}^m)$.

4 THE f−msi ALGORITHM

The main motivation of f−msi is addressing some limitations of r−msi. While r−msi runs in $O(n^2 \log n)$ time, a significant speed-up over msi, it is still slower than classical linkage methods like

`average-link` and `complete-linkage` that run in quadratic time. Moreover, `r-msi` may produce dendrograms with inversions (Section E.2), which is undesirable in terms of interpretability. `f-msi` addresses both limitations while allowing scalable implementations and maintaining the strong empirical performance of `r-msi`.

`f-msi` also follows the pseudo-code presented in Algorithm 2. The distance between two clusters A and B employed by `f-msi` is

$$\texttt{dist}_f(A, B) := \min\{|A|, |B|\}\texttt{dist}(r_A, r_B).$$

Moreover, when A and B are merged, the representative of $A \cup B$ is set to r_A if $|A| \geq |B|$, and it is set to r_B, otherwise.

The derivation of this linkage distance starts with that of `r-msi`. In `r-msi` the representative of a cluster A is its medoid m_A, while in `f-msi` is r_A. Thus, replacing m_A and m_B with r_A and r_B, respectively, in the linkage function employed by `r-msi` we have a related linkage distance

$$\texttt{dist}_{\texttt{msir}}(A, B) := \min\left\{\sum_{x\in B} \texttt{dist}(r_A, x) - \texttt{dist}(r_B, x), \sum_{x\in A} \texttt{dist}(r_B, x) - \texttt{dist}(r_A, x)\right\}$$

We simplify $\texttt{dist}_{\texttt{msir}}$ to obtain a very simple surrogate function that is suitable for scalable implementations. Using the triangle inequality, we can upper bound $|\texttt{dist}(r_A, x) - \texttt{dist}(r_B, x)|$ by $\texttt{dist}(r_A, r_B)$ in the righthand side of the above equation, obtaining:

$$\texttt{dist}_{\texttt{msir}}(A, B) \leq \texttt{dist}(r_A, r_B)\min\{|A|, |B|\}.$$

The rightmost term is exactly the linkage function employed by `f-msi`. Of course, being an upper bound alone does not guarantee that minimizing this surrogate leads to a good merge. In the sequel, we provide further theoretical justification for this choice.

Theoretical Justification of `f-msi`.

We first observe that representative inheritance controls the degradation in medoid quality after a merge. In particular, Proposition 4.1 (proof in Appendix D.1) shows that if the representative of the larger of the two clusters being merged is an α-approximation ($\alpha \geq 1$) to its medoid, then it is also a $(4\alpha + 1)$-approximation to the medoid of the resulting cluster. While this bound does not imply a uniform constant-factor guarantee throughout the entire execution, it provides additional justification for the representative inheritance rule employed by f-msi. See Appendix J for experiments showing that `f-msi` representatives provide a good approximation for medoids on real datasets.

Proposition 4.1. *Let A and B, with $|A| \geq |B|$, be the clusters merged by* `f-msi` *at some iteration. If the inherited representative r_A satisfies* $\texttt{dsum}(r_A, A) \leq \alpha\texttt{dsum}(m_A, A)$*, then* $\texttt{dsum}(r_A, A \cup B) \leq (4\alpha + 1)\texttt{dsum}(m_{A\cup B}, A \cup B)$.

The choice of inheriting the representative of the larger cluster is important. In fact, if the representative of the smaller cluster is inherited instead, the approximation to the medoid of the merged cluster can be poor, as discussed at the end of Section D.1.

Having established that the inheritance rule controls the degradation in representative quality after a merge, we now show that the surrogate linkage itself remains theoretically connected to the k-medians objective. Specifically, Theorem 4.2 (Proof in Appendix D.2) shows that, whenever the inherited representatives are constant-factor approximations to the medoids, merging the pair selected by `f-msi` yields a clustering whose k-medians cost is within a constant factor of that obtained by the original `msi` merge.

Theorem 4.2. *Given a set of points $\mathcal{X}$ that lie in a metric space, let $\mathcal{C}$ be a partition of $\mathcal{X}$ into $k + 1$ clusters obtained along the execution of* `f-msi`*. Assume that for each cluster $C \in \mathcal{C}$, the representative r_C of C satisfies* $\texttt{dsum}(r_C, C) \leq \alpha\texttt{dsum}(m_C, C)$*, where $\alpha \geq 1$. Moreover, let $\mathcal{C}^m$ and $\mathcal{C}^f$ be the k-clustering that are obtained by merging two clusters of $\mathcal{C}$ according to* `msi` *and* `f-msi` *rule, respectively. Then,* $\texttt{cost}(\mathcal{C}^f) \leq (8\alpha + 3)\texttt{cost}(\mathcal{C}^m)$.

At this point, one may ask if Theorem 4.2 holds for any natural linkage function. In fact, for the linkage distances of subcubic time methods as `medoid` (Herr et al., 2016), `complete-linkage` or `average-link`, the cost can increase by an $\Omega(n)$ factor as shown at the end of Section D.2.

Time Complexity. We first state a general condition under which a linkage method admits an efficient implementation via the NN-chain algorithm (Murtagh, 1985):

Theorem 4.3. *[Müllner (2011)] Let* `Link` *be a linkage method, with linkage distance* $\texttt{dist}_L$*. If for all disjoint clusters* A, B *and* C*: (i)* $\texttt{dist}_L(A \cup B, C)$ *can be computed in* $O(T)$ *time (given* $\texttt{dist}_L(A, C)$ *and* $\texttt{dist}_L(B, C)$*), and (ii)* $\texttt{dist}_L(A, B) \leq \min\{\texttt{dist}_L(A, C), \texttt{dist}_L(B, C)\}$ *implies* $\min\{\texttt{dist}_L(A, C), \texttt{dist}_L(B, C)\} \leq \texttt{dist}_L(A \cup B, C)$ *then* `Link` *can be implemented in* $O(n^2 \cdot T)$ *time via the* `NN-chain` *algorithm*

The distance $\texttt{dist}_f$ satisfies the condition (ii) of Theorem 4.3. In fact, assume w.l.o.g. that the representative of $A \cup B$ is the representative r_A of cluster A. Then,

$$\min\{\texttt{dist}_f(A, C), \texttt{dist}_f(B, C)\} \leq \texttt{dist}_f(A, C) = \min\{|A|, |C|\}\texttt{dist}(r_A, r_C) \leq \min\{|A \cup B|, |C|\}\texttt{dist}(r_A, r_C) = \texttt{dist}_f(A \cup B, C)$$

Note that when A and B are merged, $\texttt{dist}_f(A \cup B, C)$ can be calculated in $O(1)$ time for each active cluster C. For that, it is enough to maintain the sizes of the active clusters as well as the distance between their representatives. Therefore, it follows from Theorem 4.3 that `f-msi` can be implemented in $O(n^2)$ time.

We observe that the derivation of $\texttt{dist}_f$ as an upper bound on $\texttt{dist}_{\texttt{msir}}$, as well as Theorem 4.2, relies on the triangle inequality. The quadratic-time implementation itself does not.

Inversions. The following theorem proves that `f-msi` does not produce inversions. In Section E, we present its proof and an example in which `msi` produces inversions.

Theorem 4.4. `f-msi` *does not produce inversions.*

4.1 Scalable Implementations

f-msi for graphs. We show how to implement `f-msi` for datasets represented as graphs with weighted edges in $O((m + n\sqrt{m}) \log(m + n))$ time, where n and m are, respectively, the number of vertices and edges in the graph. In these datasets, nodes that are not adjacent are assumed to have a distance K (e.g. 1) larger than any other distance in the dataset. 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). For example, when the points in $\mathcal{X}$ correspond to sets and `dist` is the Jaccard distance, intersecting sets are connected by an edge and have a distance smaller than 1, whereas disjoint sets are not adjacent and have distance $K = 1$,

The key idea of the graph implementation is that each potential merge along the execution of `f-msi` between clusters A and B, with representatives r_A and r_B, can be modeled by an edge (r_A, r_B) with weight $\texttt{dist}_f(A, B)$. Thus, by always selecting the edge with minimum weight, merging its clusters, and updating the affected edges accordingly, we obtain a graph implementation.

Algorithm 3 shows a pseudo-code of a *lazy* implementation of this approach for a graph G. It keeps the weighted edges associated with the potential merges in a binary heap H. Initially, H contains the (regular) edges of G and one special edge for each isolated vertex in G, modeling a merge between clusters whose representatives are not connected by an edge of G. Initially, the weights of the regular edges and the special edges are equal to the distance of their endpoints and to K, respectively.

The pseudo-code uses C_x to denote the cluster represented by a node x. The algorithm is lazy because when a merge is performed, it does not update the edges corresponding to the affected clusters; it waits until these edges are considered in the loop to handle them properly, that is, discard, merge, or update their weights. In Appendix F, we provide a more detailed explanation of the pseudo-code, and use amortized analysis to prove the following theorem regarding its running time.

Theorem 4.5. *For a graph with* n *vertices and* m *edges, the graph version of* `f-msi` *runs in* $O((m + n\sqrt{m}) \log(m + n))$ *time.*

Subquadratic Approximation. The design of `f-msi` allows a subquadratic approximate version for points in $\mathbb{R}^d$. Given $\gamma > 1$, at each iteration, this version merges a pair of clusters A and B such that $\texttt{dist}_f(A, B) \leq \gamma \min\{\texttt{dist}_f(C, C') | C, C' \in \mathcal{C}\}$, where $\mathcal{C}$ is the set of active clusters at the beginning of the iteration.

Algorithm 3 Graph version of `f-msi` method

while there are at least two active clusters **do**
  Remove the edge e of minimum weight from H
  if e is regular **then**
    Let u and v be the endpoints of e, and assume w.l.o.g. $|C_u| \geq |C_v|$
    if u and v are active representatives **then**
      if $\texttt{key}(e) = \texttt{dist}_f(C_u, C_v)$ **then**
        Merge C_u and C_v; Set u as the representative of $C_u \cup C_v$
        $Z_v = \{z$ is active, z is adjacent to v and z is not adjacent to any active representative$\}$
        For each $z \in Z_v$, add to H a special edge e_z with $\texttt{key}(e_z) = K \cdot |C_z|$
      else
        Reinsert e in the heap with $\texttt{key}(e) = \texttt{dist}_f(C_u, C_v)$
  else
    if the representative v associated with the special edge e is active **then**
      if $\texttt{key}(e) = K \cdot |C_v|$ **then**
        Pick some active representative u, with $u \neq v$
        Merge C_u and C_v; Set u as the representative of $C_u \cup C_v$
      else
        Reinsert e in H with $\texttt{key}(e) = K \cdot |C_v|$

The algorithm is obtained by adapting the subquadratic implementation of Ward's method proposed by Abboud et al. (2019). It relies on Approximated Nearest Neighbors (ANN) queries, requiring a dynamic data structure that maintains the set $\mathcal{X}$ of input points under insertions and deletions, and given a query point q, returns a point $p' \in \mathcal{X}$ satisfying $\texttt{dist}(q, p') \leq \sqrt{\gamma} \min_{s \in \mathcal{X}} \texttt{dist}(q, s)$. Let T be the maximum time required for any insertion, deletion, or query operation. Dynamic LSH-based data structures, as an example, provide such guarantees with high probability, with T sublinear in n (Andoni and Razenshteyn, 2015). For a full presentation of the algorithm, see Appendix H.

As discussed in this appendix, the inheritance of `f-msi` and the form of $\texttt{dist}_f$ allows us to simplify the approach of Abboud et al. (2019), avoiding the bucketing of clusters according to their sizes, and so saving a logarithmic factor on the running time. The following theorem holds.

Theorem 4.6. *Given a set of n points in $\mathbb{R}^d$ and a $\sqrt{\gamma}$-Approximate Nearest Neighbor data structure which supports insertion, deletion and query time in time T, there exists a γ-approximation of* `f-msi` *method running in time $O\left(n \cdot T \cdot \log_{\sqrt{\gamma}}(n\Delta)\right)$, where Δ is the ratio between the largest and smallest distance of two points in the input set.*

5 Experiments

In this section, we present our main experiments. Details about machine and dataset specifications, as well as further experiments, are provided in Appendix I.

We evaluated `r-msi` and `f-msi` against `msi`, `complete-linkage`, `average-link` and `PAM` on multiple datasets across different domains. All these methods can be applied to arbitrary distance functions. We consider two criteria, the k-medians objective (primary goal) and ARI scores (secondary goal). While the primary focus of this work is on hierarchical clustering and linkage methods, we included `PAM` because it is a widely-used non-hierarchical baseline that helps to contextualize the performance of `r-msi` and `f-msi`. For this comparison, we use `FasterPAM` implementation (Schubert and Rousseeuw, 2021) provided by the `kmedoids` Python library (Schubert and Lenssen, 2022).

To assess scalability, we compare the graph version of `f-msi` with `TeraHAC` (Dhulipala et al., 2023), a highly scalable state-of-the-art graph HAC baseline, using its public implementation. We also compare the subquadratic variant of `f-msi` with our implementation of the method proposed by Cohen-Addad et al. (2021), which embeds the input points into the leaves of a Hierarchically Separated Tree (HST) and then computes an optimal k-medians solution in the embedded space.

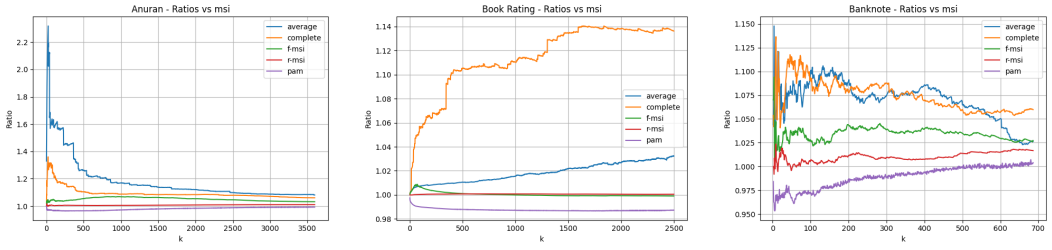
To evaluate the k-medians cost of the algorithms, we use the medoids of the clusters they produce. This computation is already part of `msi` and `r-msi`, whereas for the other methods we add a post-processing phase in which these medoids are computed in $O(n^2)$ time (See Section G.1).

Table 1: Average time in seconds ± standard deviation for each dataset and method to compute dendrogram and medoids (when needed).

Dataset	n	`avg-link`	`complete-link`	`msi`	`r-msi`	`f-msi`
Allen Brain	5000	13.7 ± 0.25	14.5 ± 0.23	103.3 ± 0.25	31.6 ± 0.16	13.3 ± 0.29
Anuran	7193	28.8 ± 0.58	30.9 ± 0.70	90.5 ± 0.25	61.1 ± 0.36	29.8 ± 0.51
BookRatings	5000	14.4 ± 0.10	14.4 ± 0.10	1013.7 ± 1.63	36.5 ± 0.08	13.4 ± 0.06
Ecoli	4514	10.5 ± 0.32	11.3 ± 0.21	36.8 ± 0.06	24.9 ± 0.07	10.7 ± 0.33
MovieLens	6040	22.3 ± 0.49	21.8 ± 0.51	658.7 ± 0.69	60.6 ± 0.18	20.0 ± 0.42
News20	5000	13.7 ± 0.32	14.3 ± 0.16	539.2 ± 0.34	33.3 ± 0.19	13.4 ± 0.13
Ozone	3694	5.9 ± 0.13	6.4 ± 0.20	26.9 ± 0.03	20.0 ± 0.02	6.1 ± 0.13
Parkinsons	5874	18.6 ± 0.38	19.7 ± 0.42	59.1 ± 0.11	41.5 ± 0.17	19.3 ± 0.40
Reuters	5000	13.8 ± 0.14	14.6 ± 0.19	64.8 ± 0.06	31.9 ± 0.01	13.7 ± 0.13
Rice	3809	7.6 ± 0.38	8.0 ± 0.20	24.8 ± 0.02	17.3 ± 0.01	7.6 ± 0.25
Wine	4898	23.4 ± 0.46	24.4 ± 0.36	73.1 ± 0.34	51.2 ± 0.23	23.8 ± 0.17

Running time Table 1 shows the running times of the methods for some datasets in which the quadratic time algorithms took more than 1 second. 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 three times per dataset. All quadratic-time methods exhibit comparable running times. `r-msi` is slower than the quadratic methods by a factor of 2 to 3.5. In contrast, `msi` is up to 75 times slower than the quadratic-time methods for some datasets, consistent with its $O(n^3)$ time complexity. As expected, obtaining solutions for all values of k with `PAM` is substantially more expensive, since, unlike hierarchical methods, it requires solving separate clustering problems for different values of k. Thus, we do not report its times.

k-medians objective. Figure 1 shows the performance of the algorithms for the k-median cost, as a function of k, for 3 representative datasets. The graphs for other 15 datasets are available in Section I.3. The point associated with k (horizontal axis) for a method M represents the ratio between the k-medians cost of the k-clustering built by M and that of the k-clustering built by `msi`. We choose `msi` as a reference because it produced the best results among hierarchical methods for this criterion.



(a) Anuram (b) Bookrating (c) Banknote

Figure 1: Ratio between the k-median cost of each method and that of `msi` for each value of $k < n/2$, for 3 representative datasets.

Our main conclusions are the following:

- `r-msi` and `f-msi` outperformed all hierarchical methods but `msi`, with `r-msi` being better than `f-msi`. Their superiority is consistent on most datasets and values of k. Although they perform slightly worse than `msi`, its substantially lower running time justifies their use in practice;
- As expected, `PAM` outperforms our methods and `msi`, since it is not constrained to building a hierarchy. Importantly, however, the performance gap is not large (often smaller than 10%), despite this additional structural constraint;

ARI and Robustness to Noise. Table 2 presents the ARI scores of the evaluated methods on 10 classification datasets, using k as the number of classes. For `PAM` we present the average of 5 runs and the stdev. The results of `r-msi` and `f-msi` are competitive with those of `msi` and `PAM`.

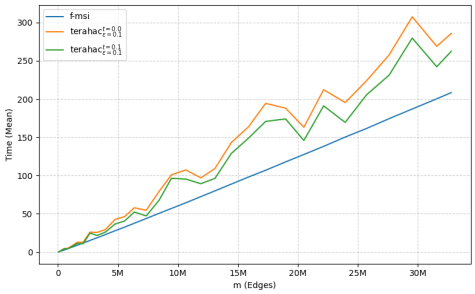
Table 2: ARI for classification datasets

dataset	k	avg-link	complete-link	f-msi	r-msi	msi	PAM
Beans	7	0.32	0.42	0.46	0.63	**0.72**	0.62 (0.04)
Banknote	2	0.08	0.11	**0.60**	0.01	0.08	0.05 (0.05)
Anuram	10	0	**0.86**	0.45	0.44	0.47	0.41 (0.00)
Rice	2	0	0.18	0.63	0.56	0.57	**0.69** (0.00)
Mice	8	0.23	0.14	0.23	0.30	**0.35**	0.24 (0.02)
Iris	3	0.72	0.71	0.67	0.65	0.64	**0.74** (0.00)
PenDigits	10	0.39	0.41	**0.64**	0.59	**0.64**	0.58 (0.02)
LetterRec.	26	0.07	0.07	0.14	0.13	**0.16**	0.12 (0.00)
Parkinson	2	-0.03	-0.03	0.17	**0.20**	0.11	0.17 (0.00)
BreastCancer	2	0.78	0.51	0.80	0.77	0.77	**0.84** (0.00)
Average Score		0.253	0.336	0.479	0.429	0.450	0.459

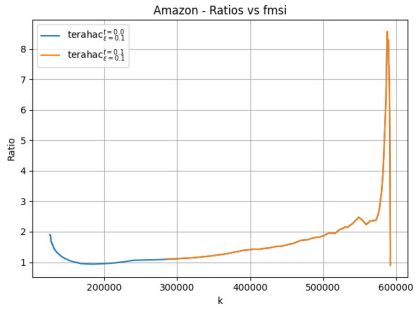
We also evaluated the robustness of the methods on synthetic datasets formed by well-separated Gaussian clusters in $\mathbb{R}^d$ contaminated with uniformly distributed noise points. Our methods clearly outperform `average-link` and `complete-linkage`, as demonstrated in the appendix I.6.

Scalability. To study how `f-msi` scales to large datasets, we obtained a graph with $5,9 \times 10^5$ nodes and 3.3×10^7 edges from `Amazon Reviews` dataset, representing users who review at least one common item. We consider vertex-induced subgraphs of this graph, obtained by uniformly sampling vertices at varying sizes.

For each sample, we run the graph version of `f-msi` and measure the running time. The density m/n^2 of each of these samples is $\approx 10^{-4}$, so the graphs are very sparse. In Figure 2, we compare our method with `TeraHAC`, using its recommended parameters. Our method is competitive in terms of running time and achieves lower k-medians costs. This is consistent with `TeraHAC` approximating `average-link` rather than targeting the k-medians objective. Interestingly, our graph implementation scales substantially better than its theoretical worst-case bound suggests.



(a) Running time (seconds) versus number of edges.



(b) Ratio between k-medians cost of `TeraHAC` and `f-msi` as a function of k

Figure 2: Comparison between `f-msi` and `TeraHAC` for `Amazon Reviews` dataset.

In Section I.5, we show that a practical implementation of the subquadratic variant of `f-msi` produces clusterings with k-medians cost close to those of `f-msi` and can process datasets with two million points in just a few minutes. In addition, these experiments suggest the subquadratic variant of `f-msi` significantly outperform the method by Cohen-Addad et al. (2021) in terms of k-medians cost.

Practical Guideline. In practice, the choice between `msi`, `r-msi`, and `f-msi` is mainly driven by computational budget. For running times of a few seconds on a standard notebook, `msi` is suitable for a few hundred points, while `r-msi` handles larger instances. For large sparse datasets, we recommend `f-msi`, while its approximate version is a reasonable alternative for large dense datasets. These guidelines are supported by our experiments and running-time analyses, and apply to sequential implementations.

AI USE STATEMENT

In this work, we used generative AI tools to assist with coding, identifying related literature, discussing research ideas, and improving the writing and presentation of the manuscript.

All AI-assisted work was reviewed and verified by the authors, including testing AI-assisted code and independently checking suggested references. We take responsibility for the final content of this work, including text, claims, code, and other artifacts produced with the aid of generative AI.

REPRODUCIBILITY

Our code is available in the supplemental material.

REFERENCES

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

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

Amir Abboud, Vincent Cohen-Addad, and Hussein Houdrouge. Subquadratic high-dimensional hierarchical clustering. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Marcel R Ackermann, Johannes Blömer, Daniel Kuntze, and Christian Sohler. Analysis of agglomerative clustering. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 675–684. SIAM, 2010.

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

Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, pages 793–801. ACM, 2015.

Kevin Atkinson. SCOWL: Spell checking oriented word lists. `http://wordlist.aspell.net/`, 2019. Version 2019.10.06.

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

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

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. ISBN 978-3-540-46415-0.

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

Vincent Cohen-Addad, Silvio Lattanzi, Ashkan Norouzi-Fard, Christian Sohler, and Ola Svensson. Parallel and efficient hierarchical k-median clustering. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=bhdntUKwA1`.

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

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.

W. H. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1, 1984.

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. URL https://proceedings.mlr.press/v139/dhulipala21a.html.

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.

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. doi: 10.1145/3617341. URL https://doi.org/10.1145/3617341.

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. doi: 10.1038/s41467-022-30755-0.

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. ISSN 0027-8424. doi: 10.1073/pnas.95.25.14863.

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

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. doi: 10.1137/1.9781611975482.182. URL https://doi.org/10.1137/1.9781611975482.182.

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. doi: 10.1145/2827872. URL http://dx.doi.org/10.1145/2827872.

Georges Hebrail and Alice Berard. Individual Household Electric Power Consumption. UCI Machine Learning Repository, 2006. DOI: https://doi.org/10.24432/C58K54.

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. ISBN 978-0-9947868-1-4. doi: 10.20380/GI2016.14.

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

Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*, 2024.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985. doi: 10.1007/BF01908075.

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

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.

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.

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

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.

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

Shijie Li, Weiqiang He, Ruobing Bai, and Pan Peng. Average sensitivity of hierarchical k-median clustering. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=6BJFle1f23.

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

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. doi: 10.1093/bioinformatics/btr011.

Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proceedings of the Fourth International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 331–340, 2009. doi: 10.5220/0001787803310340.

F. Murtagh. *Multidimensional Clustering Algorithms*. Physica-Verlag, Heidelberg, Germany, 1985. ISBN 978-3-7908-0324-7.

Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview, ii. *WIREs Data Mining and Knowledge Discovery*, 7(6):e1219, 2017. doi: 10.1002/widm.1219. URL https://doi.org/10.1002/widm.1219.

Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms, 2011. URL https://arxiv.org/abs/1109.2378.

Wagner Markus Neshat, Mehdi and Bradley Alexander. Wave Energy Converters. UCI Machine Learning Repository, 2018. DOI: https://doi.org/10.24432/C5831S.

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. doi: 10.1093/nar/gkv1189.

Xingcheng Ran, Yue Xi, Yonggang Lu, Xiangwen Wang, and Zhenyu Lu. Comprehensive survey on hierarchical clustering algorithms and the recent developments. *Artif. Intell. Rev.*, 56(8):8219–8264, 2023. doi: 10.1007/S10462-022-10366-3. URL https://doi.org/10.1007/s10462-022-10366-3.

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. URL http://dblp.uni-trier.de/db/conf/lwa/lwda2021.html#Schubert21.

Erich Schubert and Lars Lenssen. Fast k-medoids clustering in rust and python. *Journal of Open Source Software*, 7(75):4183, 2022. doi: 10.21105/joss.04183. URL https://doi.org/10.21105/joss.04183.

Erich Schubert and Peter J. Rousseeuw. Fast and eager k-medoids clustering: O(k) runtime improvement of the PAM, CLARA, and CLARANS algorithms. *Information Systems*, 101, 2021. doi: 10.1016/j.is.2021.101804.

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

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

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. ISSN 0167-2681. doi: https://doi.org/10.1016/j.jebo.2010.01.004. Transdisciplinary Perspectives on Economic Complexity.

Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963. doi: 10.1080/01621459.1963.10500845. URL http://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10500845.

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

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

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.

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

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

A TIME COMPLEXITY OF r-msi.

r-msi can be implemented in $O(n^2 \log n)$ time. For that, we need to efficiently maintain $\texttt{dsum}(p, C)$ for each active cluster C produced by r-msi and each point p. Moreover, following the approach of Day and Edelsbrunner (1984), for each active cluster C we keep a binary heap P_C containing one entry for each active cluster C' that was created before C. The key of the entry associated with C' in the heap P_C is $\texttt{dist}_r(C, C')$.

To find the closest active clusters, we spend $O(n)$ time to compare the keys of the roots of the binary heaps associated with the active clusters. Let A and B be the closest active clusters. Then, we spend $O(n \log n)$ time to remove the entries associated with A and B from the binary heaps of the active clusters. Next, we calculate $\texttt{dsum}(p, A \cup B)$ for every p. This is accomplished in $O(n)$ time since $\texttt{dsum}(p, A \cup B) = \texttt{dsum}(p, A) + \texttt{dsum}(p, B)$, and both $\texttt{dsum}(p, A)$ and $\texttt{dsum}(p, B)$ are available. Then, in $O(n)$ time, we obtain the medoid of $A \cup B$. Having all this information, we calculate in linear time $\texttt{dist}_r(A \cup B, C)$ for every active cluster C and, in $O(n)$ time, we create a binary heap $P_{A \cup B}$ for $A \cup B$.

B USEFUL RESULTS ABOUT METRIC SPACES

We present some results that are useful to prove the theorems presented at Sections 3 and 4.

Proposition B.1. *Let $(U, \texttt{dist})$ be a metric space and let $S \subseteq U$. For every point $x \in U$, $|S|\texttt{dist}(x, m_S) \leq 3\texttt{dsum}(x, S)$, where m_S is the medoid of S*

Proof. By the triangle inequality, for every $s \in S$, $\texttt{dist}(x, m_S) \leq \texttt{dist}(x, s) + \texttt{dist}(s, m_S)$. Summing over all $s \in S$, we obtain

$$|S|\texttt{dist}(x, m_S) \leq \sum_{s \in S} \texttt{dist}(x, s) + \sum_{s \in S} \texttt{dist}(s, m_S).$$

Hence,

$$|S|\texttt{dist}(x, m_S) \leq \texttt{dsum}(x, S) + \texttt{dsum}(m_S, S). \tag{4}$$

Since m_S is a medoid of S, $\texttt{dsum}(m_S, S) \leq \texttt{dsum}(s, S)$, for every $s \in S$, Averaging over all $s \in S$, we get

$$\texttt{dsum}(m_S, S) \leq \frac{1}{|S|} \sum_{s \in S} \texttt{dsum}(s, S).$$

Now observe that

$$\sum_{s \in S} \texttt{dsum}(s, S) = \sum_{s,t \in S} \texttt{dist}(s, t).$$

Using the triangle inequality, $\texttt{dist}(s, t) \leq \texttt{dist}(s, x) + \texttt{dist}(x, t)$. Thus,

$$\sum_{s,t \in S} \texttt{dist}(s, t) \leq \sum_{s,t \in S} \big(\texttt{dist}(s, x) + \texttt{dist}(x, t)\big).$$

Rearranging the sums,

$$\sum_{s,t \in S} \texttt{dist}(s, t) \leq 2|S|\texttt{dsum}(x, S).$$

Consequently, $\texttt{dsum}(m_S, S) \leq 2\texttt{dsum}(x, S)$. Substituting into (4) yields

$$|S|\texttt{dist}(x, m_S) \leq \texttt{dsum}(x, S) + 2\texttt{dsum}(x, S) = 3\texttt{dsum}(x, S).$$

□

Proposition B.2. *Let S be a finite subset of a metric space. Then, for every point x in the metric space (not necessarily belonging to S).*

$$\texttt{dsum}(m_S, S) \leq \left(2 - \frac{2}{|S|}\right) \texttt{dsum}(x, S).$$

Proof. Let $s \in S$ be a point minimizing the distance to x. Then,

$$\texttt{dist}(x, s) \leq \frac{1}{|S|} \sum_{y \in S} \texttt{dist}(x, y) = \frac{\texttt{dsum}(x, S)}{|S|}.$$

By the triangle inequality, for every $y \in S \setminus \{s\}$,

$$\texttt{dist}(s, y) \leq \texttt{dist}(s, x) + \texttt{dist}(x, y).$$

Therefore,

$$\begin{aligned}
\texttt{dsum}(m_S, S) \leq \texttt{dsum}(s, S) &= \sum_{y \in S \setminus \{s\}} \texttt{dist}(s, y) \\
&\leq (|S| - 1)\texttt{dist}(s, x) + \sum_{y \in S \setminus \{s\}} \texttt{dist}(x, y) \\
&= (|S| - 2)\texttt{dist}(s, x) + \texttt{dsum}(x, S) \\
&\leq \left(1 + \frac{|S| - 2}{|S|}\right) \texttt{dsum}(x, S) \\
&= \left(2 - \frac{2}{|S|}\right) \texttt{dsum}(x, S).
\end{aligned}$$

Since m_S is a medoid of S, $\texttt{dsum}(m_S, S) \leq \texttt{dsum}(s, S)$, and the result follows. □

Proposition B.3. *Let $(U, \texttt{dist})$ be a metric space and let A and B be two clusters in a metric space. Then,*

$$\min\{\texttt{dsum}(m_A, A \cup B), \texttt{dsum}(m_B, A \cup B)\} \leq 5\texttt{dsum}(m_{A \cup B}, A \cup B)$$

Proof. Assume w.l.o.g. that $|A| \geq |B|$. It follows from Proposition B.2 that $\texttt{dsum}(m_A, A) \leq 2\texttt{dsum}(m_{A \cup B}, A)$. From the triangle inequality we have that

$$\texttt{dsum}(m_A, B) \leq |B|\texttt{dist}(m_A, m_{A \cup B}) + \texttt{dsum}(m_{A \cup B}, B)$$

For every $a \in A$, $\texttt{dist}(m_A, m_{A \cup B}) \leq \texttt{dist}(m_A, a) + \texttt{dist}(a, m_{A \cup B})$. By averaging this bound over all $a \in A$, we get that

$$\texttt{dist}(m_A, m_{A \cup B}) \leq \frac{1}{|A|}(\texttt{dsum}(m_A, A) + \texttt{dsum}(m_{A \cup B}, A)).$$

Thus,

$$\texttt{dsum}(m_A, B) \leq \frac{|B|}{|A|}(\texttt{dsum}(m_A, A) + \texttt{dsum}(m_{A \cup B}, A)) + \texttt{dsum}(m_{A \cup B}, B),$$

and

$$\begin{aligned}
\min\{\texttt{dsum}(m_A, A \cup B), \texttt{dsum}(m_B, A \cup B)\} \leq & \quad (5) \\
\texttt{dsum}(m_A, A \cup B) = & \quad (6) \\
\texttt{dsum}(m_A, A) + \texttt{dsum}(m_A, B) \leq & \quad (7) \\
2\texttt{dsum}(m_{A \cup B}, A) + 2\texttt{dsum}(m_{A \cup B}, A) + \texttt{dsum}(m_{A \cup B}, A) + \texttt{dsum}(m_{A \cup B}, B) \leq & \quad (8) \\
5\texttt{dsum}(m_{A \cup B}, A \cup B) &
\end{aligned}$$

□

C Proofs of Section 3

C.1 Proof of Theorem 3.2

We say that a cluster S is an *inner cluster* if $S \subseteq C_i^*$, for some $i \in [k]$, and that a cluster S is a *maximal inner cluster* with respect to a cluster C_i^* if $S \subseteq C_i^*$ and no cluster T that is active at some point of `r-msi` execution satisfies the following condition $S \subset T \subseteq C_i^*$. The *time* of an active cluster S is the number of the iteration in which S is merged with some other cluster. If S is not merged, then its time is defined as n. Let M_i be the sequence of maximal inner clusters with respect to C_i^* sorted by increasing order of times. Note that ties can only occur among clusters with time n, and they are arbitrarily broken. We use M_i^- to denote the sequence obtained by removing the last cluster in M_i. Moreover, we let $M = M_1 \cup \cdots \cup M_k$ and $M^- = M_1^- \cup \cdots \cup M_k^-$.

In our analyses, we use A_j and B_j to denote the clusters merged at the jth iteration of `r-msi`.

Proposition C.1. *Let I be the set of iterations in which one of the merged clusters is a maximal inner cluster or a non-inner cluster. The cost of k-clustering built by* `r-msi` *is at most*

$$\left(\sum_{i=1}^{k} \sum_{S \in M_i} \texttt{dsum}(m_S, S)\right) + \sum_{j \in I} \texttt{dist}_r(A_j, B_j) \tag{9}$$

Proof. We define $\Delta(A, B) := \texttt{dsum}(m_{A \cup B}, A \cup B) - \texttt{dsum}(m_A, A) - \texttt{dsum}(m_B, B)$. The cost of k-clustering built by `r-msi` is given by

$$\sum_{i=1}^{n-k} \Delta(A_i, B_i) = \sum_{j \notin I} \Delta(A_j, B_j) + \sum_{j \in I} \Delta(A_j, B_j) = \tag{10}$$

$$\left(\sum_{i=1}^{k} \sum_{S \in M_i} \texttt{dsum}(m_S, S)\right) + \sum_{j \in I} \Delta(A_j, B_j) \tag{11}$$

□

To establish the theorem, we need to upper bound both terms of (9). Let m_i^* be the medoid of C_i^*. We have that

$$\sum_{i=1}^{k} \sum_{S \in M_i} \texttt{dsum}(m_S, S) \leq \sum_{i=1}^{k} \sum_{S \in M_i} 2\texttt{dsum}(m_S, S) \leq \tag{12}$$

$$2 \sum_{i=1}^{k} \texttt{dsum}(m_i^*, C_i^*), \tag{13}$$

where the first inequality follows from Proposition B.2 and second from the fact that the sets in M_i are pairwise disjoint.

To upper bound the second term of (9), we associate each $j \in I$ with a pair of clusters (S_j, T_j) that satisfies the following conditions: (i) S_j and T_j are inner clusters with respect C_i^*, for some $i \in [k]$; (ii) S_j and T_j are distinct active clusters at the beginning of iteration j. Under these conditions, the choice of `r-msi` guarantees that $\texttt{dist}_r(A_j, B_j) \leq \texttt{dist}_r(S_j, T_j)$, for every $j \in I$.

For a cluster $A \in M_i^-$, we use $succ(A)$ to denote the successor of A in M_i. We split the set of iterations I into groups F and $\overline{F}$ defined as follows:

- F is the set of iterations in which one of the clusters that is merged belongs to M^-. Let $j \in F$ and assume w.l.o.g. that A_j belongs to M_i^-, for some i. We associate the iteration j with the pair (S_j, T_j) where $S_j = A_j$ and T_j is an active cluster contained in $succ(A_j)$. Note that this pair is well-defined because A_j has a successor in M_i and, when A_j is merged, either $succ(A_j)$ is an active cluster or some proper subset of $succ(A_j)$ is an active cluster.

- $\overline{F} = I - F$. Let $\mathcal{S} = \{(S', succ(S'))|S' \in M^- \text{ and } \texttt{time}(S') = n\}$. Lemma C.4 shows that there is a bijection between the iterations in $\overline{F}$ and the elements of set $\mathcal{S}$. Let $(S'_j, succ(S'_j))$ be the pair associated with $j \in \overline{F}$ via the bijection. We associate each iteration $j \in \overline{F}$ with the pair (S_j, T_j), where S_j and T_j are arbitrarily chosen subclusters of S'_j and $succ(S'_j)$ that are active at iteration j.

The following two proposition are used to establish Lemma C.4.

Proposition C.2. *If an instance satisfies the $(4+\epsilon)$-weak center separation with respect to a clustering $\mathcal{C} = (C_1, \ldots, C_k)$, then $\texttt{dist}(x, y) < \min\{\texttt{dist}(x, z), \texttt{dist}(y, z)\}$ for all $x, y \in C_i$ and $z \in C_j$, with $i \neq j$.*

Proof. For $i = 1, \ldots, k$, let c_i be the center of C_i given by Definition 3.1, and let R_i be the maximum distance of a point in C_i to c_i. It follows from the triangle inequality that $\texttt{dist}(x, y) \leq 2R_i$. Assume w.l.o.g. that $\min\{\texttt{dist}(x, z), \texttt{dist}(y, z)\} = \texttt{dist}(x, z)$. We have that

$$2\max\{R_i, R_j\} + \texttt{dist}(x, z) \geq \texttt{dist}(x, c_i) + \texttt{dist}(x, z) + \texttt{dist}(z, c_j) \geq \tag{14}$$

$$\texttt{dist}(c_i, c_j) \geq (4+\epsilon)\max\{R_i, R_j\}. \tag{15}$$

Thus, $\texttt{dist}(x, z) \geq (2+\epsilon)\max\{R_i, R_j\}$, and it follows that

$$\texttt{dist}(x, y) \leq 2R_i < (2+\epsilon)\max\{R_i, R_j\} \leq \texttt{dist}(x, z) = \min\{\texttt{dist}(x, z), \texttt{dist}(y, z)\}.$$

□

Proposition C.3. *Let A_j and B_j be the clusters merged at iteration $j \in F$. Then exactly one of them belongs to M^-.*

Proof. Assume w.l.o.g. that $A_j \in M_i^-$. First, $B_j \notin M_i^-$, otherwise we would have $A_j \cup B_j \in M_i$, contradicting the fact that A_j is a maximal inner cluster with respect to C_i^*. Moreover, assume that $B_j \in M_\ell^-$ for some $\ell \neq i$. In this case, when A_j and B_j are merged there are active clusters A' and B', with $A' \subset C_i^*$ and $B' \subset C_\ell^*$.

Assume w.l.o.g that $\texttt{dist}_r(A_j, B_j) = \texttt{dsum}(m_{A_j}, B_j) - \texttt{dsum}(m_{B_j}, B_j)$. We have that $\texttt{dist}_r(B_j, B') \leq \texttt{dsum}(m_{B'}, B_j) - \texttt{dsum}(m_{B_j}, B_j)$. Since $\mathcal{C}^*$ satisfies the weak $(4+\epsilon)$-center separation condition, then it follows from Proposition C.2 that $\texttt{dist}(m_{B'}, b) < \texttt{dist}(m_{A_j}, b)$, for every $b \in B_j$, so $\texttt{dist}_r(B_j, B') < \texttt{dist}_r(A_j, B_j)$, which contradicts the choice of `r-msi`. □

Lemma C.4. *There exists a bijection between $\overline{F}$ and the set $\mathcal{S}$.*

Proof. The iterations that do not belong to I are those that involve two clusters contained in a maximal inner cluster. The creation of any cluster C requires $|C| - 1$ merges. Thus, the number of iterations that are not in I is given by

$$\sum_{i=1}^{k} \sum_{C \in M_i} (|C| - 1) = n - \sum_{i=1}^{k} |M_i| = n - |M|.$$

Since the total number of iterations performed by `r-msi` to produce $\mathcal{C}$ is $n - k$, then the number of iterations of I is exactly $(n - k) - (n - |M|) = |M| - k = |M^-|$.

By the previous proposition, each $j \in F$ is associated with exactly one cluster in M^- that is merged by `r-msi` (those with time smaller than n). Thus, the number of iterations in $\overline{F}$ is equal to the number of clusters in M^- that are not merged by `r-msi` (those with time equal to n), establishing the bijection. □

Now, we are ready to bound (9). Fix $j \in I$ and let (S_j, T_j) be the pair associated with iteration j. Let us assume that S_j is a subcluster of some cluster in M_i^- and let m_i^* be the medoid of C_i^*. Then,

$$\texttt{dist}_r(A_j, B_j) \leq \texttt{dist}_r(S_j, T_j) = \tag{16}$$

$$\min\{\texttt{dsum}(m_{S_j}, T_j) - \texttt{dsum}(m_{T_j}, T_j), \texttt{dsum}(m_{T_j}, S_j) - \texttt{dsum}(m_{S_j}, S_j)\} \leq \tag{17}$$

$$\min\{|S_j|, |T_j|\}\texttt{dist}(m_{S_j}, m_{T_j}) \leq \tag{18}$$

$$\min\{|S_j|, |T_j|\}(\texttt{dist}(m_{S_j}, m_i^*) + \texttt{dist}(m_i^*, m_{T_j})) \leq \tag{19}$$

$$3\texttt{dsum}(m_i^*, S_j) + 3\texttt{dsum}(m_i^*, T_j). \tag{20}$$

To see that (17) holds, note that the triangle inequality implies that $\texttt{dist}(m_{S_j}, x) - \texttt{dist}(m_{T_j}, x) \leq \texttt{dist}(m_{T_j}, m_{S_j})$, for every $x \in T_j$, so that $\texttt{dsum}(m_{S_j}, T_j) - \texttt{dsum}(m_{T_j}, T_j) \leq |T_j|\texttt{dist}(m_{S_j}, m_{T_j})$. Similarly, $\texttt{dsum}(m_{T_j}, S_j) - \texttt{dsum}(m_{S_j}, S_j) \leq |S_j|\texttt{dist}(m_{S_j}, m_{T_j})$; (18) follows from the triangle inequality and (19) from Proposition B.1.

By adding this inequality for $j \in I$, and assuming that S_j belongs to $C^*_{i(j)}$, we get that

$$\sum_{j \in I} \texttt{dist}_r(A_j, B_j) \leq \sum_{j \in I} 3\texttt{dsum}(m^*_{i(j)}, S_j) + 3\texttt{dsum}(m^*_{i(j)}, T_j) \leq \tag{21}$$

$$\sum_{i=1}^{k} \sum_{C \in M_i^-} 3(\texttt{dsum}(m_i^*, C) + \texttt{dsum}(m_i^*, succ(C)) \leq 6 \sum_{i=1}^{k} \texttt{dsum}(m_i^*, C_i^*), \tag{22}$$

where the second inequality holds because the clusters in M_i appears at most twice in the sum, and the last inequality follows because the clusters in M_i are pairwise disjoint.

It follows from inequalities (9), (13) and (22) that the cost of the k-clustering produced by `r-msi` is at most $8\texttt{cost}(\mathcal{C}^*)$.

C.2 Proof of Theorem 3.3

Proof. Let A_r and B_r be the pair of clusters in $\mathcal{C}$ merged by `r-msi` and let A and B be the pair merged by `msi`. Moreover, let $f = \texttt{cost}(\mathcal{C}) - \texttt{dsum}(m_A, A) - \texttt{dsum}(m_B, B)$. We have that

$$\texttt{cost}(\mathcal{C}^r) \leq \texttt{cost}(\mathcal{C}) + \texttt{dist}_{\texttt{msi}}(A_r, B_r) \leq \tag{23}$$

$$\texttt{cost}(\mathcal{C}) + \texttt{dist}_r(A_r, B_r) \leq \tag{24}$$

$$\texttt{cost}(\mathcal{C}) + \texttt{dist}_r(A, B) = \tag{25}$$

$$f + \min\{\texttt{dsum}(m_A, A \cup B), \texttt{dsum}(m_B, A \cup B)\} \leq \tag{26}$$

$$f + 5\texttt{dsum}(m_{A \cup B}, A \cup B) \tag{27}$$

where inequality (24) holds due to the choice of `r-msi` and inequality 26 holds due to Proposition B.3.

We also have

$$\texttt{cost}(\mathcal{C}^m) = f + \texttt{dsum}(m_{A \cup B}, A \cup B) \tag{28}$$

Dividing (27) by (28), we establish the result. □

D Theoretical Justification of `f-msi`: Proofs and Examples

D.1 Proof of Proposition 4.1

Proof. It follows from the assumption on r_A and Proposition B.2 that

$$\texttt{dsum}(r_A, A) \leq \alpha\texttt{dsum}(m_A, A) \leq 2\alpha\texttt{dsum}(m_{A \cup B}, A).$$

In addition, the triangle inequality ensures that $\texttt{dist}(r_A, b) \leq \texttt{dist}(r_A, m_{A \cup B}) + \texttt{dist}(m_{A \cup B}, b)$, for every $b \in B$, so

$$\texttt{dsum}(r_A, B) \leq |B|\texttt{dist}(r_A, m_{A \cup B}) + \texttt{dsum}(m_{A \cup B}, B)$$

But $\texttt{dist}(r_A, m_{A \cup B}) \leq \texttt{dist}(r_A, a) + \texttt{dist}(m_{A \cup B}, a)$, for every $a \in A$. Averaging this inequality over $a \in A$, we get

$$\texttt{dist}(r_A, m_{A\cup B}) \leq \frac{1}{|A|}(\texttt{dsum}(r_A, A) + \texttt{dsum}(m_{A\cup B}, A)).$$

Using the above inequality together with the assumption on r_A and the assumption $|B| \leq |A|$, we get

$$\texttt{dsum}(r_A, B) \leq \frac{|B|}{|A|}(\alpha\texttt{dsum}(m_A, A) + \texttt{dsum}(m_{A\cup B}, A)) + \texttt{dsum}(m_{A\cup B}, B) \leq \tag{29}$$

$$2\alpha\texttt{dsum}(m_{A\cup B}, A) + \texttt{dsum}(m_{A\cup B}, A \cup B), \tag{30}$$

It follows that

$$\texttt{dsum}(r_A, A \cup B) \leq (4\alpha + 1)\texttt{dsum}(m_{A\cup B}, A \cup B)$$

□

Choosing the representative of the smallest cluster may be bad. As an example, consider two clusters A and B, with $|A|$ much larger than $|B|$. The points of A and B lie, respectively, in circles of radius 1 centered at $(0, 0)$ and $(D, 0)$ of the Euclidean plane. If the representative of $A \cup B$ is a point r_B in B, then

$$\frac{\texttt{dsum}(r_B, A \cup B)}{\texttt{dsum}(m_{A\cup B}, A \cup B)} \geq \frac{\texttt{dsum}(r_B, A \cup B)}{\texttt{dsum}(r_A, A \cup B)} \geq \frac{|A|(D-2)}{2|A| + (D+2)|B|}$$

As D grows the ratio approaches to $|A|/|B|$, which can be $\Omega(n)$.

D.2 Proof of Theorem 4.2

Proof. Let A_f and B_f be the pair of clusters in $\mathcal{C}$ merged by `f-msi` and let A and B be the pair merged by `msi`. Assume w.l.o.g that $|B_f| = \min\{|A_f|, |B_f|\}$. We have that

$$\begin{aligned}
&\texttt{cost}(\mathcal{C}^f) = \\
&\texttt{cost}(\mathcal{C}) + \texttt{dsum}(m_{A_f \cup B_f}, A_f \cup B_f) - \texttt{dsum}(m_{A_f}, A_f) - \texttt{dsum}(m_{B_f}, B_f) \leq && (31)\\
&\texttt{cost}(\mathcal{C}) + \left(\frac{\alpha-1}{\alpha} + \frac{1}{\alpha}\right) \min\{\texttt{dsum}(r_{A_f}, A_f \cup B_f), \texttt{dsum}(r_{B_f}, A_f \cup B_f)\} - \\
&\qquad \frac{1}{\alpha}(\texttt{dsum}(r_{A_f}, A_f) + \texttt{dsum}(r_{B_f}, B_f)) = \\
&\qquad \texttt{cost}(\mathcal{C}) + \frac{\alpha-1}{\alpha}\texttt{dsum}(r_{A_f}, A_f \cup B_f) + \\
&\frac{1}{\alpha} \min\left\{ \sum_{b \in B_f} \texttt{dist}(r_{A_f}, b) - \texttt{dist}(r_{B_f}, b), \sum_{a \in A_f} \texttt{dist}(r_{B_f}, a) - \texttt{dist}(r_{A_f}, a) \right\} \leq && (32)\\
&\texttt{cost}(\mathcal{C}) + \frac{\alpha-1}{\alpha}\texttt{dsum}(r_{A_f}, A_f \cup B_f) + \frac{1}{\alpha}(\min\{|A_f|, |B_f|\}\texttt{dist}(r_{A_f}, r_{B_f})) \leq && (33)\\
&\texttt{cost}(\mathcal{C}) + \frac{\alpha-1}{\alpha}(\texttt{dsum}(r_{A_f}, A_f) + |B_f|\texttt{dist}(r_{A_f}, r_{B_f}) + \texttt{dsum}(r_{B_f}, B_f)) + \\
&\qquad \frac{1}{\alpha}(\min\{|A_f|, |B_f|\}\texttt{dist}(r_{A_f}, r_{B_f})) = && (34)\\
&\texttt{cost}(\mathcal{C}) + \frac{\alpha-1}{\alpha}(\texttt{dsum}(r_{A_f}, A_f) + \texttt{dsum}(r_{B_f}, B_f)) + \texttt{dist}_f(A_f, B_f) \leq && (35)\\
&\texttt{cost}(\mathcal{C}) + \frac{\alpha-1}{\alpha}(\texttt{dsum}(r_{A_f}, A_f) + \texttt{dsum}(r_{B_f}, B_f)) + \texttt{dist}_f(A, B) = \\
&\texttt{cost}(\mathcal{C}) + \frac{\alpha-1}{\alpha}(\texttt{dsum}(r_{A_f}, A_f) + \texttt{dsum}(r_{B_f}, B_f)) + \min\{|A|, |B|\}\texttt{dist}(r_A, r_B) \leq && (36)\\
&\texttt{cost}(\mathcal{C}) + \frac{\alpha-1}{\alpha}(\texttt{dsum}(r_{A_f}, A_f) + \texttt{dsum}(r_{B_f}, B_f)) + \\
&\min\{|A|, |B|\}(\texttt{dist}(r_A, m_A) + \texttt{dist}(m_A, m_B) + \texttt{dist}(m_B, r_B)) \leq && (37)\\
&\texttt{cost}(\mathcal{C}) + \frac{\alpha-1}{\alpha}(\texttt{dsum}(r_{A_f}, A_f) + \texttt{dsum}(r_{B_f}, B_f)) + \\
&3\texttt{dsum}(r_A, A) + 3\texttt{dsum}(r_B, B) + \min\{|A|, |B|\}\texttt{dist}(m_A, m_B) \leq && (38)\\
&\texttt{cost}(\mathcal{C}) + (\alpha-1)(\texttt{dsum}(m_{A_f}, A_f) + \texttt{dsum}(m_{B_f}, B_f)) + 6\alpha\texttt{dsum}(m_{A \cup B}, A \cup B) + \\
&\min\{|A|, |B|\}(\texttt{dist}(m_A, m_{A \cup B}) + \texttt{dist}(m_{A \cup B}, m_B)) \leq && (39)\\
&\sum_{C \in \mathcal{C} \setminus \{A,B\}} \texttt{dsum}(m_C, C) + (\alpha-1)(\texttt{dsum}(m_{A_f}, A_f) + \texttt{dsum}(m_{B_f}, B_f)) + \\
&\qquad (6\alpha+5)\texttt{dsum}(m_{A \cup B}, A \cup B) && (40)
\end{aligned}$$

where inequality (31) holds due to the assumption that the representatives are α-approximations; inequality (32) follows because the triangle inequality ensures that $|\texttt{dist}(r_{A_f}, x) - \texttt{dist}(r_{B_f}, x)| \leq \texttt{dist}(r_{A_f}, r_{B_f})$, for every x in the metric space; (33) holds because the triangle inequality ensures that $\texttt{dist}(r_{A_f}, b) \leq \texttt{dist}(r_{A_f}, r_{B_f}) + \texttt{dist}(r_{B_f}, b)$ for every $b \in B_f$; (34) follows from the definition of $\texttt{dist}_f$ and the assumption $|B_f| \leq |A_f|$ (35) due to the choice of `f-msi`; inequality (36) holds due to the triangle inequality; (37) holds due to Proposition B.1; (38) holds due to the assumption that the representatives are α-approximations to the medoid, from Proposition B.2, and from the triangle inequality. Finally, (39) follows from the Propositions B.1.

We also have

$$\texttt{cost}(\mathcal{C}^m) = \sum_{C \in \mathcal{C} \setminus \{A,B\}} \texttt{dsum}(m_C, C) + \texttt{dsum}(m_{A \cup B}, A \cup B) \tag{41}$$

We consider the following cases:

Case 1) $A_f \notin \{A, B\}$ and $B_f \notin \{A, B\}$. In this case, dividing (40) by (41), we obtain

$$\frac{\texttt{cost}(\mathcal{C}^f)}{\texttt{cost}(\mathcal{C}^m)} \leq 6\alpha + 5$$

Case 2) $A_f \in \{A, B\}$ and $B_f \in \{A, B\}$. In this case, we have $\texttt{dsum}(m_{A_f}, A_f) + \texttt{dsum}(m_{B_f}, B_f) \leq 2\texttt{dsum}(m_{A\cup B}, A \cup B)$, so that we get the upper bound

$$\texttt{cost}(\mathcal{C}^f) \leq \sum_{C\in\mathcal{C}\setminus\{A,B\}} \texttt{dsum}(m_C, C) + (6\alpha + 5 + 2(\alpha - 1))\,\texttt{dsum}(m_{A\cup B}, A \cup B).$$

Thus,

$$\frac{\texttt{cost}(\mathcal{C}^f)}{\texttt{cost}(\mathcal{C}^m)} < 8\alpha + 3$$

Case 3) $A_f \in \{A, B\}$ and $B_f \notin \{A, B\}$. Assume w.l.o.g. that $A_f = A$. In this case, $\texttt{dsum}(m_{A_f}, A_f) \leq 2\texttt{dsum}(m_{A\cup B}, A) \leq 2\texttt{dsum}(m_{A\cup B}, A \cup B)$. Then, similarly to the previous case

$$\texttt{cost}(\mathcal{C}^f) \leq \sum_{C\in\mathcal{C}\setminus\{A,B\}} \texttt{dsum}(m_C, C) + (6\alpha + 5 + 2(\alpha - 1))\,\texttt{dsum}(m_{A\cup B}, A \cup B),$$

and

$$\frac{\texttt{cost}(\mathcal{C}^f)}{\texttt{cost}(\mathcal{C}^m)} \leq 8\alpha + 3$$

Case 4) $A_f \notin \{A, B\}$ and $B_f \in \{A, B\}$. This case is analogous to the previous one.

□

Natural linkage functions can increase the cost by $\Omega(n)$ factor. Consider $k + 1$ clusters $C_1, C_2, \ldots, C_{k+1}$, with $k \geq 3$. C_1 consists of $(n - (k - 1))/2$ points located very close to position 1 on the real line, C_2 consists of $(n - (k - 1))/2$ points located very close to position 2 on the real line; C_i, for $i = 3, \ldots, k + 1$ consists of a single point located at position $2i$.

By employing the linkage distance of `average-link`, `complete-linkage` or `medoid` (Herr et al., 2016; Schubert, 2021), C_1 and C_2 will be merged yielding a k-clustering of cost $(n-(k-1))/2$. On the other hand, merging clusters C_3 and C_4 yields a k-clustering of cost 2.

E INVERSIONS OF `msi`, `r-msi` AND `f-msi`.

E.1 PROOF OF THEOREM 4.4

Proof. Let A_i and B_i be the clusters merged at iteration i of `f-msi`. It is enough to prove that $\texttt{dist}_f(A_i, B_i) \leq \texttt{dist}_f(A_{i+1}, B_{i+1})$. 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 $\texttt{dist}_f(A_i, B_i) \leq \texttt{dist}_f(A_{i+1}, B_{i+1})$.

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, $\texttt{dist}_f(A_{i+1}, B_{i+1}) = \texttt{dist}_f(A_i \cup B_i, B_{i+1})$. The representative of A_{i+1} is either that of A_i or that of B_i.

We assume w.l.o.g. that the representative of A_{i+1} is the same as that of A_i. Thus,

$$\begin{aligned}\texttt{dist}_f(A_{i+1}, B_{i+1}) = \texttt{dist}(r_{A_i}, r_{B_{i+1}}) \cdot \min\{|A_{i+1}|, |B_{i+1}|\} \geq \\ \texttt{dist}(r_{A_i}, r_{B_{i+1}}) \cdot \min\{|A_i|, |B_{i+1}|\} = \texttt{dist}_f(A_i, B_{i+1}) \geq \\ \texttt{dist}_f(A_i, B_i),\end{aligned}$$

where the last inequality follows from the choice of `f-msi` at the ith iteration.

□

E.2 msi AND r−msi ARE NOT INVERSION-FREE

Proposition E.1. msi *and* r−msi *may produce inversions.*

Proof. Consider the five points $p_0 = (0, 1)$, $p_1 = (1, 0)$, $p_2 = (1, 5)$, $p_3 = (2, 2)$ and $p_4 = (5, 1)$ of $\mathbb{R}^2$ under the Euclidean distance. We first establish the result for r−msi, and then we argue that this same instance works for msi.

Since all clusters are singletons at the first iteration, the linkage distances are the pairwise distances, of which the smallest is $\texttt{dist}(p_0, p_1) = \sqrt{2}$, so $\{p_0\}$ and $\{p_1\}$ are merged at height $\sqrt{2}$. Both points minimize $\texttt{dsum}(\cdot\,, \{p_0, p_1\})$; assume the medoid is p_0, the other case being symmetric. At the second iteration $\texttt{dist}_r(\{p_0, p_1\}, \{p_3\}) = \sqrt{5}$, which is smaller than all the remaining linkage distances whose distinct values are $\sqrt{17}$, 5, $\sqrt{10}$, and $4\sqrt{2}$, so $\{p_3\}$ is merged at height $\sqrt{5}$. The medoid of $\{p_0, p_1, p_3\}$ is again p_0 or p_1, both at $\texttt{dsum} = \sqrt{2} + \sqrt{5}$, against $2\sqrt{5}$ for p_3; assume it is p_0.

Write $A := \{p_2\}$, $B := \{p_0, p_1, p_3\}$ and $C := \{p_4\}$ for the three clusters active at the third iteration. Their linkage distances are

$$\texttt{dist}_r(A, B) = \sqrt{17}, \qquad \texttt{dist}_r(B, C) = 5, \qquad \texttt{dist}_r(A, C) = 4\sqrt{2},$$

so A and B are merged, at height $\sqrt{17}$. The medoid of $A \cup B = \{p_0, p_1, p_2, p_3\}$ is p_3, uniquely, and the final merge occurs at

$$\texttt{dist}_r(A \cup B, C) = \texttt{dist}(p_3, p_4) = \sqrt{10}.$$

Since $\sqrt{17} > \sqrt{10}$, the heights are not monotone and r−msiproduces an inversion.

The same instance also shows that msi can produce inversions. The first two merges remain unchanged. At the third iteration, using the clusters A, B, C defined above, we have

$$\texttt{dist}_{\texttt{msi}}(A, B) = \texttt{dist}_{\texttt{msi}}(B, C) = \sqrt{5} + \sqrt{10} - \sqrt{2} < 4\sqrt{2} = \texttt{dist}_{\texttt{msi}}(A, C),$$

where we used $\texttt{cost}(B) = \sqrt{2} + \sqrt{5}$ and $\texttt{cost}(A \cup B) = 2\sqrt{5} + \sqrt{10}$. Thus, by symmetry, assume that A and B are merged. The medoid of $A \cup B$ is p_3, and the final merge with C occurs at height

$$\texttt{dist}_{\texttt{msi}}(A \cup B, C) = \sqrt{10}.$$

Since $\sqrt{5} + \sqrt{10} - \sqrt{2} > \sqrt{10}$, msi also produces an inversion on this instance.

□

F ANALYSIS OF f−msi FOR GRAPHS

We repeat the pseudo-code here to facilitate the discussion.

Correctness. We argue that the graph version takes the same decision employed by some execution of f−msi.

The key idea is to observe that the graph version maintains the following invariants at the beginning of each loop:

- If A and B are active clusters at the execution of f−msi and $\texttt{dist}(r_A, r_B) < K$, then there is an edge e in the heap associated with (r_A, r_B), with weight at most $\texttt{dist}_f(A, B)$.
- if A is an active cluster at the execution of f−msi that satisfies $\texttt{dist}(r_A, r_C) = K$ for every representative r_C of an active cluster $C \neq A$, then there is a special edge associated with r_A with $\texttt{key}(e) \leq K \cdot |A|$.

Assume that these conditions hold at the beginning of a loop. We have the following cases:

a) e is regular and either u or v is not active. In this case, e is discarded, and the conditions still hold.

Algorithm 4 Graph version of `f-msi` method

```
1: while there are at least two active clusters do
2:     Remove the edge e of minimum weight from H
3:     if e is regular then
4:         Let u and v be the endpoints of e, and assume w.l.o.g. |C_u| ≥ |C_v|
5:         if u and v are active representatives then
6:             if key(e) = dist_f(C_u, C_v) then
7:                 Merge C_u and C_v; Set u as the representative of C_u ∪ C_v
8:                 Z_v = {z is active, z is adjacent to v and z is not adjacent to any active representative}
9:                 For each z ∈ Z_v, add to H a special edge e_z with key(e_z) = K · |C_z|
10:            else
11:                Reinsert e in the heap with key(e) = dist_f(C_u, C_v)
12:    else
13:        if the representative v associated with the special edge e is active then
14:            if key(e) = K · |C_v| then
15:                Pick some active representative u, with u ≠ v
16:                Merge C_u and C_v; Set u as the representative of C_u ∪ C_v
17:            else
18:                Reinsert e in H with key(e) = K · |C_v|
```

b) e is regular and $\texttt{key}(e) < \texttt{dist}_f(C_u, C_v)$. Then, the weight of e is updated, and the property still holds.

c) e is regular and $\texttt{key}(e) = \texttt{dist}_f(C_u, C_v)$. In this case, C_u and C_v are merged and u is set as the representative of $C_u \cup C_v$. Let C be an active cluster different from both C_u and C_v. If $\texttt{dist}(r_C, u) < K$, then the edge associated with (r_C, u) has weight at most $\texttt{dist}_f(C, C_u)$ before the merge (due to the invariant), which is not larger than $\texttt{dist}_f(C, C_u \cup C_v)$. Thus, the condition of the first bullet holds. If $\texttt{dist}(r_C, u) = K$, the condition of the second bullet still holds after the merge.

d) e is special and v is not active. In this case, e is discarded, and the conditions still hold.

e) e is special and $\texttt{key}(e) < K \cdot |C_v|$. Then, the weight of e is updated, and the property still holds.

f) e is special and $\texttt{key}(e) = K \cdot |C_v|$. In this case, v is merged with some arbitrarily chosen node u, and u becomes the representative of the new cluster. This is a valid merge because $|C_v| \leq |C_u|$. Assume that $|C_u| < |C_v|$. Then, there would be an edge touching u with weight smaller than that of e, which is not possible.

These invariants imply that the decision of the graph version corresponds to a merge executed by `f-msi`.

Complexity Analysis. Before analyzing the procedure, we explain how the set Z_v is obtained in line 8. We maintain a vector `Degree-Rep` indicating for each vertex x the number of neighbors of x that are still active representatives. Initially, `Degree-Rep[x]` is set to the degree of x in the graph G. When C_u and C_v are merged in line 7, we decrease `Degree-Rep[x]` by one unit for every active vertex x adjacent to v. To build Z_v, we set Z_v as an empty set at the beginning of each while loop. Then, if `Degree-Rep[x]` becomes 0, x is added to Z_v. Note that maintaining `Degree-Rep` and the set Z_v along the execution requires $O(m + n)$ time.

The total running time is obtained by summing the costs incurred over all iterations of the while loop. The time required to construct Z_v has already been discussed, so we focus on the remaining costs. Every other line contributes $O(\log(m + n))$ time, except for line (9), which contributes $O(d(v) \log(m + n))$ time, where $d(v)$ denotes the degree of v in G. Note that the $O(\log(m + n))$ is due to the heap operations. However, when line (9) occurs, v is not a representative anymore. Thus, the overall contribution of line (9) is $O((m + n) \log(m + n))$. Therefore, the time complexity is given by $O(T \log(m + n) + (m + n) \log(m + n))$, where T is the number of times the while loop is executed. We can count T by adding the number of times each of the following 5 mutually exclusive scenarios occur.

1. u or v is not an active representative (line 5 or line 13). This scenario can happen at most $m + n$ times because when one regular or special edge is removed from the heap, it is never inserted again
2. u are v active representatives and $\texttt{key}(e) = \texttt{dist}_f(u, v)$ (line 6). This happens at most $n - 1$ times because at most $n - 1$ clusters are merged.
3. u are v active representatives and $\texttt{key}(e) < \texttt{dist}_f(C_u, C_v)$ (line 10). This more involved case is addressed below.
4. e is a special edge and $\texttt{key}(e) = K \cdot |C_v|$ (line 14). This happens at most $n - 1$ times because at most $n - 1$ clusters are merged.
5. e is a special edge and $\texttt{key}(e) < K \cdot |C_v|$ (line 17). Again, this more involved case is addressed below.

We need to analyse the number of times scenarios 3 and 5 are reached. We start with scenario 5. We can associate each occurrence of scenario 5 that involves an edge whose regular vertex is x with the last merge involving x before this occurrence. Since each occurrence of scenario 5 is associated with a different merge, then this scenario occurs at most $n - 1$ times and, hence, it contributes with $O(n \log(m + n))$ time.

Now, we focus on scenario 3. A key observation is that the number of times scenario 3 occurs is upper bounded by the number of times the key of an edge $e' = xy$ becomes different from $\texttt{dist}_f(C_x, C_y)$. This can only happen when a merge involving x or y is performed.

For a node x, let $m(x)$ denote the number of times x becomes the representative of the resulting cluster in a merge involving x. For $i = 1, \ldots, m(x)$, let $L_i(x)$ be the number of neighbors of x in G that satisfy the following conditions right before the ith such merge: they are active representatives and their sizes are larger than $|C_x|$. Then, the number of updated edges that become outdated due to the the ith merge is at most $L_i(x)$, and overall number of updated edges that become outdated is upper bounded by

$$\sum_{x \in G} \sum_{i=1}^{m(x)} L_i(x)$$

Clearly, $L_i(x) \leq d(x)$, where $d(x)$ is the degree of x in G. Moreover, $L_i(x) \leq \frac{n}{i}$ because at the beginning of the ith iteration that x is merged, $\texttt{size}(x) \geq i$ and the number of clusters with size at least i is at most n/i. Let S be the set of nodes in G with degree at most $\sqrt{m}$. Thus,

$$\sum_{x \in G} \sum_{i=1}^{m(x)} L_i(x) \leq \sum_{x \in G} \sum_{i=1}^{m(x)} \min\left\{d(x), \frac{n}{i}\right\} = \sum_{x \in S} \sum_{i=1}^{m(x)} \min\left\{d(x), \frac{n}{i}\right\} + \sum_{x \notin S} \sum_{i=1}^{m(x)} \min\left\{d(x), \frac{n}{i}\right\} \leq$$

$$\sum_{x \in S} m(x)\sqrt{m} + \sum_{x \notin S} \left(\sum_{i=1}^{\lceil n/\sqrt{m} \rceil} d(x) + \sum_{i=\lceil n/\sqrt{m} \rceil + 1}^{m(x)} \sqrt{m} \right) \leq$$

$$\sum_{x \in S} m(x)\sqrt{m} + \sum_{x \notin S} \frac{n}{\sqrt{m}} d(x) + \sum_{x \notin S} m(x)\sqrt{m}$$

By using the facts $\sum_{x \in G} m(x) = n$ and $\sum_{x \in G} d(x) = 2m$, we conclude that the above expression is $O(n\sqrt{m})$,

G COMPUTING MEDOIDS EFFICIENTLY

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.

G.1 Computing medoids in $O(n^2)$ time

Algorithm 6 outputs the medoids of the clusters $X_1, \ldots, X_{2n-1}$. The approach consists of computing $\texttt{dsum}(p, X_i)$ for each cluster X_i and each x in X_i. These values are stored in the variable `SumDist`.

Time Complexity. The overall complexity of the algorithm is

$$O\left(\sum_{i=1}^{n-1} |X_{\ell(i)}| \cdot |X_{r(i)}|\right). \tag{42}$$

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. 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 to contribute to another product. Thus, each pair of points contributes at most once to (42), so this sum is $O(n^2)$.

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

Algorithm 6 Computing medoids in $O(n^2)$ time.

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

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

G.2 Computing the medoids for a graph dataset in $O(m \log n + n \log n)$ time

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. The stamp of an edge is exactly the LCA (Lowest Common Ancestor) of u and v in $\mathcal{D}$, and all stamps can be computed in $O(m \log n + n \log n)$ time by preprocessing $\mathcal{D}$ to answer LCA queries (Bender and Farach-Colton, 2000). Our presentation is in terms of the weights of the edges: the weight of an edge uv is the distance between the points associated with u and v, while the distance between non-adjacent nodes is K. We assume $w(uv) \leq K$ for every edge, which holds in our setting, where dissimilarities are normalized to $[0, 1]$ and $K = 1$.

Splitting the median cost according to whether or not y is a neighbor of x makes it additive over the edges:

$$\begin{aligned} \sum_{y \in X_i} \texttt{dist}(x, y) \;&=\; \sum_{\substack{xy \in E \\ y \in X_i}} w(xy) \;+\; K\left(|X_i| - 1 - \deg_{X_i}(x)\right) \\ &=\; T_{X_i}(x) + K\left(|X_i| - 1\right), \quad \text{where } T_{X_i}(x) := \sum_{\substack{xy \in E \\ y \in X_i}} \big(w(xy) - K\big). \end{aligned} \tag{43}$$

As $K\left(|X_i| - 1\right)$ does not depend on x, the medoid of X_i minimizes the key T_{X_i}; in particular, the non-neighbors of x need not be tracked, since their contribution splits into the per-node term $-K \deg_{X_i}(x)$, absorbed into T_{X_i}, and a term common to all candidates.

Algorithm 7 presents a pseudo-code. It takes $\mathcal{D}$ as input and executes $n-1$ loops, each with two blocks. The first block adds $w(e)-K$ to the global array T at both endpoints of every edge e with stamp i; its overall contribution is $O(m)$, as each edge is considered once. Since the clusters containing a fixed node form a chain in $\mathcal{D}$, every edge internal to X_i has stamp i or a descendant of i, so after this block $T[v] = T_{X_i}(v)$ for all $v \in X_i$. The second block selects the node of smallest key in the candidate set

$$Z_i \;=\; \{\, medoid(X_{\ell(i)}),\; medoid(X_{r(i)}) \,\} \;\cup\; \{\, u \mid u \text{ is an endpoint of an edge with stamp } i \,\},$$

and its cost is read off Equation (43). As $|Z_i| \le 2 + 2|\{e : e \text{ has stamp } i\}|$, the contribution of this block is $O(m+n)$. Correctness follows from Lemma G.1, so the total running time is $O(m \log n + n \log n)$, dominated by the LCA preprocessing.

Lemma G.1. *Some node of Z_i minimizes T_{X_i} over X_i.*

Proof. Let $x \in X_{\ell(i)} \setminus Z_i$; the argument for $X_{r(i)}$ is analogous, and let $z = medoid(X_{\ell(i)})$. Every edge between $X_{\ell(i)}$ and $X_{r(i)}$ has stamp i, so x has no neighbor in $X_{r(i)}$ and $T_{X_i}(x) = T_{X_{\ell(i)}}(x)$. Also, $T_{X_i}(z) = T_{X_{\ell(i)}}(z) + \sum_e (w(e) - K) \le T_{X_{\ell(i)}}(z)$, where the sum ranges over the edges with stamp i incident to z and the inequality uses $w \le K$. Since $T_{X_{\ell(i)}}(z) \le T_{X_{\ell(i)}}(x)$ by the optimality of z, chaining gives $T_{X_i}(z) \le T_{X_i}(x)$. □

The assumption $w \le K$ is essential above: otherwise keys may increase and the medoid of a child may be overtaken by a node outside Z_i. For arbitrary weights one keeps, for each cluster, a binary search tree of the pairs $(T[v], v)$ keyed by T, inherits the tree of the larger child and inserts the elements of the smaller one, so that each node is re-inserted $O(\log n)$ times; the medoid is then the minimum of the tree, and the running time becomes $O(m \log n + n \log n)$.

Algorithm 7 COMPUTING MEDOIDS BY CANDIDATE SCANNING

Require: $\mathcal{D}$: dendrogram, $G = (V, E, W)$: weighted graph with $w(e) \le K$ for every $e \in E$
Ensure: The medoids of clusters $X_1, \ldots, X_{2n-1}$.

```
Array T[1 ... n] ← 0
For u = 1, ..., n: medoid(X_u) ← u; |X_u| ← 1
For i = n + 1, ..., 2n − 1
    |X_i| ← |X_ℓ(i)| + |X_r(i)|
    For every edge e = (u, v) with stamp i and weight w
        T[u] ← T[u] + (w − K);  T[v] ← T[v] + (w − K)
    End For
    Z_i ← {medoid(X_ℓ(i)), medoid(X_r(i))} ∪ {u | u is an endpoint of an edge with stamp i}
    medoid(X_i) ← arg min_{z∈Z_i} T[z]
    cost(X_i) ← T[medoid(X_i)] + K (|X_i| − 1)
End For
```

H SUBQUADRATIC APPROXIMATION

Algorithm 8 shows the subquadratic version of `f-msi`. It is based on the subquadratic version of `ward` (Abboud et al., 2019), but it incorporates some modifications that we explain below.

The external loop is executed while there are at least two active clusters. Each external loop is associated with a value ν, and it uses a heuristic to merge clusters whose `f-msi` distance is at most ν (lines (6)-(16)). When the heuristic does not find clusters to merge, the value of ν is updated, and a new iteration of the external loop starts. A data structure `ToMerge` is employed to keep clusters that the algorithm tries to merge with some active cluster in the current external loop.

Our heuristic differs from that of (Abboud et al., 2019) because it tries to merge the cluster with minimum size in `ToMerge` at each internal loop (line 7). This allows using only one data structure to support ANN queries, in contrast to the $\log n$ data structures used in (Abboud et al., 2019).

For the ANN queries, the algorithm uses a data structure $\mathcal{S}$ that maintains the representatives of the active clusters. It supports insertions, deletions, and ANN queries that work as follows: given

Algorithm 8 γ-Approximation of `f-msi` method

Require: Set of points $\mathcal{X} \in \mathbb{R}^d$ and a data structure $\mathcal{S}$ that supports insertions, deletions and $\sqrt{\gamma}$ ANN queries.

1: `Active` $\leftarrow$ points in $\mathcal{X}$.
2: S $\leftarrow$ points in $\mathcal{X}$.
3: $\nu \leftarrow$ a lower bound on the minimum distance among the points in $\mathcal{X}$
4: **while** `Active` has more than one cluster **do**
5: TOMERGE $\leftarrow$ `Active`
6: **while** TOMERGE is not empty **do**
7: Remove the cluster C of minimum size from TOMERGE
8: $x \leftarrow$ ANN$(S - \{r_c\}, r_C)$
9: **if** $\texttt{dist}_f(C_x, C) \leq \nu$ **then**
10: Merge C and C_x.
11: **if** $|C| \geq |C_x|$ **then**
12: Remove x from $\mathcal{S}$
13: **else**
14: Remove r_C from $\mathcal{S}$
15: Remove C_x from TOMERGE (if it belongs to it) and add $C \cup C_x$ to TOMERGE.
16: Remove C, C_x from `Active` and add $C \cup C_x$ to `Active`.
17: $\nu \leftarrow \sqrt{\gamma}\nu$

a point q, it returns a point $ANN(q)$ that satisfies $\texttt{dist}(q, ANN(q)) \leq \sqrt{\gamma}\texttt{dist}(q, x)$, where x is the closest point to q in the data structure.

To obtain the initial ν (line 3), we first merge duplicate points. Let $\mathcal{X}'$ be resulting set of distinct points. The algorithm performs $|\mathcal{X}'|$ ANN queries, one for each point $x \in \mathcal{X}'$ in a data structure that stores the points in $\mathcal{X}' \setminus \{x\}$. Let $ANN(x_i)$ the result associated with the query to x_i. We define $\nu = \min\{\texttt{dist}(x_i, ANN(x_i)) | i \in \mathcal{X}'\}/\sqrt{\gamma}$.

Running time. For the analysis, we assume that the ANN data structure supports insertions, deletions, and ANN queries in $n^{f(\sqrt{\gamma})}$ time.

The computation of the initial ν takes $O(n^{1+f(\sqrt{\gamma})})$.

The outer loop of Algorithm 8 is executed $\log_{\sqrt{\gamma}}(n\Delta)$ times because $\sqrt{\gamma}n\Delta\nu$ is an upper bound on the linkage distance between any pair of clusters. The data structure `ToMerge` can be implemented as a vector of size $2n-1$, where the ith entry points to the list of clusters of size i. By keeping a pointer to the list of of clusters of minimum size, the overall time incurred to remove the cluster of minimum size from `ToMerge` (line 7), along one external loop, is $O(n)$. The inner while loop executes at most $2n$ iterations because at every internal loop, the number of elements in `ToMerge` $\cup$ `Active` decreases by at least one unit. Moreover, each iteration requires at most the time of an ANN query, namely $O(n^{f(\sqrt{\gamma})}.)$

Combining these bounds gives a total running time of

$$O\left(n^{1+f(\sqrt{\gamma})} \log_{\sqrt{\gamma}}(n\Delta)\right)$$

Proof of Correctness. We need to show that each merge between clusters C and D performed by the procedure satisfies $\texttt{dist}_f(C, D) \leq \gamma\texttt{dist}_f(A, B)$, where A and B are the closest clusters in terms of `f-msi` distance at the moment in which the merge occurs

The correctness is a consequence of the following lemma because at each external loop only clusters with `f-msi` distance not larger than ν are merged.

Lemma H.1. *Let ν_ℓ be the value of ν at the beginning of the ℓth external loop. Then, for every ℓ, at the beginning of the ℓth external loop, every pair of clusters A and B in* `Active` *satisfies*

$$\texttt{dist}_f(A, B) \geq \frac{\nu_\ell}{\gamma}.$$

Proof. At the beginning of the first loop, ν is by construction a lower bound on the distance between any pair of points. Thus, the linkage distance between any pair of clusters is larger than ν.

Fix $\ell \geq 1$. Now, we argue that the invariant holds at the beginning of the $(\ell+1)$th loop. Since the clusters in `Active` at the beginning of the $(\ell+1)$th loop are those at the end of the ℓth loop, it is enough to show that at the end of the ℓth loop, there cannot be two clusters C_i and C_j in `Active` satisfying $\texttt{dist}_f(C_i, C_j) < \frac{\nu_\ell}{\sqrt{\gamma}}$.

Suppose, for the sake of contradiction, that such clusters exist, and assume without loss of generality that $|C_i| \leq |C_j|$.

Since C_i is active at the end of the ℓth loop, it is eventually extracted from ToMerge during this loop and is not merged after that extraction. Notice that this remains true if C_i is created during the loop: every newly created cluster is inserted into ToMerge, and the loop continues until ToMerge is empty.

Let r_i and r_j denote the representatives of C_i and C_j, respectively. Consider the moment when C_i is extracted from ToMerge. Since representatives are inherited upon merges, there is an active cluster $C'_j \subseteq C_j$ whose representative is r_j: namely, the active cluster containing the point r_j on the merge path that eventually produces C_j. Thus, r_j belongs to the ANN search set $\mathcal{S}$ at this moment. Notice that C'_j need not belong to ToMerge.

Let q be the point returned by $\mathrm{ANN}(r_i, \mathcal{S} \setminus \{r_i\})$, and let C_q be the active cluster represented by q. By the ANN guarantee $\texttt{dist}(r_i, q) \leq \sqrt{\gamma}\,\texttt{dist}(r_i, r_j)$.

Therefore,

$$\begin{aligned}
\texttt{dist}_f(C_i, C_q) &= \min\{|C_i|, |C_q|\}\texttt{dist}(r_i, q) && (44)\\
&\leq |C_i|\texttt{dist}(r_i, q) && (45)\\
&\leq \sqrt{\gamma}\,|C_i|\texttt{dist}(r_i, r_j) && (46)\\
&= \sqrt{\gamma}\,\texttt{dist}_f(C_i, C_j) && (47)\\
&< \nu_\ell, && (48)
\end{aligned}$$

where the equality in the fourth line follows from $|C_i| \leq |C_j|$.

Hence, when C_i is extracted from ToMerge, the algorithm finds an active cluster whose `f-msi` linkage with C_i is at most ν_ℓ, and therefore C_i is merged. This contradicts the assumption that C_i belongs to `Active` at the end of the ℓth external loop.

Since $\nu_{\ell+1} = \sqrt{\gamma}\nu_\ell$ and the set `Active` does not change from the end of the ℓth loop to the beginning of the $(\ell+1)$th loop, we can conclude that there are no clusters C_i and C_j in `Active` satisfying $\texttt{dist}_f(C_i, C_j) < \frac{\nu_{\ell+1}}{\gamma}$ at the beginning of the $(\ell+1)$th loop. □

Avoiding unnecessary ANN queries. We use a simple optimization to reduce the number of calls to the ANN data structure. For each representative x, we store the distance δ_x to the point returned by its most recent ANN query. Suppose that x currently represents a cluster C. Since representatives are inherited after merges, the location of x does not change, while representatives may only be removed from the set of active representatives. Hence, the distance from x to its nearest candidate cannot decrease.

More precisely, if the ANN data structure provides a $\sqrt{\gamma}$-approximate nearest neighbor, then the true distance from x to any currently active representative is at least $\delta_x/\sqrt{\gamma}$. Let

$$s_{\min} = \min\{|C'| : C' \in \texttt{Active}\}.$$

Since

$$\texttt{dist}_f(C, C') = \min\{|C|, |C'|\}\,\texttt{dist}(r_C, r_{C'}),$$

for every active cluster $C' \neq C$, we have

$$\texttt{dist}_f(C, C') \geq s_{\min}\frac{\delta_x}{\sqrt{\gamma}}.$$

Therefore, whenever

$$s_{\min}\frac{\delta_x}{\sqrt{\gamma}} > \nu,$$

no active cluster can be merged with C in the current iteration, and the ANN query for x can be safely skipped.

This optimization is particularly useful because the same representative may survive several merges and be considered repeatedly as ν increases. By retaining the result of its previous ANN query, we can often certify that no admissible merge exists without accessing the ANN data structure.

I Experimental Study

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

All experiments, except those described in Appendix I.5 , 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.

I.1 Datasets

We used classical tabular datasets available in UCI Kelly et al. (2024), and also from specific domains, namely, text, movies/books rating, and biology. For the tabular datasets, we normalized the values in each column 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 Balaban et al. (2019); Kosakovsky Pond et al. (2018), we considered two datasets, one of them (Ecoli) is a collection of complete genomes of *Escherichia coli*, downloaded from the NCBI RefSeq database O'Leary et al. (2016). Each genome was processed using Jellyfish Marçais and Kingsford (2011), 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 Dimitrov et al. (2022). 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 learn developers (2017) and the Reuters-21578 Lewis (1987) collections. In both cases, we considered a random subset of 5000 documents that 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 and cosine 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 Harper and Konstan (2015) and a book-ratings Zając (2017) 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.

To evaluate the algorithms on a non-Euclidean metric space, we use a collection of English words obtained from the SCOWL (Spell Checker Oriented Word Lists) dataset Atkinson (2019). Specifically, we use the size-10 English word list, containing 4,374 words. Each word is treated as an object, and the distance between two objects is their Levenshtein (edit) distance, i.e., the minimum number of single-character insertions, deletions, and substitutions required to transform one word into the other. Unlike our vector datasets, no feature normalization is applied. This experiment therefore evaluates the algorithms directly on a discrete, non-Euclidean metric space.

Table 3 summarizes the datasets used, where n denotes the number of data points and d the number of attributes per point.

Table 3: Datasets: n is the number of points and d is the number of attributes.

Dataset	n	d	Source
Anuran	7193	22	Colonna and Gordo (2015)
Banknote	1371	5	Lohweg (2013)
Concrete	1028	8	Yeh (2007)
Geographical Music	1057	116	Zhou (2014)
Ozone	3694	72	Zhang and Yuan (2008)
Parkinsons	5874	19	Tsanas and Little (2009)
Qsarfish	906	6	Ballabio and Todeschini (2019)
Rice	3809	7	ric (2019)
Wine	4898	11	Cortez and Reis (2009)
Reuters	5000	300	Lewis (1987)
News20	5000	300	learn developers (2017)
Book Rating	5000	300	Zając (2017)
Movie Lens	6040	300	Harper and Konstan (2015)
Allen Brain	5000	3000	Dimitrov et al. (2022)
Ecoli	4514	511	O'Leary et al. (2016)
Beans	13611	16	dry (2020)
Mice	552	77	Higuera and Cios (2015)
Iris	150	4	Fisher (1936)
PenDigits	5000	16	Alpaydin and Alimoglu (1996)
BreastCancer	683	9	Wolberg (1990)
LetterRecognition	5000	16	Slate (1991)
Individual Household Electricity	2,075,259	9	Hebrail and Berard (2006)
Wave Energy Converter	288,000	49	Neshat and Alexander (2018)
SCOWL	4,374	N/A	Atkinson (2019)

I.2 Evaluated Methods

In order to assess the quality of our methods and its scalable implementations, we compared them with the following ones:

- `msi`. This linkage method proposed by Schubert (2021) uses
$$\texttt{dist}_{\texttt{msi}}(A, B) = \texttt{dsum}(m_{A\cup B}, A \cup B) - \texttt{dsum}(m_A, A) - \texttt{dsum}(m_B, B),$$
as the distance between clusters A and B, where m_C denotes the medoid of cluster C, It can be seen as a greedy approach to minimizing the k-medians cost function, and it runs in $O(n^3)$ time.
- `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.
- `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.
- `single-linkage`. It uses the minimum distance between points in A and B as the distance between two clusters A and B. In our experiments, `single-linkage` yields substantially larger k-medians costs than the other methods. We therefore omit its curves from the figures, as their scale would make the differences among the remaining methods difficult to visualize.
- `CLNSS`. This method proposed by Cohen-Addad et al. (2021) embeds the original points in a tree metric (HST) with provable $O(\log n)$ distortion, and then optimally solves the k-medians for the embedding points. Our implementation is discussed in Section I.5

- `TeraHAC`. This method proposed by Dhulipala et al. (2023) handles large sparse datasets. It implements a variant of `average-link`.

`complete-linkage`, `average-link` and `f-msi` were implemented using the NN-Chain framework Murtagh (1985); Müllner (2011). For `f-msi` this is justified by Theorem 4.3. The linkage distances of `msi` and `r-msi` do not satisfy condition (ii) of that theorem, so the NN-Chain framework does not apply to them. We therefore implemented both using the approach of Day and Edelsbrunner (1984) discussed in Appendix A, with lazy deletion to handle obsolete heap entries. This yields $O(n^2 \log n)$ time for `r-msi` and $O(n^3)$ time for `msi`, as each evaluation of $\texttt{dist}_{\texttt{msi}}$ requires computing the medoid of $A \cup B$, at a cost of $O(|A| + |B|)$.

I.3 k-MEDIAN COST FULL RESULTS

Figures 3 and 4 show the performance of the algorithms for the k-median cost, as a function of k, for our datasets. Recall that the point associated with k (horizontal axis) for a method M represents the ratio between the k-medians cost of the k-clustering built by M and that of the k-clustering built by `msi`. We choose `msi` as a reference because it produced the best results among hierarchical methods for this criterion.

I.4 `f-msi` FOR GRAPHS: MORE EXPERIMENTS

For our experiments, we employed the following datasets:

Amazon Reviews. We use the "All_Beauty" category from the Amazon Reviews 2023 dataset Hou et al. (2024). We construct a graph where vertices represent users, and the distance associated with an edge (u, v) is defined as 1 minus the Jaccard similarity between the sets of items reviewed by users u and v (i.e., the Jaccard distance), ignoring rating values. The graph has 592,092 nodes and 32,783,308 edges.

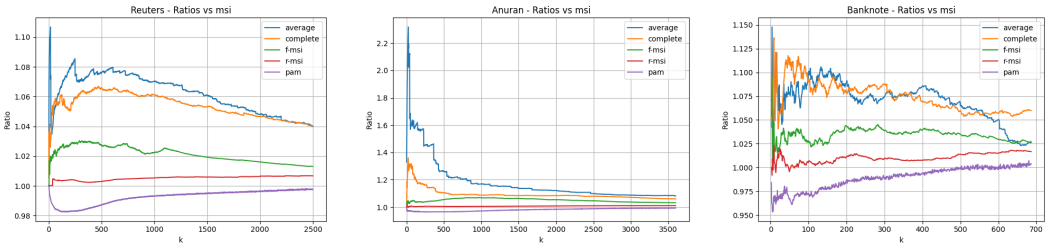
ca-CondMat. This dataset is the collaboration network of the arXiv Condensed Matter category (Leskovec et al., 2007), comprising 23,133 authors and their coauthorship relations. From this data, we construct a graph in which vertices correspond to authors and an edge connects two authors that have at 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.

Additional Comparison with `TeraHAC`. Figure 5 compares `f-msi` for graphs with `TeraHAC` for datasets `ca-CondMat`. For `TeraHAC`, we used the parameters $\epsilon = 0.1$ and $t = 0.01$ recommended in the paper. We also used the parameter $t = 0$ to recover the full dendrogram because when $t > 0$ some nodes are pruned. `TeraHAC` used all 10 threads (ParlayLib default). The results are consistent with those obtained for Amazon Reviews (presented in the body): `f-msi` achieves lower k-medians cost over most values of k.

I.5 THE SUBQUADRATIC VARIANT OF `f-msi`: EXPERIMENTS

We conducted experiments to assess the practical performance of the subquadratic implementation of `f-msi`, denoted here by `approx-fmsi`. In particular, we investigate two questions: (i) whether the clusterings produced by `approx-fmsi` implementation have k-medians costs comparable to those obtained by the exact implementation of `f-msi`, and (ii) whether `approx-f-msi` scales to datasets that are beyond the reach of quadratic-time linkage methods. In our experiments, we also evaluate how `f-msi` and `approx-f-msi` compare with the sequential method proposed by Cohen-Addad et al. (2021), denoted here by `CLNSS`, a nearly linear time method with logarithmic approximation guarantees for the k-medians objective.

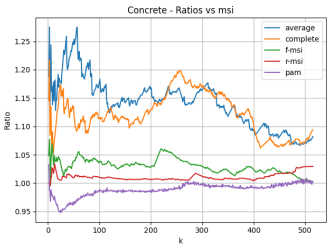
Since we were unable to find a public implementation, we implemented in C++ the RHST construction described in Section 3 of Cohen-Addad et al. (2021), using the random quadtree embedding from its Appendix F. Recall that the features of our datasets were normalized to $[0, 1]$. In our implementation, the points were mapped to the integer grid $\{0, \ldots, 2^{20}\}^d$, using a common scaling factor across all dimensions to preserve the Euclidean geometry up to discretization. We then sampled a random shift vector r uniformly from $[0, \Delta]^d$, where $\Delta = 2^{20}$, and added the same vector r to every point



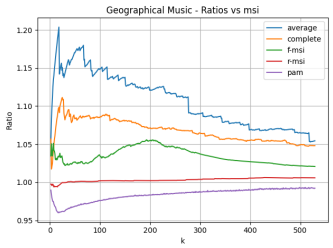
(a) Reuters

(b) Anuram

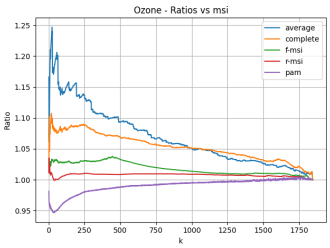
(c) Banknote



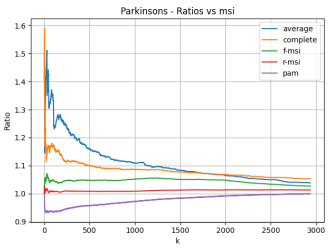
(d) Concrete



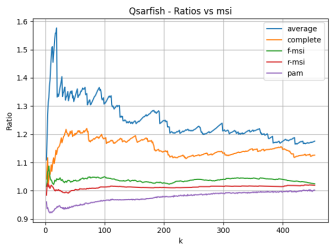
(e) Geo Music



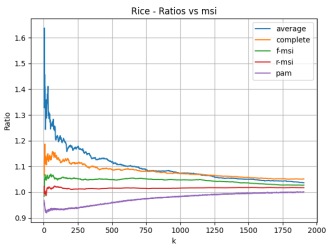
(f) Ozone



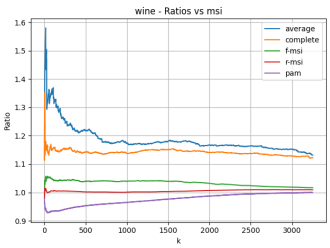
(g) Parkinsons



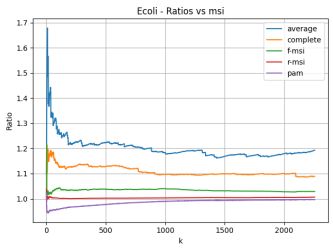
(h) QSAR Fish



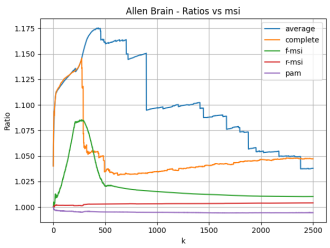
(i) Rice



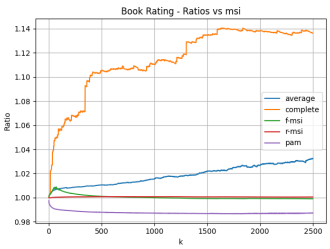
(j) Wine



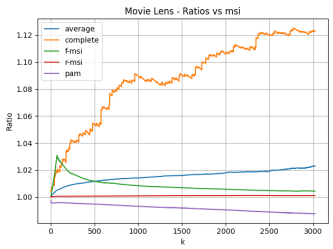
(k) Ecoli



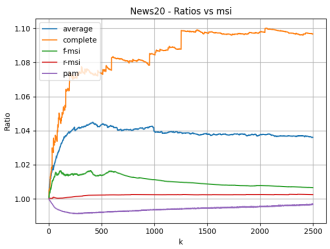
(l) Allenbrain



(m) Book Ratings

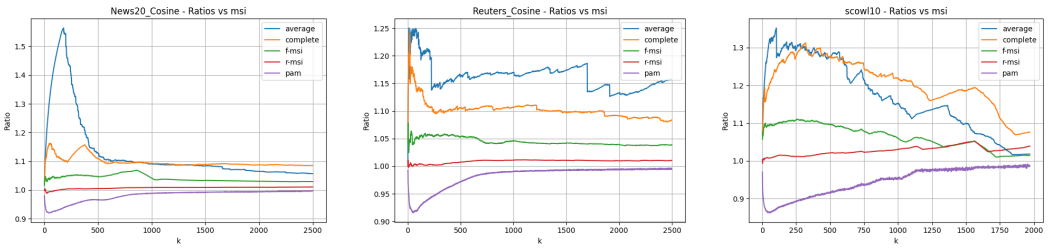


(n) MovieLens



(o) Newsgroups

Figure 3: Ratio between the k-median cost of each method and that of `msi` for each value of $k < n/2$



(a) Newsgroups cosine (b) Reuters cosine (c) Scowl

Figure 4: Ratio between the k-median cost of each method and that of `msi` for each value of $k < n/2$

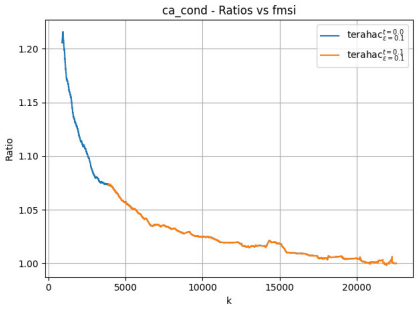


Figure 5: Comparison between `f-msi` and `TeraHAC` for `ca-CondMat`.

before constructing the quadtree hierarchy. The centers were selected according to the benefit-based procedure (Algorithm 2 of `CLNSS`).

For the ANN queries of `approx-fmsi`, we consider $\gamma \in \{1.1, 1.5, 2\}$ and the FLANN library (Muja and Lowe, 2009), with its default parameters, `checks=32` and `trees=4`. The theoretical analysis assumes an ANN oracle satisfying the approximation guarantee stated in Theorem 4.6, whereas FLANN is used here as a practical implementation and does not certify this guarantee. This library was also used in the experiments reported by Abboud et al. (2019).

All the experiments in this section were conducted on Google Colab using a virtual machine with a 2.20 GHz Intel Xeon CPU (2 vCPUs) and 12 GiB of RAM.

Figure 6 reports the ratio between the k-medians costs obtained by `CLNSS`, `approx-fmsi` (with different values of γ), and the exact implementation of `f-msi` for six datasets. Each point for `approx-fmsi` and `CLNSS` represents the mean over five runs with different random seeds, with shaded regions indicating one standard deviation. `approx-fmsi` produces clusterings whose k-medians costs remain close to those obtained by exact `f-msi` across the evaluated datasets, with low variability across random seeds. Smaller values of γ generally lead to solutions closer to those produced by the exact implementation. `CLNSS` yields substantially higher k-medians costs and greater variability than `approx-fmsi`, consistent with the experiments reported by Li et al. (2025), where `CLNSS` is outperformed by `complete-linkage` in most of the evaluated datasets.

Table 4 reports the average running times, over 5 seeds, for the instances in which `f-msi` took more than 1 second. The standard deviations are very low and we omit them. As expected, the larger the γ, the faster the `approx-fmsi`. These, together with the results of Figure 6 suggest a trade-off between k-medians cost and running time. We also observe that `CLNSS` is the fastest method, which is not surprising given its nearly linear time complexity.

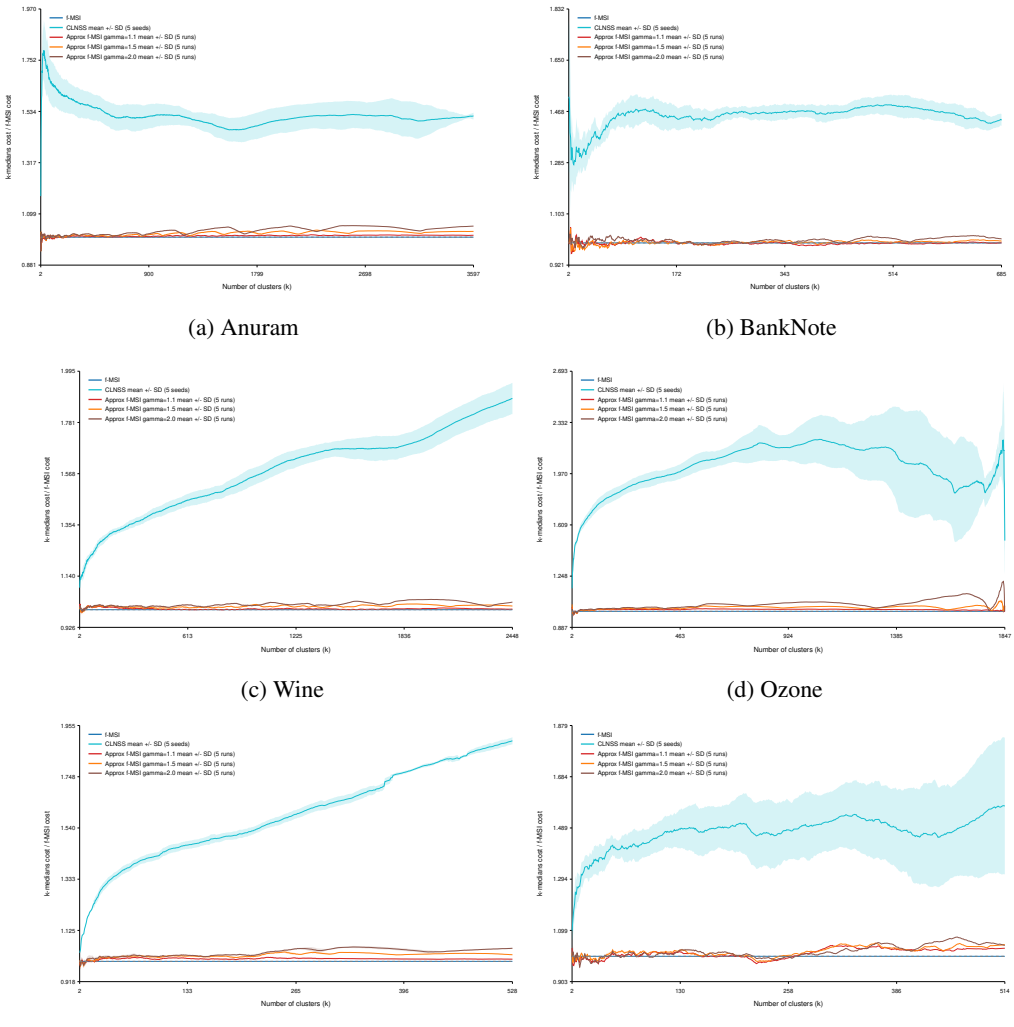
We also evaluate the scalability of `approx-fmsi` on substantially larger datasets that are beyond the practical reach of quadratic-time linkage methods. We consider the `Wave Energy Converter` dataset, containing 288,000 points with 49 features, and `Individual Household Electricity`, containing 2,075,259 points with 7 numeric features. On the former, `approx-fmsi` with $\gamma = 2$ required approximately 2 minutes to construct the entire dendrogram,

Table 4: Running times in seconds, average of 5 seeds

dataset	**n**	`f-msi`	`approx-fmsi(1.1)`	`approx-fmsi(1.5)`	`approx-fmsi(2)`	`CLNSS`
Anuram	7,195	42,1	3.2	0.96	0.89	0.14
Ozone	3.695	8.0	1.0	0.30	0.25	0.11
Wine	4,898	16.3	1.6	0.52	0.5	0.09

while on the latter it took approximately 7 minutes. In comparison, `CLNSS` required approximately 20 seconds for the former and 1 minute for the latter.

Figure 7 shows the ratio between the k-medians costs achieved by `CLNSS` and that of `approx-fmsi`, with $\gamma = 2$. For datasets of this scale, computing the exact medoid of every cluster would require an expensive post-processing step, defeating the purpose of the scalability evaluation. We therefore compute the k-medians cost using the centers maintained by each method: the centers produced by `CLNSS` and the inherited representatives maintained by `approx-fmsi`. In Section J, we provide evidence that the inherited representatives approximate the medoids well.

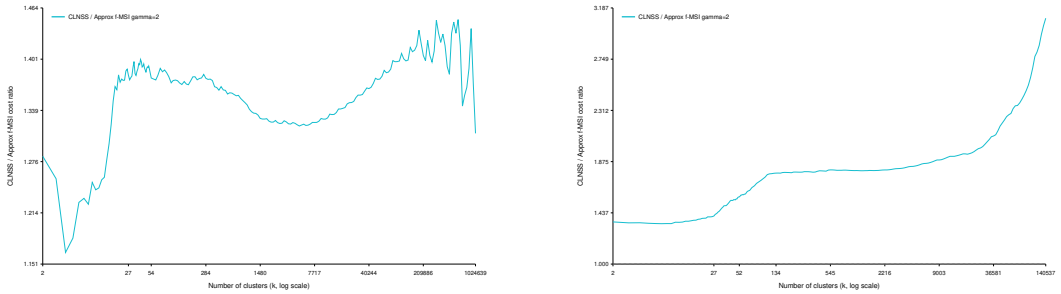


(a) Anuram (b) BankNote

(c) Wine (d) Ozone

(e) Geographical Music (f) Concrete

Figure 6: Ratio between the k-median cost of each method and that of `f-msi` for each value of $k < n/2$, for 6 representative datasets.



(a) Individual Household Electricity

(b) Wave Energy Converter

Figure 7: Ratio between the k-median cost of `CLNSS` and that of `approx-fmsi` with $\gamma = 2$.

I.6 Synthetic Data with Uniform Noise

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.

To assess clustering quality, we report the Adjusted Rand Index (ARI) (Hubert and Arabie, 1985). 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 8 - 13 show the ARI achieved for $n = 100$, $k = 10$, $\alpha = 0.1$ and 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.

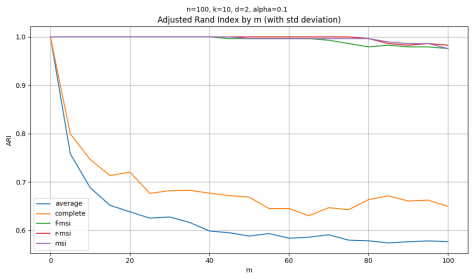


Figure 8: ARI for d = 2.

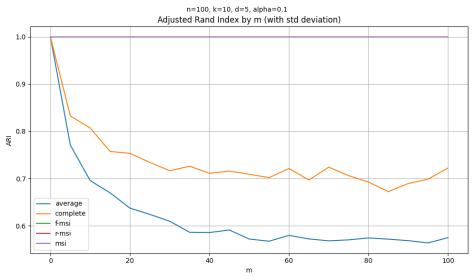


Figure 9: ARI for d = 5.

The results show that both `msi` and `f-msi` significantly outperform `complete-linkage` and `average-link`. The results of `msi` are slightly better than those of `f-msi`. Hence, `f-msi` not only provides compact clusters but also offers consistent empirical robustness across noisy datasets.

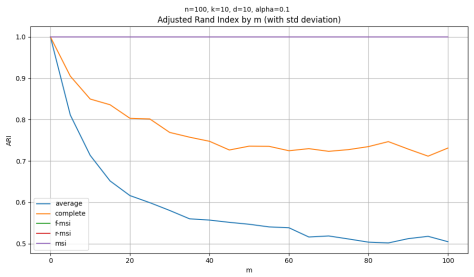


Figure 10: ARI for d = 10.

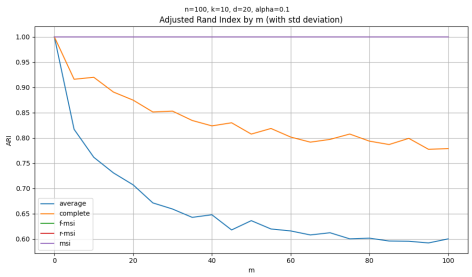


Figure 11: ARI for d = 20.

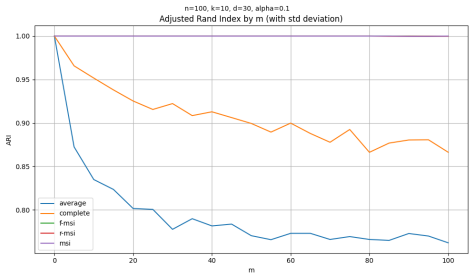


Figure 12: ARI for d = 30.

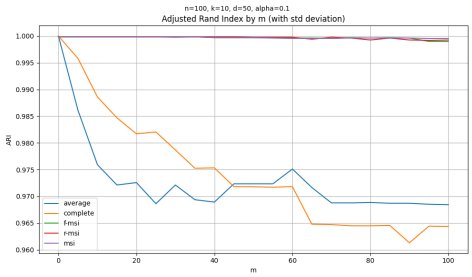


Figure 13: ARI for d = 50.

J THE PRICE OF INHERITED REPRESENTATIVES

Along the execution of `f-msi`, the representative of a cluster is never recomputed: when A and B are merged, with $|A| \geq |B|$, $r_{A\cup B}$ is set to r_A. Proposition 4.1 guarantees that this rule degrades the representatives in a controlled way: if r_A is an α-approximation to the medoid of A, then r_A is a $(4\alpha + 1)$-approximation to $m_{A\cup B}$. This appendix measures how pessimistic that factor is in practice.

The distinction matters because the two regimes of our experiments treat representatives differently. In Section I, except Section I.5, the k-medians cost of every method is evaluated with the medoids of the clusters it produces, obtained through the $O(n^2)$ post-processing of Section G.1. For the large instances of Section I.5, that post-processing is prohibitive and the cost is computed with the representatives maintained by the algorithm. The experiments below quantify the gap between these two regimes.

We compare two variants defined over the *same* dendrogram produced by `f-msi`: `f-msi-inherited`, in which each cluster is represented by the point it inherits during the merges, and `f-msi`, in which each cluster is represented by its medoid. Both variants induce identical partitions at every level of the hierarchy, so any difference between them is due exclusively to the choice of representatives.

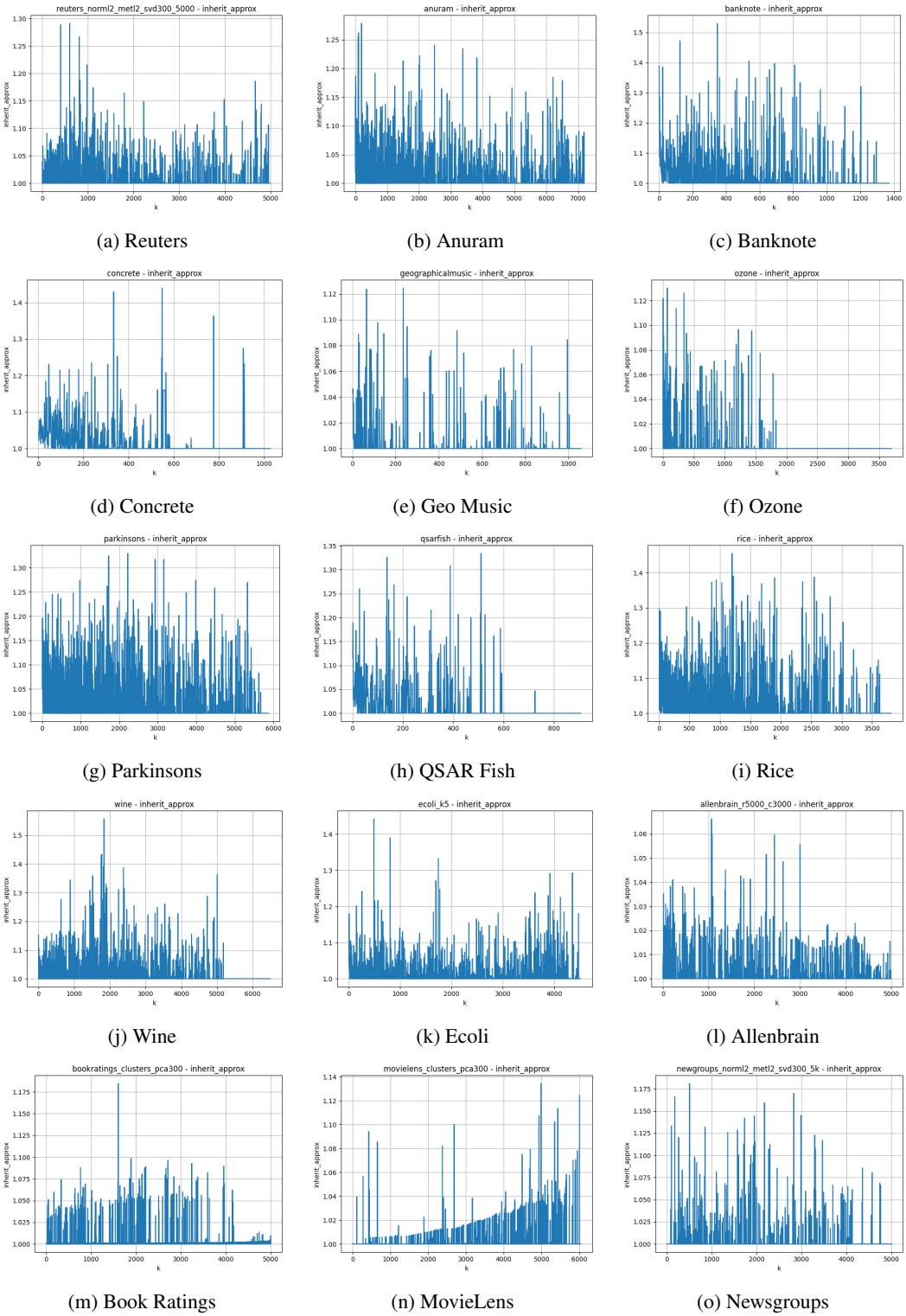
Inheritance ratio. For a cluster C created along the execution of `f-msi`, with inherited representative r_C and medoid m_C, its **inheritance ratio** is

$$\rho(C) \;:=\; \frac{\texttt{dsum}(r_C, C)}{\texttt{dsum}(m_C, C)},$$

with $\rho(C) := 1$ when both terms vanish, which covers all singletons. Since $r_C \in C$, we have $\rho(C) \geq 1$, with equality if and only if r_C is a medoid of C. In the terminology of Proposition 4.1, $\rho(C)$ is the smallest α for which r_C is an α-approximation for C.

Per-cluster behaviour. Figure 14 reports $\rho(C)$ for every cluster created along the hierarchy. The kth position of the horizontal axis from let to right corresponds to the cluster obtained by `f-msi` when $n - k$ clusters are actives while the vertical axis is the inheritance ratio of the cluster created at that merge. For $k \leq 100$, the average inherited ratio is 1.03 and in only 3% of these cases it is larger than 1.15. For $k > 100$, the average inherited ratio is smaller than 1.01.

Clustering-level behaviour. Figure 15 presents the comparison at the level of entire clusterings. For each k, we take the k-clustering induced by the dendrogram of `f-msi` and compute its k-medians cost under the representatives of `f-msi-inherited` and under those of `f-msi`, each normalised by the k-medians cost of the k-clustering built by `msi`. The plots follow the format of Figure 1, so the `f-msi` curve coincides with the one reported there and the vertical gap between the two curves is precisely the price of keeping the inherited representatives.



(a) Reuters (b) Anuram (c) Banknote

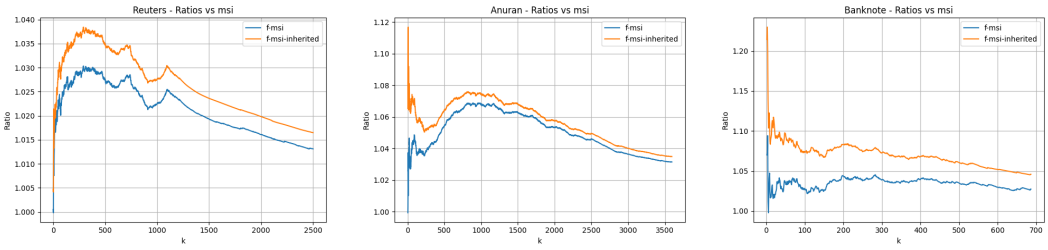
(d) Concrete (e) Geo Music (f) Ozone

(g) Parkinsons (h) QSAR Fish (i) Rice

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

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

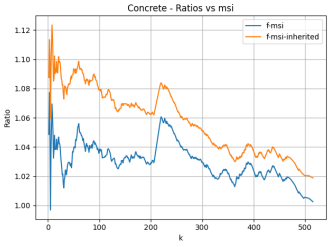
Figure 14: Inheritance ratio of `f-msi`.



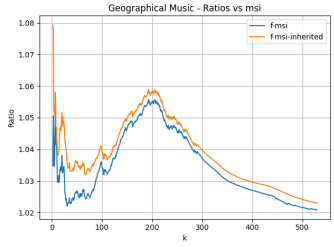
(a) Reuters

(b) Anuram

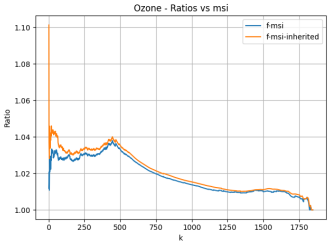
(c) Banknote



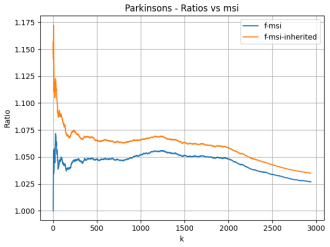
(d) Concrete



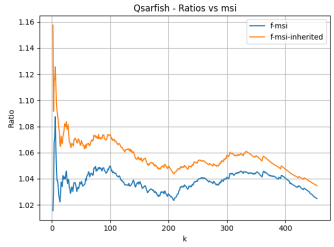
(e) Geo Music



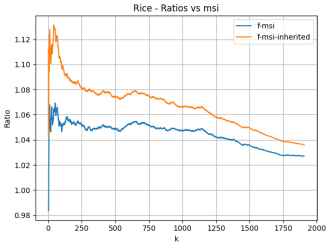
(f) Ozone



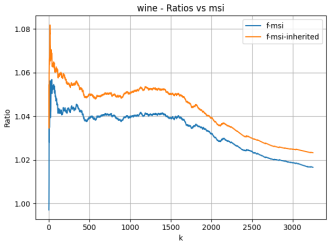
(g) Parkinsons



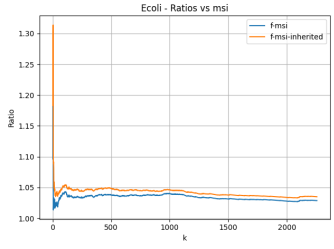
(h) QSAR Fish



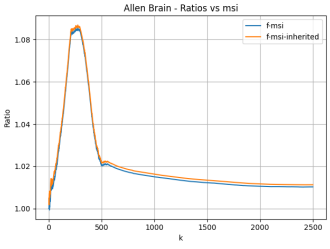
(i) Rice



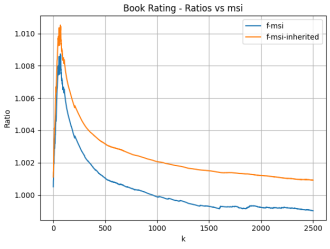
(j) Wine



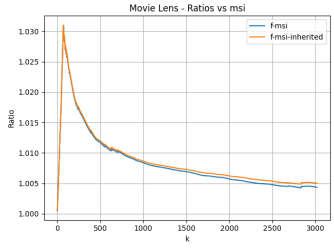
(k) Ecoli



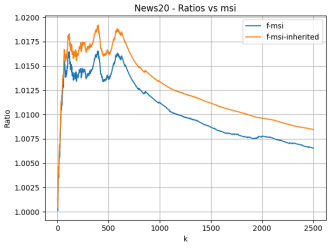
(l) Allenbrain



(m) Book Ratings



(n) MovieLens



(o) Newsgroups

Figure 15: Ratio between the k-median cost of each method and that of `msi` for each value of $k < n/2$