Pontifícia Universidade Católica
do Rio de Janeiro

Lucas Saadi Murtinho

Clustering under constraints: explainability via decision trees and separability with minimum size

Tese de Doutorado

Thesis presented to the Programa de Pós–graduação em Ciências – Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências – Informática.

Advisor: Prof. Eduardo Sany Laber

Rio de Janeiro
August 2024

Pontifícia Universidade Católica
do Rio de Janeiro

Lucas Saadi Murtinho

Clustering under constraints: explainability via decision trees and separability with minimum size

Thesis presented to the Programa de Pós–graduação em Ciências – Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências – Informática. Approved by the Examination Committee.

Prof. Eduardo Sany Laber
Advisor
Departamento de Informática – PUC-Rio

Prof. Marco Serpa Molinaro
Departamento de Informática – PUC-Rio

Prof. Edward Hermann Haeusler
Departamento de Informática – PUC-Rio

Prof. Diego Parente Paiva Mesquita
Escola de Matemática Aplicada – FGV

Prof. Daniel Ratton Figueiredo
Programa de Engenharia de Sistemas e Computação – UFRJ

Rio de Janeiro, August 23rd, 2024

Lucas Saadi Murtinho

Lucas Saadi Murtinho is a Master in Computer Science from PUC-Rio.

Bibliographic data

Saadi Murtinho, Lucas

Clustering under constraints: explainability via decision trees and separability with minimum size / Lucas Saadi Murtinho; advisor: Eduardo Sany Laber. – Rio de janeiro: PUC-Rio , Departamento de Informática, 2024.

v., 112 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Informática – Teses. 2. Otimização e raciocínio automático – Teses. 3. Clusterização;. 4. Explicabilidade;. 5. Árvores de decisão;. 6. Separabilidade;. 7. Tamanho mínimo.. I. Laber, Eduardo Sany. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

To Martin and Laila, for sharing their dad
with study and research.

Acknowledgments

I started my graduate studies with a keen interest in computer science in general and machine learning in particular, but without a clear plan of how to approach these subjects in a structured manner. I wish to thank my advisor, Eduardo Laber, not only for guiding me in this work, but also for starting to show me how I can build this structure myself. It should go without saying that any mistakes in this thesis are my full responsibility.

All the other professors under which I had the chance of studying during these two years were kind and generous, and I also had the great luck of finding many bright and fun minds among my fellow students. You are too many to mention here, but I hope you know who you are. Thanks.

This study was financed in part by the Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq – grant 141806/2020-6.

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

Abstract

Saadi Murtinho, Lucas; Laber, Eduardo Sany (Advisor). **Clustering under constraints: explainability via decision trees and separability with minimum size**. Rio de Janeiro, 2024. 112p. Tese de doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

We investigate two methods of clustering with constraints on the partitions being generated: *explainable clustering*, in which the partition must be induced by a binary decision tree (i.e., by cuts that are parallel to the axes); and *minimum-size clustering*, in which all clusters must have at least a predetermined number of elements.

For explainable clustering, we present theoretical algorithms and bounds for the k-centers, k-medians, k-means, and minimum-spacing cost functions. We also introduce three practical algorithms for the popular k-means cost function: `ExGreedy`, which presents results generally better than comparable algorithms in the literature; `ExShallow`, with a penalty term related to the depth of the tree that induces the partition, allowing for a trade-off between performance (reducing the cost function) and explainability (generating shallower trees); and `ExBisection`, to our knowledge the first explainable clustering algorithm based on decision trees for the k-means cost function that builds an explainable partition from scratch (i.e., without starting from an unrestricted partition).

For minimum-size clustering, our focus is on inter-clustering measures. We show that `Single-Linkage`, the algorithm that maximizes the minimum spacing, also maximizes the minimum-spanning-tree cost of a graph induced by the partition it generates; however, it is also prone to generating small clusters, which motivates the search for algorithms that perform well for these cost functions without suffering from this tendency. We introduce one approximation algorithm for each cost function, and present the results of experiments showing that they produce partitions that perform better than the popular k-means algorithm for these instances of the clustering task.

Keywords

Clustering; Explainability; Decision trees; Separability; Minimum size.

Resumo

Saadi Murtinho, Lucas; Laber, Eduardo Sany. **Clusterização sob restrições: explicabilidade via árvores de decisão e separabilidade com tamanho mínimo**. Rio de Janeiro, 2024. 112p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Investigamos dois métodos de clusterização com restrições nas partições geradas: a *clusterização explicável*, em que a partição deve ser induzida por uma árvore de decisão binária (ou seja, por cortes paralelos aos eixos); e a *clusterização de tamanho mínimo*, na qual todos os clusters devem ter pelo menos um número predeterminado de elementos.

Para a clusterização explicável, apresentamos algoritmos e garantias teóricas para as funções de custo k-centers, k-medians, k-means e espaçamento mínimo. Introduzimos também três algoritmos práticos para a popular função de custo k-means: `ExGreedy`, com resultados geralmente melhores do que os de algoritmos comparáveis na literatura; `ExShallow`, com um termo de penalidade relacionado à profundidade da árvore que induz a partição, permitindo um equilíbrio entre desempenho (redução da função de custo) e explicabilidade (geração de árvores mais rasas); e `ExBisection`, que, até onde sabemos, é o primeiro algoritmo de clusterização explicável baseado em árvores de decisão para a função de custo k-means que constrói uma partição explicável do zero (ou seja, sem usar uma partição irrestrita como ponto de partida).

Para a clusterização de tamanho mínimo, focamos em medidas inter-clusterização. Mostramos que `Single-Linkage`, o algoritmo que maximiza o espaçamento mínimo, também maximiza o custo da árvore de geração mínima de um grafo induzido pela partição gerada por ele; no entanto, este algoritmo tende a gerar muitos clusters pequenos, o que motiva a busca por algoritmos com bons resultados para essas funções de custo que garantam um número mínimo de elementos por cluster. Introduzimos um algoritmo de aproximação para cada função de custo e apresentamos os resultados de experimentos que mostram que eles produzem partições com melhores resultados do que o popular algoritmo k-means para essas instâncias do problema de clusterização.

Palavras-chave

Clusterização; Explicabilidade; Árvores de decisão; Separabilidade; Tamanho mínimo.

Table of contents

List of figures

List of tables

Cluster together like stars!

Henry Miller, *Stand Still Like the Hummingbird.*

1
Introduction

Clustering, or cluster analysis, is a very important task in computer science, with applications in a myriad of other disciplines. (Jain (2010)) dates the first appearance of "data clustering" in the title of an academic paper (on anthropology) back to 1954, and notes that "taxonomists, social scientists, psychologists, biologists, statisticians, mathematicians, engineers, computer scientists, medical researchers, and others who collect and process real data have all contributed to clustering methodology."

(Jain (2010)) also notes that "clustering is a more difficult and challenging problem than classification," and it's easy to make the case for this assertion: when dealing with labeled data, one can compare the model's results with the ground truth, whereas no such guidance exists when dealing with unlabeled data. The mere task of defining the number of clusters in which to partition a dataset can be hard in some cases. (For a contrarian view on the difficulty of clustering, see (Daniely et al. (2012)).)

Taking an even further step back, defining what a cluster *is* may be complicated. In hierarchical clustering, for instance, elements are sequentially connected one to the other in order of their similarity according to some measure. This may lead to very different results than clustering using a representative-based method such as Lloyd's algorithm (Lloyd (1982)), in which each cluster has a reference center, or representative point, and elements are associated to the cluster whose representative is closest to them. Figure 1.1, reproduced from (Scikit-Learn (2024)), shows how different clustering algorithms may partition the same dataset in different ways.

Selecting the best clustering algorithm is also, therefore, a matter of knowing how one wants the data to be clustered – that is, what is being optimized when the data is being partitioned. In the next section, we define the clustering task in general, and present a couple of examples before moving on to discuss the constraints that will guide our work in this thesis.

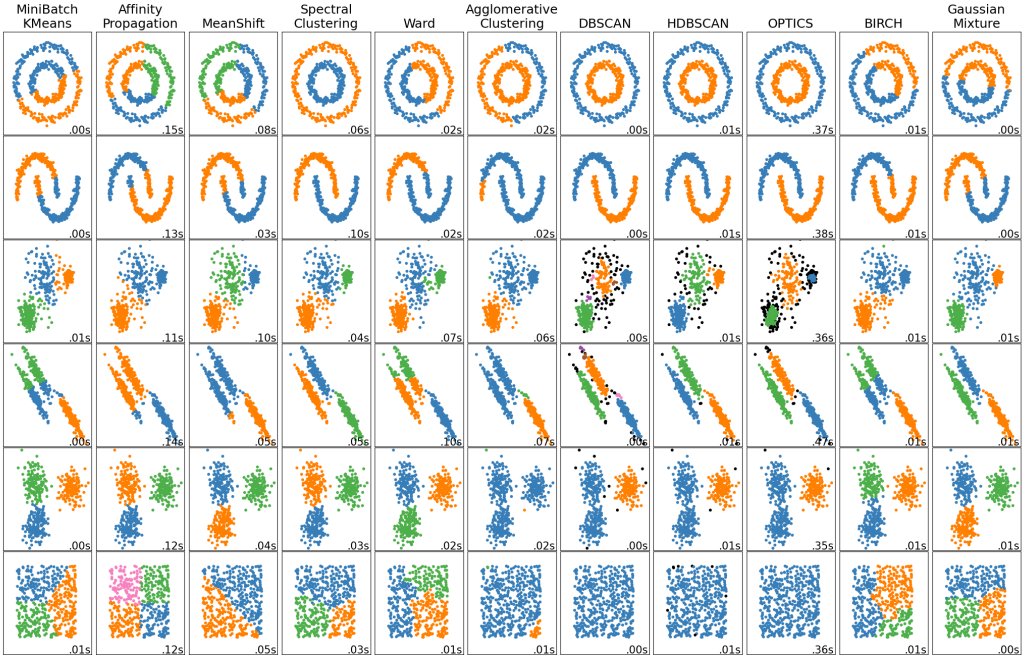


Figure 1.1: Comparison of partitions generated by different clustering algorithms on toy datasets (reproduced from (Scikit-Learn (2024)).

1.1 The clustering task and some of its versions

When partitioning a dataset, it is typically desired that data grouped in the same cluster be similar to each other (intra-cluster objectives) or that data grouped in different clusters be sufficiently distinct from each other (inter-cluster objectives). The definitions below briefly describe the clustering problems analyzed in this thesis.

Problem definition 1 (Clustering problem.) *Given a dataset $\mathcal{X}$ of n points, an integer $k \geq 2$, and a function $f(\cdot)$ mapping a partition $\mathcal{P}$ of the elements in $\mathcal{X}$ onto $\mathbb{R}$, the k-clustering problem consists in finding the k-partition $\mathcal{P}^*$ that optimizes f.*

Different functions lead to different versions of clustering problems. For the famous k-means clustering problem, for instance, the goal is to minimize

$$\text{k-means}(\mathcal{P}) = \sum_{i=1}^{k} \sum_{x \in P_i} ||x - \mu_i|_2^2, \tag{1-1}$$

where $\mathcal{P} = \{P_i | 1 \leq i \leq k\}$ is a partition of $\mathcal{X}$ with k groups and μ_i is the representative of P_i, assuming $x \in \mathcal{X} \subseteq \mathbb{R}^d$. The optimal representative μ_i for a partition P_i is known to be the centroid, or arithmetic mean, of all points in P_i.

For some tasks, it is desirable to maximize the distance between clusters. In the minimum spacing problem, for instance, our goal is to maximize

$$\text{Min-Spacing}(\mathcal{P}) = \min\{\texttt{spacing}(C_i, C_j) \,|\, C_i, C_j \in \mathcal{P}, i \neq j\}, \tag{1-2}$$

where

$$\texttt{spacing}(C_i, C_j) = \min\{\texttt{dist}(\texttt{x}, \texttt{y}) \,|\, \texttt{x} \in \texttt{C}_\texttt{i}, \texttt{y} \in \texttt{C}_\texttt{j}\} \tag{1-3}$$

and `dist` measures the distance between 2 points in $\mathcal{X}$; that is, we want to maximize the minimum distance between two points from different clusters.

However, the evaluation of a partition can go beyond these criteria, with additional constraints or preferences. In this thesis, we develop and evaluate clustering algorithms with two additional objectives:

Explainability. The selected partition should be easy to explain, according to some explainability criterion. For this thesis, a k-partition will be considered explainable if it can be induced by a binary decision with k leaves. (This somewhat crude definition will be expanded when discussing the `ExShallow` algorithm; see Sections 1.2 and 4.2.)

Avoiding small clusters. It may be desirable to prevent partitions from having a large number of very small clusters. In particular, the `Single-Linkage` algorithm, that is guaranteed to find the optimal partition for (1-2), has a strong tendency to produce a large number of small clusters, including singletons. So it may be useful to have algorithms for these problems that try to find partitions in which all clusters have a minimum number of elements.

The following sections delve deeper into these two clustering constraints.

1.2 Explainability

As machine learning models have become used in a wide range of fields, the topic of *explainability* has grown in importance. Understanding the reasoning behind a model's decision may be crucial to increase user confidence; satisfying legal requirements; conforming to moral and ethical expectations; and verifying the model's work. Since more complex models tend to be harder to interpret but are also more capable of returning good results, there is a trade-off between model performance and explainability. The challenge of navigating this trade-off is increasingly being explored in the machine learning and pattern recognition literature, with explainable models for specific tasks such as Covid-19 diagnostics using chest X-ray images (Malhotra et al. (2022)) and more general tasks such as anomaly detection (Mokoena et al. (2022)) and hyperspectral image classification (Shi et al. (2022)).

Although initial efforts towards explainability focused on supervised learning models (Burkart & Huber (2021)), a number of studies on explainable unsupervised models, and clustering models in particular, have appeared more recently. One idea that has earned considerable attention in the literature is to partition the data based on axis-aligned cuts, which can be induced by binary decision trees: at each node u of the tree, a value θ and a dimension i are selected, so that all data points that have reached u go to one of its two children according to whether their values for dimension i are smaller than θ or not. In this kind of approach, usually, each cluster is associated with a leaf.

Problem definition 2 (Explainable clustering problem.) *Given a dataset $\mathcal{X}$ of n points in $\mathbb{R}^d$, an integer $k \geq 2$, and a function $f(\cdot)$ mapping a partition $\mathcal{P}$ of the elements in $\mathcal{X}$ onto $\mathbb{R}$, the explainable k-clustering problem consists in finding the partition $\mathcal{P}^*$ that optimizes f, provided that $\mathcal{P}^*$ can be induced by a binary decision tree with k leaves.*

Restricting the universe of possible partitions to explained ones can lead to a loss in performance – i.e., the cost of the partition may increase. (Moshkovitz et al. (2020)) quantifies this with the *price of explainability*, or the maximum additional loss (in proportional terms) incurred by using the optimal explainable partition rather than the optimal unrestricted one.

Definition 1 (Price of explainability.) *Given a problem of the type described in Problem definition 2, its price of explainability is*

$$\rho(\mathcal{P}) = \max_I \left\{ \frac{OPT_{exp}(I)}{OPT_{unr}(I)} \right\} \tag{1-4}$$

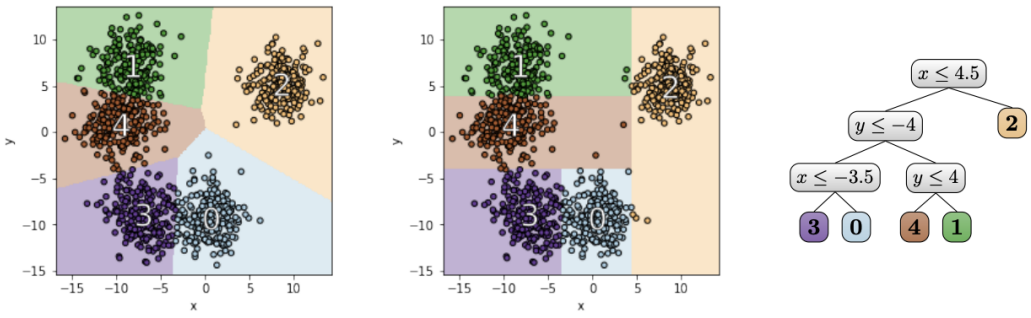
if the goal is to minimize a cost function and

$$\rho(\mathcal{P}) = \max_I \left\{ \frac{OPT_{unr}(I)}{OPT_{exp}(I)} \right\} \tag{1-5}$$

if the goal is to maximize a value function. In both equations, I runs over all instances of the problem; $OPT_{exp}(I)$ is the value of an optimal explainable clustering (via binary decision trees with k leaves) for instance I; and $OPT_{unr}(I)$ is the value of an optimal unrestricted clustering for I.

The best possible value for $\rho(\mathcal{P})$, when $OPT_{unr}(I) = OPT_{exp}(I)$ for all instances of the problem, is 1; the closer $\rho(\mathcal{P})$ is to 1, the better. We can naturally extend the definition above to consider the price of explainability for a specific instance of a problem.

Figure 1.2, reproduced from (Moshkovitz et al. (2020)), shows a toy dataset partitioned in two different ways. The unrestricted partition to the left



(a) Optimal 5-means clusters (b) Tree based 5-means clusters (c) Threshold tree

Figure 1.2: Example of unrestricted (left) and explainable (middle) 5-partitions, reproduced from (Moshkovitz et al. (2020)). The unrestricted partition to the left is optimal for the k-means cost function (1-1). The partition in the middle approximates the one from the left using the `IMM` algorithm from (Moshkovitz et al. (2020)), and it is explained by the decision tree to the right.

is optimal for the k-means cost function (1-1). The explainable partition in the middle, explained by the decision tree to the right, approximates the optimal one, but some points end up being reassigned to different representatives; as we'll see in Chapter 4, different ways of evaluating these reassignments lead to different algorithms for building explainable partitions based on unrestricted ones.

1.2.1 Our contributions

We proved some theoretical results of interest in regards to the price of explainability using different cost functions:

- For the minimum-spacing problem, we show that the price of explainability is $\Theta(n-k)$.
- For the k-centers problem, we present a lower bound of $\Omega\left(k\sqrt{d}\frac{\sqrt{\log\log k}}{\log^{1.5} k}\right)$ and an upper bound of $\mathcal{O}\left(\sqrt{d}k^{\frac{d-1}{d}}\right)$, which is tight for constant d.
- For the explainable k-medians problem, we present an algorithm with a price of explainability of $\mathcal{O}(d\log k)$, an improvement on the upper bound of $\mathcal{O}(k)$ from (Moshkovitz et al. (2020)) when $d < k/\log k$. This upper bound was later improved to $\mathcal{O}(\log k \log\log k)$ (Esfandiari et al. (2021), Gamlath et al. (2021), Makarychev & Shan (2021a)) and, later still, to $\mathcal{O}(\log k)$ (Makarychev & Shan (2023), Gupta et al. (2023)), matching the lower bound from (Moshkovitz et al. (2020)).
- For the k-means problem, our upper bound of $\mathcal{O}(dk\log k)$ improves on the bound of $\mathcal{O}(k^2)$ from (Moshkovitz et al. (2020)) when

$d < k/\log k$. This upper bound was later improved to $\mathcal{O}(k \log k)$ (Esfandiari et al. (2021), Gamlath et al. (2021)) and, later still, $\mathcal{O}(k \ln \ln k)$ (Gupta et al. (2023)), close to the lower bound of $\Omega(k)$ from (Moshkovitz et al. (2020)).

Table 1.1 shows the lower and upper bounds we provide for the four problems, as well as the best bounds currently found in other papers. Our theoretical results are presented in detail in Chapter 3.

Criterion	**Lower bound**	**Our upper bound**	**Best upper bound**
k-medians	$\Omega(\log k)$ [2]	$O(d \log k)$ [1]	$O(\log k)$ [3,4]
k-means	$\Omega(k)$ [2]	$O(dk \log k)$ [1]	$\mathcal{O}(k \ln \ln k)$ [4]
k-centers	$\Omega\left(\frac{\sqrt{d}k^{1-1/d}}{\log^{1.5} k}\right)$ [1]	$O\left(\sqrt{d}k^{1-1/d}\right)$ [1]	
minimum spacing	$\Theta(n-k)$ [1]		

Results from:
[1]: (Laber & Murtinho (2021)).
[2]: (Moshkovitz et al. (2020)).
[3]: (Makarychev & Shan (2023)).
[4]: (Gupta et al. (2023)).

Table 1.1: Lower and upper bounds for the price of explainability of different clustering problems.

On the practical side, we present three algorithms for explainable clustering with the popular k-means cost function (1-1):

1. `ExGreedy` (Section 4.1) iteratively selects axis-aligned cuts to minimize a cost function related to the k-means cost function. In this sense, it is similar to other algorithms in the literature, such as `IMM` (Moshkovitz et al. (2020)) and `ExKMC` (Frost et al. (2020)).

2. `ExShallow` (Section 4.2) introduces a penalty cost related to the depth of the generated tree, allowing for a trade-off between performance (minimizing (1-4)) and explainability (minimizing the tree's depth).

3. `ExBisection` (Section 4.3) is (to our knowledge) the first decision-tree algorithm explicitly designed for explainable k-means clustering that does not use as starting point a group of k representatives retrieved from an unrestrained (and therefore possibly not explainable) partition of the data.

For the development and presentation of `ExShallow`, we also expand on the discussion related to explainability by arguing that shallower trees are more explainable and should therefore be preferred. To that end, we use two

explainability metrics, the weighted average depth (`WAD`) of the tree and the weighted average explanation size (`WAES`) of each cluster. The latter metric, to the best of our knowledge, is presented in our work. Both are discusses in Section 4.2.1.

Three additional explainable clustering algorithms from the literature are presented in Section 4.4. In Section 4.5 we present the results of experiments comparing all six algorithms over 16 different datasets. `ExShallow` induces the best price of explainability in 10 of these datasets, while `ExBisection` presents the best results for explainability metrics in 7 of them. Furthermore, the algorithm with the best theoretical guarantees, `RandomThresholds` (Esfandiari et al. (2021), Gamlath et al. (2021), Makarychev & Shan (2021a)), performs poorly in these experiments, validating the search for heuristics that work well in practice.

1.3 Separability with minimum size constraints

`Single-Linkage` is a well-known hierarchical clustering algorithm that is guaranteed to obtain the k-partition that optimizes (1-2) – i.e., it maximizes the minimum distance between points from different clusters (Kleinberg & Tardos (2006)). However, it also tends to generate a large number of clusters with few elements, including singletons. Figure 1.3 shows the proportion of singletons in partitions generated by `Single-Linkage` for 10 different datasets as the number of clusters increases. Even when the number of clusters is small, the amount of singletons generated by the algorithm tends to be high.

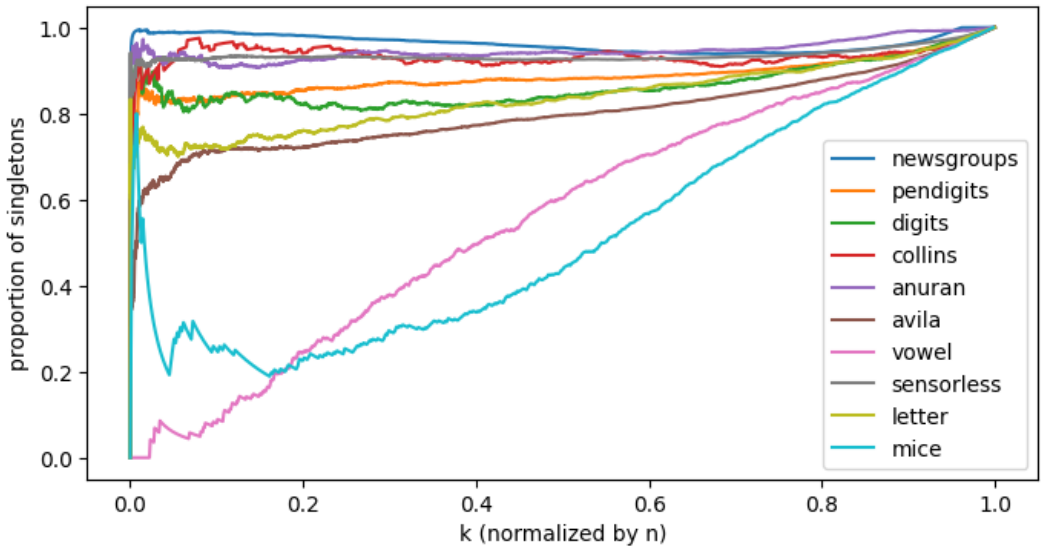


Figure 1.3: Proportion of singletons generated by `Single-Linkage`.

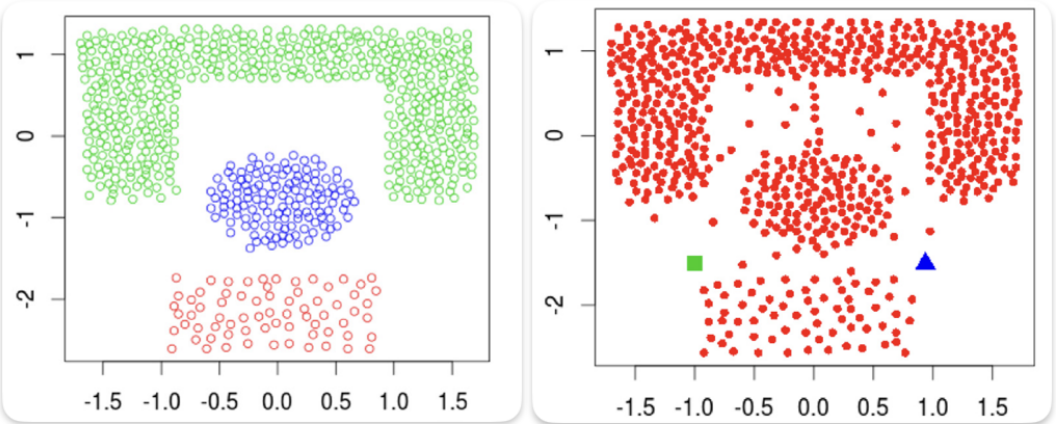


Figure 1.4: The effect of noise in the partition returned by `Single-Linkage` (reproduced from (Ros & Guillaume (2019)).

Figure 1.4, reproduced from (Ros & Guillaume (2019)), shows the potential downside of this behavior of `Single-Linkage`. The subfigure to the left shows the 3-partition returned by `Single-Linkage` for a "well-behaved" dataset, with three very well-defined clusters. The subfigure to the right shows how adding some noise to the dataset leads to `Single-Linkage` returning a completely different partition, with two singletons and the remaining points of the dataset all in a single cluster.

In some applications, it may be important to ensure both that clusters are well-separated and have a minimum size. When training a machine learning model, for instance, ensuring data diversity may be crucial for achieving good results (Gong et al. (2019)). In situations in which all available data cannot be used for training (e.g. training in the cloud with budget constraints), it is important to have a method for selecting a diverse subset of the data. It is therefore natural to define a new problem, in which the goal is to produce a partition with a restriction on the minimum size of the clusters.

Problem definition 3 (Minimum-size clustering problem.) *Given a dataset $\mathcal{X}$ of n points, an integer k, an integer L, and a function $f(\cdot)$ mapping a partition $\mathcal{P}$ of the elements in $\mathcal{X}$ onto $\mathbb{R}$, the minimum-size (k, L)-clustering problem consists in finding the k-partition $\mathcal{P}^*$ that optimizes $f(\mathcal{P}^*)$, provided that all k clusters in $\mathcal{P}$ have at least L points.*

Since intra-clustering problems tend not to have the issue of producing small clusters, our focus on Problem 3 will be on inter-clustering measures.

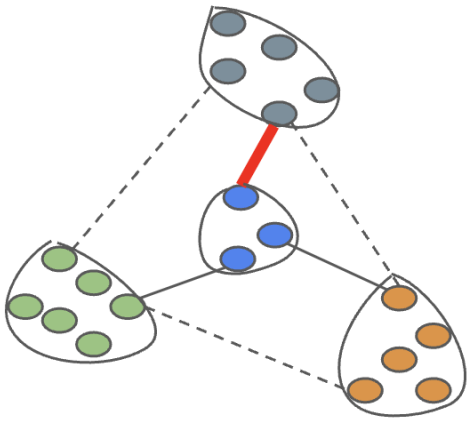


Figure 1.5: A toy example showing how `Min-Spacing` and `MST-Cost` are defined. The partition $\mathcal{P}$ of the dataset induces a complete graph $\mathcal{G}_\mathcal{P}$, in which each node represents a cluster from the partition. The spacing between any two clusters $C_1, C_2 \in \mathcal{P}$ is the smallest distance between a pair of points $x, y \,|\, x \in C_1, y \in C_2$; an edge $e \in \mathcal{G}_\mathcal{P}$ that connects the nodes representing C_1 and C_2 has weight equivalent to the spacing between C_1 and C_2. `Min-Spacing` is given by the weight of the red edge (the smallest distance between two points from different clusters); `MST-Cost` is the sum of the weights of the solid edges (including the red one), which combine to make the minimum spanning tree of the graph.

1.3.1 Our contributions

On the theoretical side, we show in Section 5.2 that `Single-Linkage` generates a partition $\mathcal{P}$ that maximizes not only `Min-Spacing` (1-2), as is well established in the literature (Kleinberg & Tardos (2006)), but also `MST-Cost`, whose introduction itself is a contribution of this thesis (see Section 5.2). `MST-Cost` is the cost of the minimum spanning tree of a complete weighted graph $\mathcal{G}_\mathcal{P}$ induced by the partition $\mathcal{P}$, in which each node represents a cluster and each edge is weighted by the spacing between the clusters associated to the nodes it connects. We use a toy example to illustrate both metrics in Figure 1.5.

We then present two algorithms with approximation guarantees for Problem 3 with these two cost functions:

- The first, presented in Section 5.3, returns a $(k, (1-\epsilon)L)$-clustering whose value for (1-2) satisfies Min-Spacing$(\mathcal{A}_t) \geq$ Min-Spacing$(\mathcal{C}^*)$, where $\mathcal{C}^*$ is the (k, L)-clustering with maximum `Min-Spacing` (1-2). The approximation we reach is essentially tight, as proven in Section 5.3.2.
- The second, presented in Section 5.4, is a $\left(\frac{(1-\epsilon)\rho}{2}, \frac{1}{H_{k-1}}\right)$-approximation for maximizing function `MST-Cost` (5-1), where $\rho := \min\left\{\frac{n/k}{L}, 2\right\}$ and

H_{k-1} is the $(k-1)$-th harmonic number. We prove in Section 5.4.1 that, unless $P = NP$, this problem is APX-HARD for a fixed k (with a hard constraint on the number of points per cluster).

In Section 5.5 we present the results of our experiments over 10 different real-world datasets, showing that our algorithms tend to produce partitions with much better separability results than the popular k-means algorithm (which we use as a baseline because it is much less likely to produce very small clusters than `Single-Linkage`), while allowing for a trade-off between separability and minimum size.

1.4 Published research

The research that makes up this thesis has been previously published in three academic papers:

1. (Laber & Murtinho (2021)) includes theoretical results (lower and upper bounds) for the price of explainability of four different explainable clustering problems, as well as a practical algorithm for explainable k-means. This work was accepted as a long paper (3% acceptace rate) in one of the most prestigious machine learning conferences in the world.

2. (Laber et al. (2023)) includes two new practical algorithms for explainable k-means clustering, as well as experiments comparing them to other algorithms from the literature.

3. (Laber & Murtinho (2023)) includes our results for separability with a minimum size of elements per cluster.

1.5 Thesis organization

We detail the relationship between our contributions and the literature in Chapter 2. Problem 2 (explainable clustering) is the subject of Chapters 3 and 4: in Chapter 3, we present theoretical results for four cost functions, while in Chapter 4 we present and discuss practical algorithms when using the popular k-means cost function (1-1). We discuss Problem 3 (separability with minimum size), focusing on the minimum spacing and MST inter-clustering criteria ((1-2) and (5-1)), in Chapter 5. Chapter 6 concludes the thesis.

2
Related work

We present in this chapter some important related work to contextualize the contributions presented in this thesis. Section 2.1 discusses the topic of explainability, first in general and then focusing on the explainable clustering framework established by (Moshkovitz et al. (2020)). In Section 2.1.1.1 we present a chronology of theoretical results for explainable clustering problems via decision trees, including our contributions, while in Section 2.1.1.2 we focus on practical algorithms for the explainable k-means problem. In Section 2.2 we discuss the topic of clustering with a guaranteed minimum size per clustering.

2.1
Explainability

Although the need for explainable machine learning models has been long advocated by some (see, for instance, (Bratko (1997), Caruana et al. (1999), Rani et al. (2006))), the topic has increased in importance in recent years, as these models have been more widely used for a large variety of topics and reasons. This has lead both to the development of new tools and methods to explain, interpret, or approximate complex models (Ribeiro et al. (2016), Lundberg et al. (2017), Bertsimas et al. (2018), Chen et al. (2019), Moshkovitz et al. (2020), Hüyük et al. (2021), Yin & Neubig (2022), Slack et al. (2023)) and to attempts to formally define explainability and intepretability. We summarize some of these below.

(Lipton (2018)) presents several desiderata of model interpretability, including: trust in the model (both in the sense that it's performing accurate predictions and that it's using the right set of variables to reach them); causality; transferability; informativeness; fair and ethical decision making. It also establishes different levels of interpretability: simulatability (when "a person can contemplate the entire model at once"), decomposability (when, for instance, we are able to easily understand each node in a decision tree, or each parameter in a linear regression – provided the input variables themselves are easy to understand), and algorithmic transparency (when we can understand how the model learned its parameters, for instance). It also mentions tools

for post-hoc interpretability, such as text explanations, visualization, local explanations (for instance, (Ribeiro et al. (2016), Lundberg et al. (2017))), and explanation by example (Caruana et al. (1999)).

(Doshi-Velez & Kim (2018)) presents a taxonomy of use cases for interpretable machine learning, including: scientific understanding, safety, ethics, mismatched objectives (when the model's objective is a proxy for the true goal of the task, and we want outside evaluation to verify whether this true goal is being achieved), and multi-objective trade-offs. It also proposes three evaluation approaches for model interpretability, from the more costly (and specific) to the less: application-grounded (asking humans with extensive domain knowledge to evaluate the model's explainability value in real-world tasks), human-grounded (performing experiments on simplified tasks to verify general human understanding of the model), and functionally-grounded (using some formal definition of interpretability to evaluate if the model meets it). We note that, by assuming decision trees are inherently explainable, we rely on the latter approach in this work.

(Rudin (2019)) argues that instead of relying on post-hoc, local explanations such as the ones introduced by (Caruana et al. (1999), Ribeiro et al. (2016), Lundberg et al. (2017)), we should focus on prioritizing the use of interpretable models. Our models `ExGreedy` (Section 4.1) and `ExShallow` (Section 4.2) can be seen as a middle-of-the-road approach: their starting point is the output of a potentially black-box model (an unrestricted partition of the data), which is approximated by a decision tree that can then be used as an independent, and fully explainable, model. Our third practical model for k-means explainable clustering, `ExBisection` (Section 4.3), does not rely on an initial partition of the data and is therefore a more straightforward fully explainable model in the vein proposed by (Rudin (2019)).

2.1.1 Explainable clustering via decision trees

In (Moshkovitz et al. (2020)), a polynomial-time algorithm is presented that receives a (non-explainable) partition $\mathcal{P}_u$ to the k-means (or k-medians) clustering problems and builds a decision tree, in a top-down fashion, by selecting at each node the cut that, among those that separate at least two representatives in $\mathcal{P}_u$, minimizes the number of data points separated from their representatives in $\mathcal{P}_u$. The same paper includes a proof that the price of explainability (1-4) for this algorithm is $\mathcal{O}(k^2)$ for the k-means problem.

After (Moshkovitz et al. (2020)), new algorithms with improved bounds on the price of explainability for different clustering problems were proposed:

Explainable k-centers. We present in (Laber & Murtinho (2021)) a lower bound of $\Omega(k^{1-1/d})$ for small values of d with respect to k and $\Omega\left(k\sqrt{d}\frac{\sqrt{\log\log k}}{\log^{1.5} k}\right)$ otherwise, and an upper bound of $\mathcal{O}\left(\sqrt{d}k^{\frac{d-1}{d}}\right)$ which is tight for constant values of d. These results are reproduced in Section 3.1 below. For the cases when $d = \Omega(\log k)$, the lower bound was tightened to $\Omega(k\sqrt{d})$ by (Esfandiari et al. (2021)).

Explainable k-medians with ℓ_2 norm. (Makarychev & Shan (2021b)) presents the first, and to our knowledge only, analysis of this problem, showing a lower bound of $\Omega(\log k)$ and an upper bound of $\mathcal{O}(\log^{3/2} k)$ for its price of explainability.

Apart from these results, the research following (Moshkovitz et al. (2020)) focused on explainable k-medians and k-means clustering. We summarize its main results below.

2.1.1.1 Theoretical results for explainable k-medians and k-means

(Moshkovitz et al. (2020)) is the first article in the literature to formally define the problem of explainable clustering via decision trees (Problem 2). As well as introducing the concept of price of explainability (1-4), it presents the first theoretical guarantees for the explainable k-means and k-medians cost functions: it proves that the price of explainability (when using a tree with k leaves) is $\Omega(\log k)$ for k-medians and $\Omega(k)$ for k-means, and shows an algorithm that yields an upper bound of $\mathcal{O}(k)$ for k-medians and $\mathcal{O}(k^2)$ for k-means. The algorithm, `IMM`, starts from an unrestricted partition of the data, $\mathcal{P}_u$ and iteratively finds the axis-aligned cuts that minimize the number of points separated from their representatives in $\mathcal{P}_u$; we describe it in more detail in Section 4.4.1. The same paper shows that adapting traditional decision-tree algorithms, which split the data based on information-theoretic criteria such as entropy or information gain, to the explainable-clustering problem may yield arbitrarily bad results, motivating the search for algorithms specifically built for this problem.

Our contributions. (Laber & Murtinho (2021)) presents an algorithm with a price of explainability of $\mathcal{O}(d \log k)$ for k-medians and $\mathcal{O}(dk \log k)$ for k-means, improving the bounds from (Moshkovitz et al. (2020)) when $d < k/\log k$. The algorithm also uses an unrestricted partition as a starting point, and iteratively

selects cuts that will split the data along the dimension of maximum diameter. We present these contributions in Sections 3.2 and 3.3.

Improvements over our contributions. The algorithm that achieves the best price of explainability guarantees for the explainable k-medians and k-means problems, `RandomThresholds`, also starts from an unrestricted partition $\mathcal{P}_u$, similar to the algorithms from (Moshkovitz et al. (2020), Laber & Murtinho (2021)). However, instead of deterministically finding the axis-aligned cuts that optimize some characteristic of the partition (such as the number of mistakes or the diameter of the dimension), it separates the representatives from $\mathcal{P}_u$ by randomly selecting axis-aligned cuts.

Three independent papers presented closely related versions of this algorithm, highlighting different properties of it:

- (Esfandiari et al. (2021)) shows that the price of explainability for `RandomThresholds` is $\mathcal{O}(\log k \log\log k)$ for k-medians, or $\mathcal{O}(d \log^2 k)$ when d is small with relation to k; for k-means, the prices of explainability is $\mathcal{O}(k \log k)$. For both problems, this improves on the results from (Moshkovitz et al. (2020), Laber & Murtinho (2021)).
- (Gamlath et al. (2021)) extends the analysis of `RandomThresholds` to the ℓ_p-norm, achieving a lower bound of $\Omega(k^{p-1})$ and an upper bound of $\mathcal{O}(k^{p-1} \log^2 k)$ for the price of explainability. It also highlights that the algorithm's running time does not depend on the size of the dataset, n, as it is only preoccupied with separating representatives. The paper also includes a conjecture that the k-medians algorithm is asymptotically optimal, with an actual price of explainability of $1 + H_{k-1}$, where H_n is the n-th harmonic number (which is bounded by $\ln n \leq H_n \leq 1 + \ln n$).
- (Makarychev & Shan (2021a)) reaches the same price of explainability for explainable k-medians as (Esfandiari et al. (2021)), and a slightly worse one (by an $\mathcal{O}(\log k \log\log k)$ factor) for explainable k-means.

Two papers focus on the explainable k-means problem. By improving our idea, presented in (Laber & Murtinho (2021)), of selecting cuts on dimensions with large diameters, (Charikar & Hu (2021)) finds an upper bound of $\mathcal{O}(k^{1-2/d}\text{poly}(d \log k))$ for the price of explainability of k-means, an improvement on previous algorithms when $d = \mathcal{O}(\log k / \log\log k)$. (Makarychev & Shan (2021b)) expands the ideas of the random cut algorithm from (Makarychev & Shan (2021a)) to allow for additional leaves in the tree; its algorithm returns a tree with $(1+\delta)k$ leaves and has a price of explainability of $\mathcal{O}(1/\delta \log^2 k \log\log k)$.

Both (Makarychev & Shan (2023)) and (Gupta et al. (2023)) tighten the analysis of `RandomThresholds` for explainable k-medians, proving that its price of explainability matches the lower bound of $\Omega(\log k)$ from (Moshkovitz et al. (2020)). (Gupta et al. (2023)) also tightens the k-means price of explainability to $\mathcal{O}(k \ln \ln k)$, conjecturing that the lower bound of $\Omega(k)$ from (Moshkovitz et al. (2020)) is tight and can be achieved using `RandomThresholds`. It also proves the conjecture from (Gamlath et al. (2021)) bounding the k-medians price of explainability to the $(k-1)$-th harmonic number.

(Laber (2024)) and (Gupta et al. (2023)) independently prove hardness results for the explainable k-means and k-medians problems, thus consolidating the motivation for exploring practical algorithms for this problem. The former shows that the k-means explainable clustering problem is hard to approximate, while the latter shows that explainable k-medians and k-means cannot be approximated better than $\mathcal{O}(\ln k)$.

(Bandyapadhyay et al. (2023)) presents exact algorithms, based on dynamic programming, to solve explainable k-medians in $n^{2d}(nd)^{\mathcal{O}(1)}$ time. It also presents an algorithm that approximates the optimal k-medians clustering from the perspective of outlier removal: if we can remove εn points from the original dataset ($0 \leq \varepsilon < 1$), the problem is solvable in $\left(\frac{4d(k+\varepsilon)}{\varepsilon}\right)^k n^{\mathcal{O}(1)}$ time.

2.1.1.2 Practical algorithms for explainable k-means

To our knowledge, the first algorithm to partition a numerical dataset using decision trees is `CLTree` (Liu et al. (2005)), which finds clusters by adding synthetic points to a dataset and then building a binary classification tree to determine if a region is predominantly populated by real or synthetic points. Regions predominantly populated by real points are considered clusters, and real points in sparsely populated regions may be treated as small clusters or outliers. The algorithm includes a pruning step that is implicitly used to make the induced partition on the data more explainable. A similar approach, with a more direct appeal to the interpretability of decision, is used in `CUBT` (Fraiman et al. (2013)), in which a large tree is built and then pruned, and more than one leaf may have the same label (i.e., the tree may have more than k leaves).

Additional algorithms were developed after the price of explainability framework established in (Moshkovitz et al. (2020)). (Frost et al. (2020)) expands on `IMM` (see above) by allowing for trees with more than k leaves. The

new algorithm, `ExKMC`, uses the concept of a *surrogate cost*, which calculates the cost of a new split without recalculating the centroids of the clusters it generates. With these fixed centers, the computational cost of the algorithm is reduced, but the surrogate cost is still an upper bound on the k-means cost (as the cost can only decrease by using the mean of the points in a cluster as their representative). `ExKMC` works by finding the i-th cut that will lead to the explainable $(i+1)$-partition of minimum cost, considering the fixed centers. (Frost et al. (2020)) also includes empirical evaluations with real-world data, showing that `ExKMC` and `IMM` perform better for explainable-clustering tasks than traditional decision-tree algorithms, and confirming in practice the theoretical need for specialized algorithms shown in (Moshkovitz et al. (2020)).

Our contributions. In (Laber & Murtinho (2021)) we present `ExGreedy`, a practical algorithm for explainable k-means. In the same vein as `IMM` and `ExKMC`, `ExGreedy` starts from an unexplained partition $\mathcal{P}_u$, and at each iteration selects a cut that separates at least two of its representatives from each other. While `IMM` selects the cut that minimizes the number of elements separated from their original representatives, and `ExKMC` minimizes the surrogate cost, `ExGreedy` selects the i-th cut that minimizes the k-means cost of the partition generated by it, using the same representatives as those from $\mathcal{P}_u$. The algorithm is presented in detail in Section 4.1. The same paper includes experiments showing that `ExGreedy` tends to produce partitions with lower prices of explainability than both `IMM` and `ExKMC`.

We present in (Laber et al. (2023)) an additional consideration: ideally, the decision tree that induces an explainable partition should be shallow rather than deep, as shallow trees are considered easier to interpret (Piltaver et al. (2016)). We then introduce `ExShallow`, an adaptation of `ExGreedy` that selects the next cut according not only to the objective function described above, but also to a penalty term associated to the imbalance generated by the cut (as more balanced cuts tend to lead to shallower trees). This allows for a trade-off between performance (lower price of explainability) and explainability (shallower trees). We present `ExShallow` in more detail in Section 4.2.

In (Laber et al. (2023)) we also present `ExBisection`, which to our knowledge is the first explainable k-means algorithm that builds an explainable partition from scratch – that is, without relying on an initial, unexplained partition. We present it in Section 4.3.

The experiments from (Laber et al. (2023)) compare the performance of `IMM`, `ExKMC`, `ExGreedy`, `ExShallow`, `ExBisection`and `RandomThresholds` on

16 real-world datasets, and are reproduced in Section 4.5.

Other related work. A number of papers (Fraiman et al. (2013), Bertsimas et al. (2018), Saisubramanian et al. (2020), Ghattas et al. (2017)) propose decision-tree algorithms to build partitions that optimize other cost functions beyond (1-1). The importance of building shallow trees for achieving interpretability has been previously discussed in (Blanco et al. (2020)), in which clustering and decision trees (constructed with the CART algorithm (Breiman (2017))) are used to locally interpret the results of a black-box model.

Although `ExShallow` presents good results in practice, using $k-1$ features (and therefore having a tree with k leaves of maximal depth) may be necessary. (Moshkovitz et al. (2020)) presents a simple dataset that can only be explained by a tree of depth $k-1$. (Deng et al. (2023)) defines the *price of depth reduction* as "the ratio of the clustering objective cost of the clustering given by the best decision tree of [limited depth] to the cost of the clustering given by the optimal decision tree", and shows instances, both in the plane and in high dimensions, for which the price of depth reduction is unbounded for k-means, k-medians, and (in the high-dimension case) k-centers.

The idea that decision trees are intrinsically explainable has recently been challenged by (Izza et al. (2022)), which argues that decision trees frequently exhibit *path explanation redundancy*, with unnecessary tests to explain a given classification. We note that, in `ExShallow`, path redundancy is seen as a feature rather than a bug – the reasoning being that, if the same variable is used for different cuts, this can mean a more compact explanation for a specific cluster, if not for the tree as a whole.

2.2 Separability with minimum size constraints

2.2.1 Separability: inter-clustering criteria

The maximum k-cut problem is a widely studied problem in the combinatorial optimization community and its solution can be naturally viewed as a clustering that optimizes a separability criterion. Given an edge-weighted graph G and an integer $k \geq 2$, the maximum k-cut problem consists of finding a k-cut (or a partition of the vertices of the graph into k groups) with maximum weight, where the weight of a cut is given by the sum of the weights of the edges that have vertices in different groups of the cut. The weight of the k-cut

can be seen as a separability criterion, where the distance between two groups is given by the sum of the pairwise distances of its points. It is well-known that a random assignment of the vertices (points) yields an $(1-1/k)$-approximation algorithm. This bound can be slightly improved using semi-definitive programming (Frieze & Jerrum (1997)).

As already mentioned, the cost function of the minimum spacing (1-2) can be maximized in polynomial time via the `Single-Linkage` algorithm (Kleinberg & Tardos (2006))[Chap 4.7], which has been the subject of a number of researches (Zahn (1971), Kleinberg (2002), Carlsson & Facundo (2010), Hofmeyr (2020)). (Kleinberg (2002)) presents an axiomatic study of clustering, the main result of which is a proof that it is not possible to define clustering functions that simultaneously satisfy three desirable properties introduced in the paper. However, it was shown that by choosing a proper stopping rule for the `Single-Linkage` it satisfies any two of the three properties. (Carlsson & Facundo (2010)) replaces the property of (Kleinberg (2002)) that the clustering function must output a partition with the property that it must generate a tree (dendogram), and then establishes that `Single-Linkage` satisfies the new set of properties. More recently, (Hofmeyr (2020)) establishes a connection between minimum spacing and spectral clustering. While the aforementioned works prove that `Single-Linkage` has important properties, in practice, it is reported that sometimes it presents poor performance due to the so-called chaining effect ((Jain et al. (1999))[Chap 3.2]).

2.2.2 Hierarchical Agglomerative Clustering

`Single-Linkage` belongs to the family of algorithms that are used to build Hierarchical Agglomerative Clustering (HAC), surveyed in (Murtagh (1983), Murtagh & Contreras (2012), Murtagh & Contreras (2017)). (Dasgupta (2016)) frames the problem of building a hierarchical clustering as a combinatorial optimization problem, where the goal is to output a hierarchy/tree that minimizes a given cost function, and proposes both a cost function that has desirable properties and an algorithm to optimize it. This result was improved and extended by a series of papers (Roy & Pokutta (2016), Charikar & Chatziafratis (2017), Cohen-Addad et al. (2019)).

The algorithms discussed in the papers do not seem to be employed in practice; recently, there has been some effort to analyse more popular HAC algorithms such as `Average-Linkage`, in which the distance between two clusters is the average distance between the elements in each cluster (Moseley & Wang (2017), Cohen-Addad et al. (2019),

Dasgupta & Laber (2024)). (Dasgupta & Long (2005)) shows a lower bound on the maximum diameter of `Single-Linkage`; (Arutyunova et al. (2024)) proves that this bound is tight. Our investigation of `Single-Linkage` can be naturally connected with this line of research.

2.2.3 Clustering with minimum-size guarantees

We are aware of only a few studies on guaranteeing a minimum size for each cluster. (Bradley et al. (2000)) notices that Lloyd's algorithm for the k-means problem tends to return clusters that are empty or have very few points when clustering data with many dimensions ($d \geq 10$) in a large number of clusters ($k \geq 20$), and proposes an adaptation of the algorithm (transforming its assignment phase to an equivalent of the minimum-cost-flow problem) that guarantees a minimum number of elements per cluster. Some experimental results from the paper suggest that, depending on the number of clusters and the minimum size required, this constraint may help find better local minima than the unconstrained version of Lloyd's algorithm.

(Ganganath et al. (2014)) presents a modified k-means algorithm so that clusters have a *maximum* size. In the assignment step, a point is assigned to a representative only if it's one of the c closest points to that representative, where c is the maximum size allowed. It also introduces an initialization process to improve the results, although this requires some previous knowledge about the data set.

(Ros & Guillaume (2019)) presents two algorithms for dealing with the chaining effect on `Single-Linkage`. The first one includes a density estimation to label some data points as noise, while the second one takes as input the number of desired clusters and guarantees they all have a minimum size proportional to the size of the dataset. The second one keeps track of the number of elements that are in a representative cluster (i.e., a cluster above the minimum established size) and keeps merging clusters until the minimum desired proportion of points are in representative clusters; these clusters are then returned. In both algorithms, some points will not belong to any clusters returned, ending up being labeled as noise.

3
Theoretical results

This chapter presents the theoretical results of our research on Problem 2, with lower and upper bounds for the price of explainability (Definion 1) of four different clustering problems: k-centers (using cost function (3-1)), k-medians (cost function (3-2)), k-means (1-1), and minimum spacing (1-2).

3.1
Explainable k-centers

Explainable k-centers is the problem of finding a k-partition induced by a binary decision tree that minimizes

$$\text{k-centers}(\mathcal{P}) = \max_{P \in \mathcal{P}} \max_{x \in P} ||x - \mu_P||_2, \tag{3-1}$$

where μ_P is the representative point associated to cluster $P \in \mathcal{P}$; i.e., the cost of a partition is the maximum distance between a point and the representative to which it is associated. Below we present lower and upper bounds for the price of explainability (1-4) of this problem.

3.1.1
Lower bound

Let $p \leq \min\{d, \log_3 k\}$ be a positive integer and b the largest integer for which $b^p \leq k$. Note that $b \geq 3$. Moreover, let $k' = b^p$.

Our instance I_{kc} has $k + k' \cdot 2d$ points. We first discuss how to construct the k points, referred as centers, that will be set as representatives in an unrestricted k-clustering for I_{kc} that has a low cost. The first k' centers will be obtained from the representation of the numbers $0, \ldots, k'-1$ in base b while the remaining $k - k'$ centers will be located sufficiently far from the others so that they will be isolated in the low-cost k-clustering for I_{kc}. Let $\mathbf{c}^0, \ldots, \mathbf{c}^{k'-1}$ be the first k' centers.

For a number $i \in [k'-1]$ let $(i_{p-1}, \ldots, i_0)_b$ be its representation in base b. For $j \in [d]$, the value of the j-th component of center $\mathbf{c}^i$ is obtained by applying $(j-1)$ times a circular shift on $(i_{p-1}, \ldots, i_0)_b$. The values of the remaining $d-p$ components of $\mathbf{c}^i$ are obtained by copying the p first values d/p times so that $c^i_j = c^i_{j'}$ if $(j - j') \mod p = 0$.

As an example, if $b = 3$, $p = 3$ and $d = 9$ then $\mathbf{c}^{14} = (14, 22, 16, 14, 22, 16, 14, 22, 16)$. In fact, since $14 = (1, 1, 2)_3$ we have that $c_1^{14} = (1, 1, 2)_3 = 14$; $c_2^{14} = (2, 1, 1)_3 = 22$ and $c_3^{14} = (1, 2, 1)_3 = 16$. The values of $c_4^{14}, \ldots, c_9^{14}$ are obtained by repeating the first 3 values.

The following observation is useful for our analysis.

Fact 1 *For every $\ell \in [p]$, the values of the ℓ-th coordinate of the k' first centers are a permutation of the integers $0, \ldots, k' - 1$.*

The remaining $k - k'$ centers, as mentioned above, should be far from each other and also far away from the k' first centers. We can achieve that by setting $\mathbf{c}^i = k^i\mathbf{1}$ for all $i > k' - 1$, where $\mathbf{1}$ is the unit vector in $\mathbb{R}^d$.

The next lemma gives a lower bound on the distance between any two centers.

Lemma 1 *For any two centers $\mathbf{c}^i$ and $\mathbf{c}^j$,*

$$||\mathbf{c}^i - \mathbf{c}^j||_2 \geq \sqrt{\lfloor d/p \rfloor} \cdot (b^{p-1}/2).$$

Proof. If one of the two centers is not among the k' first centers the result clearly holds. Thus, we assume that $i, j \leq k' - 1$.

It is enough to show that there is $\ell \in [p]$ for which $|c_\ell^i - c_\ell^j| \geq b^{p-1}/2$. In fact, if this inequality holds for some ℓ then $|c_{\ell'}^i - c_{\ell'}^j| \geq b^{p-1}/2$ for each ℓ' that is congruent to ℓ modulo p. Since there are $\lfloor d/p \rfloor$ of them, due to our construction, we get the desired bound.

Let $i = (i_{p-1}, \ldots, i_0)_b$ and $j = (j_{p-1}, \ldots, j_0)_b$ be the representations of i and j in base b, respectively. Let f be such that $|i_f - j_f|$ is maximum.

Thus, the difference between $\mathbf{c}^i$ and $\mathbf{c}^j$ in the coordinate $[(f + 1) \bmod p] + 1$ is at least

$$|i_f - j_f| \cdot \left(b^{p-1} - \sum_{g=0}^{p-2} b^g \right) \geq b^{p-1}/2,$$

where the last inequality holds because $|i_f - j_f| \geq 1$ and $b \geq 3$. ■

Now, we define the remaining points of instance I_{kc}.

For each of the first k' centers we create $2d$ associated points: $\mathbf{x}^{i,1}, \ldots, \mathbf{x}^{i,2d}$. For $j = 1, \ldots, d$, the point $\mathbf{x}^{i,2j-1}$ is identical to $\mathbf{c}^i$ in all coordinates but on the j-th one, in which its value is $c_j^i - 3/4$. Similarly, the point $\mathbf{x}^{i,2j}$ is identical to $\mathbf{c}^i$ in all coordinates but in the j-th one, in which its value is $c_j^i + 3/4$. By considering the k-clustering for I where the k representatives are the k centers $\mathbf{c}^0, \ldots, \mathbf{c}^{k-1}$ and each point $\mathbf{x}^{i,j}$ lies in the group of $\mathbf{c}^i$, we obtain the following proposition.

Proposition 1 *There exists an unrestricted k-clustering for instance I_{kc} with cost $3/4$.*

Now we analyse the cost of an optimal explainable clustering for I_{kc}. The following proposition is a simple consequence of Fact 1.

Proposition 2 *Let (j, θ) be a cut (across dimension j at value θ) that separates at least two points from the set A that includes the k' first centers and its associated $k' \cdot 2d$ points. Then, (j, θ) separates one point from its associated center.*

Proof. Since (j, θ) separates at least two points from A, then $\theta \in (-3/4, k' - 1 + 3/4)$. If $\theta < 0$, then (j, θ) separates the center that has the j-th coordinate equal to 0 from its associated point that has coordinate j equal to $-3/4$. If $\theta > k' - 1$, then (j, θ) separates the center that has the j-th coordinate equal to 0 from its associated point that has coordinate j equal to $k' - 1 + 3/4$.

Let z be an integer that satisfies both $0 \leq z \leq k' - 2$ and $\theta \in (z, z + 1)$. If $\theta - z < 1/2$ (resp. $\theta - z > 1/2$), (j, θ) separates the center that has the j-th coordinate equal to z (resp. $z + 1$) from its associated point with j-th coordinate equal to $z + 3/4$ (resp. $z + 1 - 3/4$). The existence of centers with the aforementioned values for coordinate j is guaranteed by Fact 1. ■

Lemma 2 *Any explainable k-clustering for instance I_{kc} has cost at least $\sqrt{\lfloor d/p \rfloor} \cdot (b^{p-1}/4) - 3/8$.*

Proof. Let $\mathcal{C}$ be an explainable k-clustering for instance I_{kc}. It is enough to show that there is a cluster $C \in \mathcal{C}$ that contains two points, say $\mathbf{x}$ and $\mathbf{y}$, for which

$$||\mathbf{x} - \mathbf{y}||_2 \geq \sqrt{\lfloor d/p \rfloor} \cdot (b^{p-1}/2) - 3/4.$$

In fact, in this case, due to the triangle inequality, for any choice of the representative for C, either $\mathbf{x}$ or $\mathbf{y}$ will be at distance at least $\sqrt{\lfloor d/p \rfloor} \cdot (b^{p-1}/4) - 3/8$ from it.

If two centers lie in the same cluster of $\mathcal{C}$ then it follows from Lemma 1 that their distance is at least $\sqrt{\lfloor d/p \rfloor} \cdot (b^{p-1}/2)$.

On the other hand, if every center lies on a different cluster in $\mathcal{C}$ then let $\mathbf{x}$ be the point that was separated from its center, say $\mathbf{c}^i$, by a cut that satisfies the condition of Proposition 2. Then, $\mathbf{x}$ lies in the same cluster of $\mathbf{c}^j$, for some $j \neq i$. From the triangle inequality we have that

$$||\mathbf{c}^i - \mathbf{c}^j||_2 \leq ||\mathbf{c}^i - \mathbf{x}||_2 + ||\mathbf{c}^j - \mathbf{x}||_2.$$

Hence, $||\mathbf{c}^j - \mathbf{x}||_2 \geq \sqrt{\lfloor d/p \rfloor} \cdot (b^{p-1}/2) - 3/4$. ■

By putting together Proposition 1 and Lemma 2 and optimizing the value of p, we obtain the following theorem.

Theorem 3.1 *The price of explainability for the k-centers problem satisfies*

$$\rho(k\text{-}center) \in \begin{cases} \Omega(k^{1-1/d}), & \text{if } d \leq \frac{\ln k}{\ln \ln k} \\ \Omega\left(\sqrt{d} \cdot \frac{k \cdot \sqrt{\ln \ln k}}{\ln^{1.5} k}\right), & \text{otherwise.} \end{cases}$$

Proof. Proposition 1 assures the existence of a k-clustering of cost $3/4$ for instance I_{kc}. Let $\mathcal{C}$ be an explainable clustering for I_{kc} and recall that $b^p = k'$. It follows from the previous lemma that

$$cost(\mathcal{C}) \geq \sqrt{\frac{d}{p}} \cdot \frac{b^{p-1}}{4} - 3/8 = \sqrt{\frac{d}{p}} \cdot \frac{(k')^{\frac{p-1}{p}}}{4} - 3/8.$$

Since $(b+1)^p > k$ we have

$$k' > \frac{k}{(1+1/b)^p} > \frac{k}{\exp(p/b)}.$$

Thus,

$$cost(\mathcal{C}) \geq \sqrt{\frac{d}{p}} \cdot \frac{k^{\frac{p-1}{p}}}{4\exp((p-1)/b)} - 3/8.$$

Now we set $p = d$ if $d \leq \frac{\ln k}{\ln \ln k}$ and $p = \frac{\ln k}{\ln \ln k}$, otherwise. Since $b > k^{1/p} - 1$ we have that $b > \ln k - 1 > p - 1$ for both cases and, hence,

$$cost(\mathcal{C}) \geq \sqrt{\frac{d}{p}} \cdot \frac{k^{\frac{p-1}{p}}}{4} - 3/8.$$

We obtain the desired result by replacing p in the previous equation according to each case. ■

3.1.2 Upper bound

In this section we show that the price of explainability (1-4) for the k-center problem is $O\left(\sqrt{d}k^{\frac{d-1}{d}}\right)$. Note that, for constant d, the upper bound matches the lower bound given by Theorem 3.1.

To obtain the upper bound we analyze the cost of the explainable clustering induced by the decision tree built by the algorithm presented in Algorithm 1. The algorithm has access to the set of representatives of an optimal k-clustering $\mathcal{C}^*$ for $\mathcal{X}$.

Let $\mathcal{X}'$ and S be, respectively, the subset of points in $\mathcal{X}$ and the set of representatives that reach a given node u. As long as it is possible, the algorithm applies an axis-aligned cut that does not separate any point $\mathbf{x} \in \mathcal{X}'$ from its representative. This type of cut is referred as a *clean cut* with respect to $(\mathcal{X}', S)$. When there is no such cut available for u, the algorithm partitions the bounding box of the points in $\mathcal{X}' \cup S$ into $\lfloor |S|^{1/d} \rfloor^d$ axis-aligned boxes of the same dimensions by using a decision tree that emulates a grid. By the bounding box of $\mathcal{X}' \cup S$ we mean the smallest box (hyper-rectangle) with axis-aligned sides that includes the points in $\mathcal{X}' \cup S$.

Algorithm 1 `Ex-kCenter`($\mathcal{X}'$: set of points)

1: $S \leftarrow$ representatives of the points in $\mathcal{X}'$
2: **if** $|S| = 1$ **then**
3: Return $\mathcal{X}'$ and the single representative in S
4: **else**
5: **if** there exists a clean cut w.r.t. $(\mathcal{X}', S)$ **then**
6: $(\mathcal{X}'_L, \mathcal{X}'_R) \leftarrow$ partition induced by the clean cut
7: Create a node u
8: u.`LeftChild` $\leftarrow$`Ex-kCenter`$(\mathcal{X}'_L)$
9: u.`RightChild` $\leftarrow$`Ex-kCenter`$(\mathcal{X}'_R)$
10: Return the tree rooted at u
11: **else**
12: $H \leftarrow$ bounding box for $\mathcal{X}' \cup S$
13: $D^u \leftarrow$ decision tree that partitions H into $\lfloor |S|^{1/d} \rfloor^d$ identical axis-aligned boxes
14: Return D^u as well as an arbitrarily chosen representative for each of its leaves

Theorem 3.2 *The price of explainability for k-centers is* $O\left(\sqrt{d}k^{1-1/d}\right)$.

Proof. We want to prove that, for each leaf ℓ of the tree $\mathcal{D}$ built by `Ex-kCenter`$(\mathcal{X})$, the maximum distance between a point in ℓ and its representative is $OPT\sqrt{d}k^{1-1/d}$, where OPT is the cost of the optimal unrestricted clustering.

If only clean cuts are used in the path from the root of $\mathcal{D}$ to the leaf ℓ, all points that reach ℓ lie in the same cluster of the optimal unrestricted k-clustering $\mathcal{C}^*$, and the maximum distance from a point in ℓ to the single representative in S is upper bounded by OPT. Below we prove the bound for the remaining scenarios.

Let u be the first node in the path from the root to ℓ for which a clean cut is not available. Moreover, let $\mathcal{X}^u$ be the set of points that reach u and let

$s = |S|$, that is, the number of representatives that reach u. In this case the algorithm splits the bounding box for $\mathcal{X}^u \cup S$ into boxes of dimensions

$$\frac{L_1}{\lfloor s^{1/d} \rfloor} \times \cdots \times \frac{L_d}{\lfloor s^{1/d} \rfloor},$$

where L_i is the difference between the maximum and minimum values of the i-th coordinate among points in $\mathcal{X}^u \cup S$.

The maximum distance between a point in ℓ and its representative can be upper bounded by the length of the diagonal of the axis-aligned box corresponding to ℓ. Let $m \in [d]$ be such that $L_m = \max\{L_1, \ldots, L_d\}$. Then, the length of the diagonal is upper bounded by $L_m\sqrt{d}/\lfloor s^{1/d} \rfloor \leq 2L_m\sqrt{d}/s^{1/d}$.

Thus, it suffices to show that $OPT \geq L_m/(2s)$. Let $\mathbf{c}^1, \ldots, \mathbf{c}^s$ be the s representatives that reach node u. In addition, let $\mathbf{x}^j$ be a point in $\mathcal{X}^u$ with representative $\mathbf{c}^j$ and such that $|x^j_m - c^j_m|$ is maximum, among the points in $\mathcal{X}^u$ with representative $\mathbf{c}^j$. Then, we must have

$$\sum_{j=1}^{s} 2|x^j_m - c^j_m| \geq L_m,$$

for otherwise there would be a clean cut (m, θ), with $\theta \in [a, b]$, where $a = \min\{y_m | \mathbf{y} \in \mathcal{X}^u \cup S\}$ and $b = \max\{y_m | \mathbf{y} \in \mathcal{X}^u \cup S\}$. Hence, for some point $\mathbf{x}^j$, $|x^j_m - c^j_m| \geq L_m/(2s)$. Since $OPT \geq |x^j_m - c^j_m|$, we get that $OPT \geq L_m/(2s)$. ■

3.2 Explainable k-medians

In this problem, the goal is to minimize

$$\text{k-medians}(\mathcal{P}) = \sum_{i=1}^{k} \sum_{x \in P_i} ||x - \mu_i||_1, \tag{3-2}$$

or the sum of Manhattan distances (L1 norms) between each point $x \in \mathcal{X}$ and its representative. We achieve an upper bound of $\mathcal{O}(d \log k)$ for the price of explainability (1-4) of this problem, with an algorithm that selects cuts along the dimension of maximum variation for the points they are separating.

Using an optimal unrestricted k-clustering $\mathcal{C}^*$ for $\mathcal{X}$ as a guide for building an explainable clustering, we show that the price of explainability for k-medians is $O(d \log k)$.

For a decision tree $\mathcal{D}$ and a node $u \in \mathcal{D}$, let $diam(u)$ be the d-dimensional vector whose i-th coordinate $diam(u)_i$ is given by the difference between the maximum and the minimum values of coordinate i among the representatives

that reach u. Let t_u be the number of points that reach u and are separated from their representatives by the cut employed in u. Note that a point $\mathbf{x} \in \mathcal{X}$ can only contribute to t_u if both $\mathbf{x}$ and its representative reach u. Finally, let OPT denote the cost of the optimal unrestricted clustering $\mathcal{C}^*$.

The following lemma from (Moshkovitz et al. (2020)), expressed in our notation, will be useful.

Lemma 3 *(Moshkovitz et al. (2020)) Let $\mathcal{C}^*$ be an optimal unrestricted k-clustering for $\mathcal{X}$ and let $\mathcal{D}$ be a decision tree for $\mathcal{X}$ in which each representative of $\mathcal{C}^*$ lies in a distinct leaf. Then, the clustering $\mathcal{C}$ induced by $\mathcal{D}$ satisfies*

$$cost(\mathcal{C}) \leq OPT + \sum_{u \in \mathcal{D}} t_u ||diam(u)||_1. \tag{3-3}$$

3.2.1 Algorithm

In order to obtain a low-cost explainable clustering, we focus on finding a decision tree $\mathcal{D}$ for which the rightmost term of (3-3) is small. This is the approach taken by `IMM` (Moshkovitz et al. (2020)), a greedy strategy that at each node u selects the cut that yields the minimum possible value for t_u, and which we will discuss in Section 4.4.1.

Although we follow the same approach, our strategy for building the tree is significantly different. In order to explain it, we first rewrite the rightmost term of (3-3):

$$\sum_{u \in \mathcal{D}} t_u ||diam(u)||_1 = \sum_{i=1}^{d} \sum_{u \in \mathcal{D}} t_u diam(u)_i.$$

Motivated by Lemma 3 and the above identity, our strategy constructs d decision trees $\mathcal{D}_1, \ldots, \mathcal{D}_d$, where $\mathcal{D}_i$ is built with the aim of minimizing

$$\sum_{u \in \mathcal{D}} t_u diam(u)_i, \tag{3-4}$$

ignoring the impact on the coordinates $j \neq i$.

Next, we build a decision tree $\mathcal{D}$ for $\mathcal{X}$ by picking nodes from these d trees. More precisely, to split a node u of $\mathcal{D}$ the strategy first selects a coordinate $i \in [d]$ for which $diam(u)_i$ is maximum. Next, it applies the cut that is associated with the node in $\mathcal{D}_i$ which is the least common ancestor (LCA) of the set of representatives that reach u.

In Algorithm 2, S' is a subset of the set S of representatives of $\mathcal{C}^*$. Moreover, $\mathcal{X}'$ is a subset of the points in $\mathcal{X}$. The procedure is initially called with $\mathcal{X}' = \mathcal{X}$ and $S' = S$.

Algorithm 2 `BuildTree`$(\mathcal{X}' \cup S')$

1: Create a node u and associate it with $\mathcal{X}' \cup S'$
2: **if** $|S'| = 1$ **then**
3: Return the leaf u
4: **else**
5: Select $i \in [d]$ for which $diam(u)_i$ is maximum.
6: $v \leftarrow$ node in $\mathcal{D}_i$ which is the LCA of the centers in S'
7: Split $\mathcal{X}' \cup S'$ into $\mathcal{X}'_L \cup S'_L$ and $\mathcal{X}'_R \cup S'_R$ using the cut associated with v.
8: u.`LeftChild` $\rightarrow$ `BuildTree`$(\mathcal{X}'_L \cup S'_L)$
9: u.`RightChild` $\rightarrow$ `BuildTree`$(\mathcal{X}'_R \cup S'_R)$
10: Return the decision tree rooted at u

To fully specify the algorithm we need to explain how the decision trees $\mathcal{D}_i$ are built. Let $\mathbf{c}^1, \ldots, \mathbf{c}^k$ be the reference centers sorted by coordinate i, that is, $c_i^j < c_i^{j+1}$ for $j = 1, \ldots, k-1$. Moreover, let (i, θ^j) be the cut that separates the points in $\mathcal{X}$ with the i-th coordinate smaller than or equal to $\theta^j = (c_i^j + c_i^{j+1})/2$ from the remaining ones.

For $1 \leq a \leq b \leq k$, let $\mathcal{F}_{a,b}$ be the family of binary decision trees with $(b-a)$ internal nodes and $b-a+1$ leaves defined as follows:

(i) if $a = b$, then $\mathcal{F}_{a,b}$ has a single tree and this tree contains only one node.

(ii) if $a < b$, then $\mathcal{F}_{a,b}$ consists of all the decision trees $\mathcal{D}'$ with the following structure: the root of $\mathcal{D}'$ is identified by a number $j \in \{a, \ldots, b-1\}$ and associated with the cut (i, θ^j); one child of the root of $\mathcal{D}'$ is a tree in the family $\mathcal{F}_{a,j}$ while the other is a tree in $\mathcal{F}_{j+1,b}$.

Let $\mathcal{F}_{a,b}$ be the family of binary search trees for the numbers in the set $\{a, \ldots, b-1\}$, and let T_j be the number of points in $\mathcal{X}$ that are separated from their centers by cut (i, θ^j). For every tree $\mathcal{D}' \in \mathcal{F}_{a,b}$ we define $UB_i(\mathcal{D}')$ as

$$UB_i(\mathcal{D}') = \sum_{j=a}^{b-1} T_j \cdot diam(j)_i,$$

where $diam(j)$ is the diameter of the node identified by j in $\mathcal{D}'$.

The tree $\mathcal{D}_i$ is, then, defined as

$$\mathcal{D}_i = \text{argmin}\{UB_i(\mathcal{D}') \mid \mathcal{D}' \in \mathcal{F}_{1,k}\}.$$

The motivation for minimizing $UB_i()$ is that, for every tree $\mathcal{D}' \in \mathcal{F}_{1,k}$,

$UB_i()$ is an upper bound on (3-4), that is,

$$\sum_{u \in \mathcal{D}'} t_u diam(u)_i \leq \sum_{j=1}^{k-1} T_j \cdot diam(j)_i = UB_i(\mathcal{D}').$$

To see that, let j be the integer identified with the node $u \in \mathcal{D}'$. By definition $diam(u)_i = diam(j)_i$. Moreover, we have $t_u \leq T_j$ because t_u only accounts the points that are separated from their representatives among those that reach u, while T_j accounts all the points in $\mathcal{X}$ regardless of whether they reach u or not.

To see how to construct $\mathcal{D}_i$ efficiently, let

$$OPT_{a,b} = \begin{cases} \min\{UB_i(\mathcal{D}') \mid \mathcal{D}' \in \mathcal{F}_{a,b}\} & \text{if } a < b \\ 0 & \text{if } a = b. \end{cases}$$

Hence, $UB_i(\mathcal{D}_i) = OPT_{1,k}$. The following relation holds for all $a < b$:

$$OPT_{a,b} = \min_{a \leq j \leq b-1} \left\{ T_j(c_i^b - c_i^a) + OPT_{a,j} + OPT_{j+1,b} \right\}. \tag{3-5}$$

Thus, given a set of k representatives and the values T_j's, $\mathcal{D}_i$ can be computed in $O(k^3)$ time by solving equation (3-5), for $a = 1$ and $b = k$, via standard dynamic programming techniques.

3.2.2 Analysis

We prove that the cost of the clustering induced by $\mathcal{D}$ is $O(d \log k) \cdot OPT$. To reach this goal, we first show the following bound for the diameter loss of $\mathcal{D}_i$, relying on the fact that $\mathcal{D}_i$ can be seen as a binary search tree with non-uniform probing costs.

Lemma 4 *The tree $\mathcal{D}_i$ satisfies*

$$UB_i(\mathcal{D}_i) \leq 2 \log k \left(\sum_{j=1}^{k-1} (c_i^{j+1} - c_i^j) T_j \right). \tag{3-6}$$

Proof.

Let $\mathcal{D}'$ be a tree in $\mathcal{F}_{1,k}$. By construction, the set of centers that reach the node in $\mathcal{D}'$ identified by j is a contiguous subsequence of $\mathbf{c}^1, \ldots, \mathbf{c}^k$. Let $r(j)$ and $s(j)$ be, respectively, the first and the last indexes of the centers of this subsequence. Thus,

$$UB_i(\mathcal{D}') = \sum_{j=1}^{k-1} T_j \cdot diam(j)_i = \sum_{j=1}^{k-1} T_j \sum_{\ell=r(j)}^{s(j)-1} (c_i^{\ell+1} - c_i^{\ell}). \tag{3-7}$$

We can show that the right-hand side of the above equation satisfies

$$\sum_{j=1}^{k-1} T_j \sum_{\ell=r(j)}^{s(j)-1} (c_i^{\ell+1} - c_i^{\ell}) = \sum_{\ell=1}^{k-1} (c_i^{\ell+1} - c_i^{\ell}) \cdot \sum_{j \in An(\ell, \mathcal{D}')} T_j, \tag{3-8}$$

where $An(\ell, \mathcal{D}')$ is the set of nodes that are ancestors (including ℓ) of the node identified by ℓ in $\mathcal{D}'$.

To see that, fix $j, \ell \in [k-1]$. The term $T_j(c_i^{\ell+1} - c_i^{\ell})$ contributes the left-hand side of (3-8) if the centers $\mathbf{c}^{\ell}$ and $\mathbf{c}^{\ell+1}$ reach the node j in $\mathcal{D}'$. This happens if and only if j is an ancestor of the node identified by ℓ in $\mathcal{D}'$.

Now, we use Theorem 4.5 from (Charikar et al. (2002)). It states that for any vector $(p_1, \ldots, p_k)$ of k non-negative real numbers there exists a binary search tree B having k nodes, with each of them associated with a number in $[k]$, that satisfies

$$\sum_{j \in An(\ell, B)} p_j \leq (\log k + o(\log k)) p_\ell \leq 2 \log k \cdot p_\ell$$

for every node ℓ of B.

Let $\mathcal{D}_c$ be a tree obtained via the result of (Charikar et al. (2002)) for the vector $(T_1, \ldots, T_{k-1})$. It satisfies

$$\sum_{j \in An(\ell, \mathcal{D}_c)} T_j \leq 2 \log(k-1) \cdot T_\ell.$$

By using this inequality, (3-7), and (3-8), we get that

$$UB_i(\mathcal{D}_c) = \sum_{\ell=1}^{k-1} (c_i^{\ell+1} - c_i^{\ell}) \sum_{j \in An(\ell, \mathcal{D}_c)} T_j \leq 2 \log k \sum_{\ell=1}^{k-1} (c_i^{\ell+1} - c_i^{\ell}) T_\ell.$$

The result follows because the minimality of $\mathcal{D}_i$ guarantees that $UB_i(\mathcal{D}_i) \leq UB_i(\mathcal{D}_c)$. ■

Lemma 5 *Let $OPT_i = \sum_{\mathbf{x} \in \mathcal{X}} |x_i - c(\mathbf{x})_i|$ be the contribution of the coordinate i for the cost of an optimal unrestricted clustering $\mathcal{C}^*$, where $c(\mathbf{x})$ is the representative of $\mathbf{x}$. Then,*

$$OPT_i = \sum_{\mathbf{x} \in \mathcal{X}} |x_i - c(\mathbf{x})_i| \geq \sum_{j=1}^{k-1} \frac{(c_i^{j+1} - c_i^{j}) T_j}{2}, \tag{3-9}$$

where $c(\mathbf{x})$ is the representative of $\mathbf{x}$.

Proof. The proof consists of projecting the points of $\mathcal{X}$ and the representatives onto the axis i and then counting the number of times the interval $[c_i^j, c_i^{j+1}]$ appears in the segments that connect points in $\mathcal{X}$ to their representatives. This is exactly the same line of reasoning employed to prove Lemma 6 from the supplementary version of (Moshkovitz et al. (2020)).

Let $\mathbf{c}^1, \ldots, \mathbf{c}^k$ be the representatives sorted by increasing order of coordinate i. Recall that $\theta^j = (c_i^j + c_i^{j+1})/2$. For every $\mathbf{x} \in \mathcal{X}$, let $Cut(\mathbf{x}) = \{j | (i, \theta^j)$ separates $\mathbf{x}$ from $c(\mathbf{x})\}$.

Fix $\mathbf{x} \in \mathcal{X}$. If $j \in Cut(\mathbf{x})$ then either $[c_i^j, \theta^j]$ or $[\theta^j, c_i^{j+1}]$ is included in the real interval with endpoints x_i and $c(x)_i$. Thus, we have that

$$|x_i - c(x)_i| \geq \sum_{j \in Cut(\mathbf{x})} \frac{c_i^{j+1} - c_i^j}{2}.$$

By adding the above inequality for all $\mathbf{x} \in \mathcal{X}$ we conclude that the number of times that $(c_i^{j+1} - c_i^j)/2$ contributes to the right-hand side, for every $j \in [k-1]$, is exactly the number of times that (i, θ^j) separates a point $\mathbf{x}' \in \mathcal{X}$ from its representative $c(\mathbf{x}')$. This number is exactly T_j. ■

From (3-7) and (3-9), we obtain

$$UB_i(\mathcal{D}_i) \leq 4 \log k \cdot OPT_i. \tag{3-10}$$

Finally, we prove that a factor of d is incurred when we build the tree $\mathcal{D}$ from the nodes of the trees $\mathcal{D}_1, \ldots, \mathcal{D}_d$:

Lemma 6 *Let $\mathcal{D}$ be the decision tree built by Algorithm 2. Then,*

$$\sum_{v \in \mathcal{D}} t_v ||diam(v)||_1 \leq d \sum_{i=1}^{d} UB_i(\mathcal{D}_i). \tag{3-11}$$

Proof. For a node $j \in \mathcal{D}_i$, let $S_{i,j}$ be the (possibly empty) set of nodes in the tree $\mathcal{D}$ that correspond to j, that is, the nodes that use the cut associated with the node j from $\mathcal{D}_i$. We have

$$\sum_{v \in \mathcal{D}} t_v ||diam(v)||_1 = \sum_{i=1}^{d} \sum_{j \in \mathcal{D}_i} \sum_{u \in S_{i,j}} t_u ||diam(u)||_1. \tag{3-12}$$

Moreover, we have that

$$\sum_{u \in S_{i,j}} t_u ||diam(u)||_1 \leq \sum_{u \in S_{i,j}} t_u \cdot d \cdot diam(u)_i \leq \tag{3-13}$$

$$\sum_{u \in S_{i,j}} d \cdot t_u \cdot \max_{u \in S_{i,j}} \{diam(u)_i\} \leq \sum_{u \in S_{i,j}} d \cdot t_u \cdot diam(j)_i, \tag{3-14}$$

where the first inequality in (3-13) holds because i is the coordinate for which the diameter of u is maximum and the inequality (3-14) holds because the set of centers in u is a subset of the set of centers that reach the node identified by j in $\mathcal{D}_i$.

Claim 1 *For a node $u \in S_{i,j}$, let $\mathcal{X}_u \subseteq \mathcal{X}$ be the set of points that reach u in $\mathcal{D}$. Then, $\mathcal{X}_u \cap \mathcal{X}_{u'} = \emptyset$ for every $u, u' \in S_{i,j}$, with $u \neq u'$.*

Proof. Let w be the least common ancestor of u' and u in $\mathcal{D}$. If $w \notin \{u, u'\}$ then the cut associated with w splits $\mathcal{X}_w$ into two disjoint regions, one of them containing $\mathcal{X}_u$ and the other containing $\mathcal{X}_{u'}$ so that $\mathcal{X}_u$ and $\mathcal{X}_{u'}$ are disjoint.

If $w \in \{u, u'\}$ let us assume without loss of generality that $w = u$. In this case, the cut (i, θ^j), associated with u, splits $\mathcal{X}_u$ into two regions, one of them containing all the representatives that reach u'. These centers are contained in the set of representatives of one of the children of j in $\mathcal{D}_i$ and, hence, the LCA in $\mathcal{D}_i$ of the set of centers that reach u' is not j, that is, $u' \notin S_{i,j}$. This contradiction shows that this case cannot occur. ■

From the previous claim we get that

$$\sum_{u \in S_{i,j}} t_u \leq T_j.$$

It follows from (3-13)-(3-14) and the above inequality that

$$\sum_{u \in S_{i,j}} t_u ||diam(u)||_1 \leq d \cdot T_j \cdot diam(j)_i.$$

Hence, it follows from (3-12) that

$$\sum_{v \in \mathcal{D}} t_v ||diam(v)||_1 \leq \sum_{i=1}^{d} \sum_{j \in \mathcal{D}_i} d \cdot T_j \cdot diam(j)_i = d \sum_{i=1}^{d} UB_i(\mathcal{D}_i).$$

■

From (3-10), (3-11), and the identity $OPT = \sum_{i=1}^{d} OPT_i$, we obtain

$$\sum_{v \in \mathcal{D}} t_v ||diam(v)||_1 \leq 4d \log k \cdot OPT.$$

This, together with Lemma 3, allows us to establish the main theorem of this section.

Theorem 3.3 *The price of explainability* (1-4) *for* `k-medians` (3-2) *is* $O(d \log k)$.

3.3 Explainable k-means

In this problem, the goal is to minimize (1-1), or the sum of the squared Euclidean distances (L2 norms) between each point $x \in \mathcal{X}$ and its

representatives. Using an algorithm similar to the one used for k-medians cost, we reach an upper limit of $\mathcal{O}(dk \log k)$ for the price of explainability (1-4).

3.3.1 Bounds for low dimensions

The result we obtained for the k-medians problem can be extended to the k-means problem:

Theorem 3.4 *The price of explainability for k-means is $O(dk \log k)$.*

Proof. From an algorithmic perspective, in order to establish the theorem, we only need to replace the definition of $UB_i(\mathcal{D}')$ for a tree $\mathcal{D}'$ in $\mathcal{F}_{a,b}$ with

$$UB_i'(\mathcal{D}') = \sum_{j=a}^{b-1} T_j \cdot (diam(j)_i)^2.$$

Note that the only difference is the replacement of $diam(j)_i$ with $(diam(j)_i)^2$. As a consequence, for the k-means problem, the tree $\mathcal{D}_i$ is defined as the tree $\mathcal{D}'$ in $\mathcal{F}_{1,k}$ for which $UB_i'(\mathcal{D}')$ is minimum. It can also be constructed via dynamic programming.

Theorem 3.4 can be proved by using arguments similar to those employed to bound the price of explainability for k-medians. The following inequalities are, respectively, counterparts of the inequalities (3-3), (3-6), (3-9) and (3-11):

$$cost(\mathcal{C}) \leq OPT + \sum_{v \in \mathcal{D}} t_v ||diam(v)||_2^2, \tag{3-15}$$

$$UB_i'(\mathcal{D}_i) \leq 2k \log k \left(\sum_{j=1}^{k-1} (c_i^{j+1} - c_i^j)^2 \cdot T_j \right), \tag{3-16}$$

$$\left(\sum_{j=1}^{k-1} (c_i^{j+1} - c_i^j)^2 \cdot T_j \right) /2 \leq OPT_i, \tag{3-17}$$

$$\sum_{v \in \mathcal{D}} t_v ||diam(v)||_2^2 \leq d \sum_{i=1}^{d} UB_i'(\mathcal{D}_i). \tag{3-18}$$

From the three last inequalities and the identity $OPT = \sum_{i=1}^{d} OPT_i$, we obtain

$$\sum_{v \in \mathcal{D}} t_v ||diam(v)||_2^2 \leq 4dk \log k \cdot OPT.$$

This together with the inequality (3-15) allows us to establish Theorem 3.4.

Inequality (3-15) is proved in (Moshkovitz et al. (2020)). The validity of inequalities (3-17) and (3-18) can be established by using exactly the same arguments employed to prove their counterparts. More specifically, the proof of Lemma 5 can be used for the former while the proof of Lemma 6 can be used for the latter.

Inequality (3-16) incurs an extra factor of k with respect to its counterpart. In order to prove this inequality, we apply the arguments of the proof of Lemma 4. The only required adaptation consists of replacing Equation (3-7) with the inequality

$$UB_i'(\mathcal{D}_i) \leq k \sum_{j=1}^{k-1} T_j \cdot \sum_{\ell=r(j)}^{s(j)-1} (c_i^{\ell+1} - c_i^{\ell})^2. \tag{3-19}$$

Inequality (3-19) holds because

$$UB_i'(\mathcal{D}_i) = \sum_{j=1}^{k-1} T_j \cdot (c_i^{s(j)} - c_i^{r(j)})^2,$$

and Jensen's inequality assures that

$$(c_i^{s(j)} - c_i^{r(j)})^2 \leq k \sum_{\ell=r(j)}^{s(j)-1} (c_i^{\ell+1} - c_i^{\ell})^2.$$

■

3.4 Explainable minimum spacing

In this problem, the goal is to maximize (1-2), i.e., the minimum distance between two points belonging to different clusters. Using a simple instance and algorithm, we prove that the price of explainability for this problem (1-5) is $\Theta(n-k)$.

3.4.1 Lower bound

The following simple construction shows that the price of explainability is $\Omega(n-k)$.

Lemma 7 *The price of explainability for the minimum spacing clustering problem* (1-5) *is* $\Omega(n-k)$.

Proof. Let $C_1 = \{(0,i)|0 \leq i \leq p\} \cup \{(i,0)|0 \leq i \leq p\}$. Moreover, for $i = 2, \ldots, k$, let $C_i = \{(i-1)(p-1), (p-1)\}$. The dataset $\mathcal{X}$ for our instance is given by $C_1 \cup \ldots \cup C_k$.

The unrestricted k-clustering $(C_1, \ldots, C_k)$ has spacing $p - 1 = (n - k)/2 - 1$. On the other hand, every explainable k-clustering has spacing 1. To see that, note that we cannot have all the points of $C_1 \cup C_2$ in the same cluster, for otherwise we would have at most $k-1$ clusters. Thus, we need to separate

at least 2 points from $C_1 \cup C_2$ and the only way to accomplish that, via axis-aligned cuts, forces the separation of 2 points in C_1 that are at distance 1 from each other. Thus, the spacing will be 1. ■

3.4.2 Upper bound

We present an algorithm that always obtains an explainable clustering with spacing $\mathcal{O}(n-k) \cdot OPT$, where OPT is the spacing of the optimal unrestricted clustering. That, together with the previous lemma, proves that the price of explainability for the minimum spacing problem is $\Theta(n-k)$.

Algorithm 3 receives an optimal k-clustering $\mathcal{C}^*$ as input and uses it as a guide to transform an initial single cluster containing all points of $\mathcal{X}$ into an explainable k-clustering. The existence of cluster C at line (*) follows from a simple pigeonhole argument. The motivation for this choice is that C has two points at distance at least OPT, which is used to show the existence of a cut with a large enough margin.

Algorithm 3 `Ex-SingleLink`$(\mathcal{X})$

1: $\mathcal{C}^* \leftarrow$ optimal unrestricted k-clustering for points in $\mathcal{X}$.
2: $\mathcal{C} \leftarrow$ single cluster containing all points of $\mathcal{X}$
3: **for** $i = 1, \ldots, k-1$ **do**
4: Select a cluster $C \in \mathcal{C}$ that contains two points that lie in different clusters in $\mathcal{C}^*$. (*)
5: Split C using an axis-aligned cut that yields a 2-clustering (C', C'') with maximum possible spacing.
6: Remove C from $\mathcal{C}$ and update $\mathcal{C}$ to $\mathcal{C} \cup \{C', C''\}$

Lemma 8 *Given a set of points $\mathcal{X}$,* `Ex-SingleLink`*$(\mathcal{X})$ obtains a k-clustering $\mathcal{C}$ with spacing at least $OPT/(n-k)$, where OPT is the spacing of an optimal unrestricted clustering.*

Proof. First, we observe that it is always possible to properly execute line (*) of `Ex-SingleLink`: if we pick k points covering all the k clusters of $\mathcal{C}^*$, by the pigeonhole principle, two of them will lie in the same group in $\mathcal{C}$ since $\mathcal{C}$ has less than k groups when line (*) is executed.

To establish the result it suffices to prove that there is always an axis-aligned cut that splits the selected cluster C into two clusters with spacing at least $OPT/(n-k)$.

Let $\mathbf{p}$ and $\mathbf{q}$ be two points in C that lie in distinct clusters in $\mathcal{C}^*$ and let $G = (V, E)$ be a graph, where V is the set of points in C and E connects points

in V with distance smaller than $OPT/(n-k)$. Moreover, let $F = (T_1, \ldots, T_\ell)$ be a forest that is obtained by running Kruskal's MST algorithm on G.

Claim 2 *Points in C that belong to distinct clusters of $\mathcal{C}^*$ must also belong to different trees in forest F.*

Proof. For the sake of contradiction, assume that the claim does not hold. In this case, there would be a path from $\mathbf{p}$ to $\mathbf{q}$ in F and this path would have an edge joining two points that belong to different clusters in $\mathcal{C}^*$, which cannot occur since their distance is at least $OPT > OPT/(n-k)$. ■

The previous claim implies that $\ell \geq 2$ since $\mathbf{p}$ and $\mathbf{q}$ belong to different clusters. We say that an axis-aligned cut is *good* with respect to a cluster C if it satisfies the following properties: (i) it separates the points in C into two non-empty clusters and (ii) it does not separate points that lie in the same tree of F. If a good cut exists, then we can use it to split C into two clusters with spacing at least $OPT/(n-k)$ since, by construction, points in different trees have distance at least $OPT/(n-k)$.

For the sake of contradiction, assume that such a cut does not exist. For each $j \in [d]$ let $I_j^{\mathbf{pq}}$ be the real interval that starts in $\min\{p_j, q_j\}$ and ends in $\max\{p_j, q_j\}$, that is, $I_j^{\mathbf{pq}} = [\min\{p_j, q_j\}, \max\{p_j, q_j\}]$.

Moreover, for each tree T in F, let I_j^T be the interval that starts at $\min\{x_j | \mathbf{x} \text{ is a node in } T\}$ and ends at $\max\{x_j | \mathbf{x} \text{ is a node in } T\}$. Finally, for each edge $e = uv$ in F and each $j \in [d]$, let I_j^e be the real interval that starts at $\min\{u_j, v_j\}$ and ends at $\max\{u_j, v_j\}$. For a real interval I, let $len(I)$ be its length.

Since there are no good cuts, for $j = 1, \ldots, d$, we have

$$\sum_{T \in F} len(I_j^T) \geq len(I_j^{\mathbf{pq}}).$$

From the triangle inequality we obtain

$$\sum_{T \in F} \sum_{e \in T} len(I_j^e) \geq \sum_{T \in F} len(I_j^T).$$

From the two previous inequalities we get

$$\sum_{e \in F} len(I_j^e) = \sum_{T \in F} \sum_{e \in T} len(I_j^e) \geq len(I_j^{\mathbf{pq}}).$$

From Jensen's inequality we obtain

$$\sum_{e \in F} len(I_j^e)^2 \geq \frac{(len(I_j^{\mathbf{pq}}))^2}{f},$$

where f is the number of edges in F. By adding the above inequality for all $j \in [d]$ we get

$$\sum_{e \in F} ||e||_2^2 \geq \frac{1}{f}||\mathbf{p} - \mathbf{q}||_2^2 \geq \frac{OPT^2}{f},$$

where $||e||_2$ is the distance between the two endpoints of edge e.

The last inequality implies $||e||_2 \geq OPT/f$ for some edge e. Thus, to obtain a contradiction, it suffices to show that $f \leq n-k$, since we cannot have edges in F with distance $\geq OPT/(n-k)$.

To see that $f \leq n-k$, let k' be the number of clusters in $\mathcal{C}$ that are singletons and let S' be the set of points in these clusters. Moreover, let $S \subseteq \mathcal{X} - S'$ be a set of $k-k'$ points with each of them belonging to a different cluster in $\mathcal{C}^*$. Note that cluster C is not a singleton since $\mathbf{p}, \mathbf{q} \in C$. Since both C and S are subsets of $\mathcal{X} - S'$ we have $|C \cup S| = |C| + |S| - |C \cap S| \leq n - k'$ so that $|C| - |C \cap S| \leq n - k$. It follows from Claim 2 that the number of trees in F is at least $|C \cap S|$ and, as a result, its number of edges f satisfies $f \leq |C| - |C \cap S| - 1 < n - k$ edges. ■

From Lemmas 7 and 8 we can establish the main result of this section.

Theorem 3.5 *The price of explainability for the minimum-spacing problem is* $\Theta(n-k)$.

4
Practical algorithms for explainable k-means clustering

This chapter presents three practical algorithms for Problem 2 with the popular k-means cost function (1-1).

In Section 4.1 we present `ExGreedy` (Laber & Murtinho (2021)), a simple greedy algorithm that builds explainable partitions from unrestricted ones by greedily selecting the cut that minimizes the k-means cost function (1-1) and separates at least one representative and one data point from the rest of the remaining data set.

In Section 4.2 we add some nuance to the discussion on explainability, considering how some decision trees are more easily understandable than others, and introduce `ExShallow` (Laber et al. (2023)), an adaptation of `ExGreedy` with a penalty term that allows a trade-off between the minimization of the cost function and the explainability of the partition.

In Section 4.3 we present `ExBisection` (Laber et al. (2023)), an algorithm that builds an explainable partition without using an unrestricted one as a starting point.

In Section 4.4 we briefly discuss other algorithms for the explainable k-means clustering problem, and in Section 4.5 we present the results of experiments performed between on these and our algorithms over 16 datasets.

4.1
`ExGreedy`: Greedily towards an explainable k-partition

`ExGreedy` is a simple greedy algorithm for building explainable partitions for the k-means problem. It starts with the set S of representatives of an unrestricted k-clustering $\mathcal{C}^{ini}$ for the dataset $\mathcal{X}$ and then builds a decision tree $\mathcal{D}$ with k leaves, where each of them includes exactly one representative from S.

Let u be a node of the decision tree and let $\mathcal{X}^u$ and $\mathcal{S}^u$ be, respectively, the set of points and the set of representatives of $\mathcal{C}^{ini}$ that reach u. We define the cost of a partition (L, R) of the points in $\mathcal{X}^u \cup \mathcal{S}^u$ as

$$\texttt{Cost}(\gamma, \mathcal{X}', \mathcal{S}') := \left(\sum_{\mathbf{x} \in \mathcal{X}'_{\mathcal{L}}} \min_{\mathbf{c} \in \mathcal{S}'_{\mathcal{L}}} ||\mathbf{x} - \mathbf{c}||_2^2 + \sum_{\mathbf{x} \in \mathcal{X}'_{\mathcal{R}}} \min_{\mathbf{c} \in \mathcal{S}'_{\mathcal{R}}} ||\mathbf{x} - \mathbf{c}||_2^2 \right). \tag{4-1}$$

To split a node u that is reached by more than one representative, `ExGreedy` selects the axis-aligned cut that induces a k-partition with minimum cost. Algorithm 4 describes the procedure in pseudocode.

Algorithm 4 `ExGreedy`$(\mathcal{X}', \mathcal{S}')$

$\mathcal{X}'$: set of points; $\mathcal{S}'$: set of representatives

1: **if** $|S'| = 1$ **then**
2: Return $\mathcal{X}'$ and the single representative in S'
3: **else**
4: $\mathcal{C} \leftarrow$ set of non-equivalent cuts w.r.t. $\mathcal{X}' \cup \mathcal{S}'$ that separate at least two centers in $\mathcal{S}'$
5: $\gamma^* \leftarrow \arg\min_{\gamma \in \mathcal{C}}\{\texttt{Cost}(\gamma, \mathcal{X}', \mathcal{S}')\}$
6: $(\mathcal{X}^*_L, \mathcal{X}^*_R) \leftarrow$ partition of $\mathcal{X}'$ induced by γ^*
7: $(\mathcal{S}^*_L, \mathcal{S}^*_R) \leftarrow$ partition of $\mathcal{S}'$ induced by γ^*
8: Create a node u
9: $u.\texttt{LeftChild} \leftarrow \texttt{ExGreedy}(\mathcal{X}^*_L, \mathcal{S}^*_L)$
10: $u.\texttt{RightChild} \leftarrow \texttt{ExGreedy}(\mathcal{X}^*_R, \mathcal{S}^*_R)$
11: Return the tree rooted at u

Note that `ExGreedy` will find $k-1$ cuts, and that after each cut (except for the last one) the k-partition used when calculating (4-1) is not necessarily explainable. At any point during the procedure, the only clusters guaranteed to be explainable are those associated to a leaf of the tree – i.e., those generated when line 1 is true. Once $k-1$ cuts are selected, each cluster is associated to a leaf, and partition explainability is guaranteed. Contrast this with `ExKMC`, in which the i-th cut is chosen by evaluating the cost of an explainable $(i+1)$-partition (see Secion 4.4.2 for more details). Also note that the representatives remain the same after a cut is performed.

4.1.1 An efficient implementation

`ExGreedy` can be implemented in $O(ndkH + nd\log n)$ time, where H is the depth of the resulting decision tree. Note that $H \leq k$ and in many relevant applications k is small. The time complexity corresponds to H iterations of Lloyd's k-means algorithm.

To achieve this time complexity, in the preprocessing phase, `ExGreedy` builds the following data structures:

- a list $\texttt{SL}_\texttt{i}$, for each $i \in [d]$, containing the points in $\mathcal{X} \cup S$ sorted by coordinate i;
- a list $\texttt{M}_\mathbf{x}$ of size k, for each $\mathbf{x} \in \mathcal{X}$, that stores the k centers sorted by increasing order of their distances to $\mathbf{x}$.

The lists $\texttt{SL}_\texttt{i}$ can be built in $O(dn \log n)$ time and the lists $\texttt{M}_\mathbf{x}$ in $O(nk \log k)$ time.

To decide how to split the root, `ExGreedy` finds the partition with minimum cost for each coordinate $i \in [d]$ and then selects the one with minimum cost among them.

Fix $i \in [d]$. The algorithm scans the list $\texttt{SL}_\texttt{i}$ from left to the right and evaluates the cost of $n-k+1$ partitions where the j-th one, namely (L_j, R_j), separates the first j points in $\texttt{SL}_\texttt{i}$ from the remaining ones. During the scan the algorithm makes use of two vectors of size n, $\texttt{V}_L$ and $\texttt{V}_R$. Right after evaluating (L_j, R_j), $\texttt{V}_L$ (resp. $\texttt{V}_R$) stores, for each $\mathbf{x}$ that lies at L_j (resp. R_j), the center that is closest to $\mathbf{x}$ among those that also lie in L_j (resp. R_j). The only difference is that $\texttt{V}_L[\mathbf{x}]$ stores the center directly while $\texttt{V}_R[\mathbf{x}]$ stores the position of the center in $\texttt{M}_\mathbf{x}$.

Let us consider the moment in which the algorithm has just calculated the cost $\texttt{Cost}_\texttt{j}$ of the jth partition (L_j, R_j). To obtain $\texttt{Cost}_{\texttt{j+1}}$ and update $\texttt{V}_L$ and $\texttt{V}_R$, the algorithm first set $\texttt{Cost}_{\texttt{j+1}} = \texttt{Cost}_\texttt{j}$ and then proceeds according to the following cases:

Case 1. The $(j+1)$-th point in $\texttt{SL}_\texttt{i}$ corresponds to a point $\mathbf{x}$ in $\mathcal{X}$. Then, the algorithm evaluates $\texttt{Cost}_{\texttt{j+1}}$ in $O(k)$ time as follows:

i it obtains the center $\mathbf{c}_R$ in R_j that is closest to $\mathbf{x}$. This is done in $O(1)$ time since $\texttt{V}_R[\mathbf{x}]$ points to this center;

ii By scanning $\texttt{M}_\mathbf{x}$ it obtains the center $\mathbf{c}_L$ in L_{j+1} that is closest to $\mathbf{x}$ and then updates $\texttt{V}_L[\mathbf{x}]$ to $\mathbf{c}_L$. This requires $O(k)$ time

iii it updates $\texttt{Cost}_{\texttt{j+1}}$ to $\texttt{Cost}_\texttt{j} + ||\mathbf{x} - \mathbf{c}_L||_2^2 - ||\mathbf{x} - \mathbf{c}_R||_2^2$.

Case 2. The $(j+1)$-th point in $\texttt{SL}_\texttt{i}$ corresponds to a reference center $\mathbf{c}$ in S. Then, the algorithm evaluates $\texttt{Cost}_{\texttt{j+1}}$ in $O(n)$ amortized time as follows:

i for each point $\mathbf{x}$ in L_j, `ExGreedy` compares $||\mathbf{c} - \mathbf{x}||_2^2$ with $||\texttt{V}_L[x] - \mathbf{x}||_2^2$. If the former is smaller, $\texttt{Cost}_{\texttt{j+1}}$ is updated to $\texttt{Cost}_{\texttt{j+1}} + ||\mathbf{x} - \mathbf{c}||_2^2 - ||\mathbf{x} - \texttt{V}_L[x]||_2^2$ and $\texttt{V}_L[\mathbf{x}]$ is updated to $\mathbf{c}$. This requires $O(n)$ time.

ii for each point $\mathbf{x}$ in R_j it verifies whether $V_R[\mathbf{x}]$ points to $\mathbf{c}$. In the negative case, nothing is done. In the positive case, it scans $\texttt{M}_\mathbf{x}$, starting from $V_R[\mathbf{x}]$ towards its end, until it finds a center $\mathbf{c}'$ that lies in R_j. Then it updates $\texttt{Cost}_{\texttt{j+1}}$ to $\texttt{Cost}_\texttt{j} - ||\mathbf{x} - \mathbf{c}||_2^2 + ||\mathbf{x} - \mathbf{c}'||_2^2$. This operation requires $O(n)$ amortized time since the total cost spent on these scans, when we take into account moving the k centers, is $O(nk)$.

The algorithm applies the cut with minimum cost and then recurses on each child of the root. To process a child u of the root, the implementation updates the data structures $\texttt{SL}_\texttt{i}$ and $\texttt{M}_\texttt{x}$ to only comprise the points and reference centers that reach u. Each list $\texttt{SL}_\texttt{i}$ can be updated in $O(n)$ time by removing the points and the reference centers that do not reach u. Similarly, each list $\texttt{M}_\texttt{x}$ can be updated in $O(k)$ time by removing the points and the reference centers that do not reach u.

4.2 `ExShallow`: Depth matters

When we treat partitions induced by decision trees as explainable, we rely on the fact that decision trees are widely considered to be explainable models by machine learning standards. However, the explainability of a decision tree greatly depends on the depths of its leaves, as empirically demonstrated by (Piltaver et al. (2016)) in a study on how tree structure parameters (number of leaves, branching factor, tree depth) influence tree interpretability. The conclusion, based on empirical data from a survey with 98 questions answered by 69 respondents, is that *question depth* (the depth of the deepest leaf that is required when answering a question about a classification tree) turns out to be the most important parameter; explaining leaves that are far from the root involves many tests, which makes it harder to grasp the model's logic.

`ExShallow` incorporates this fact to the construction of explainable partitions based on decision trees by adapting the cost function used by `ExGreedy`, incorporating a penalty term to discourage the selection of cuts that lead to unbalanced (and therefore deeper) trees. To quantify this cost, two measures are introduced: the Weighted Average Depth (`WAD`) and the Weighted Average Explanation Size (`WAES`).

4.2.1 Measures of explainability for decision tree-induced partitions

Let $\mathcal{P} = (C_1, \ldots, C_k)$ be a k-partition induced by a binary decision tree $\mathcal{D}$ with k leaves, where the cluster C_i is associated with the leaf i. Furthermore, associate a condition to each edge of the tree: the left edge leaving a node v is associated with the condition $x_{i_v} \leq \theta_v$ and the right one with the condition $x_{i_v} > \theta_v$.

The explanation of a cluster C in a decision tree $\mathcal{D}$ is given by the logical AND of the conditions associated with the edges in the path from the root of $\mathcal{D}$ to the leaf associated with C. We say that a condition is *redundant* with respect to cluster C if its removal does not change the explanation for C. As an

example, if the explanation of cluster C is $x_1 > 30$ AND $x_2 \leq 20$ AND $x_1 > 70$, then the condition $x_1 > 30$ is redundant.

Then, we can define the *weighted average depth* (`WAD`) of a decision tree $\mathcal{D}$ as

$$\texttt{WAD}(\mathcal{D}) = \frac{\sum_{i=1}^{k} |C_i|\ell_i}{n} \tag{4-2}$$

and its *weighted average explanation size* (`WAES`) as

$$\texttt{WAES}(\mathcal{D}) = \frac{\sum_{i=1}^{k} |C_i|\ell_i^{nr}}{n}, \tag{4-3}$$

where l_i and ℓ_i^{nr} are, respectively, the number of conditions and non-redundant conditions (w.r.t. C_i) in the path from the root to leaf i.

`WAD` is a very natural metric; its relevance is advocated in (Piltaver et al. (2016)), and it is used in (McSherry (2002)) to evaluate decision-tree algorithms for classification. To the best of our knowledge, `WAES` was introduced by our work in (Laber et al. (2023)); it is related to the research from (Feitosa et al. (2022)), in which bounds for both the average and the maximum explanation size (defined as the maximum number of different attributes used in a path from the root to a leaf of the tree) are presented and discussed.

In terms of explainability, a decision tree is a single structure that allows us to visualize explanations for all clusters (some of them potentially having redundant conditions), and `WAD` gives the average length (weighted by the cluster's sizes) of these explanations. For each specific cluster, however, we may derive more compact explanations by removing redundant conditions, and `WAES` measures the average size of these explanations, again weighted by the cluster's sizes.

To illustrate the impact of considering these metrics when building explainable partitions, we present in Figures 4.1 and 4.2 two decision trees that partition the `Avila` dataset (De Stefano et al. (2018)) into 12 clusters. Both trees induce the same partition; however, the tree from Figure 4.1, produced by `ExGreedy`, has `WAES` ≈ 5.4, while the one from Figure 4.2, produced by `ExShallow`, has `WAES` ≈ 3.7 – i.e., a "gain" of almost 2 conditions on average. For `WAD`, the gain is even larger, of over 2 conditions (≈ 6.2 vs. ≈ 3.8). (Note that both algorithms do not, as a rule, return the same partition; `ExShallow` may select partitions that have a higher cost, but are more explainable, than those of `ExGreedy`.)

4.2.2 The algorithm

`ExShallow` builds a decision tree in a top-down fashion as shown in

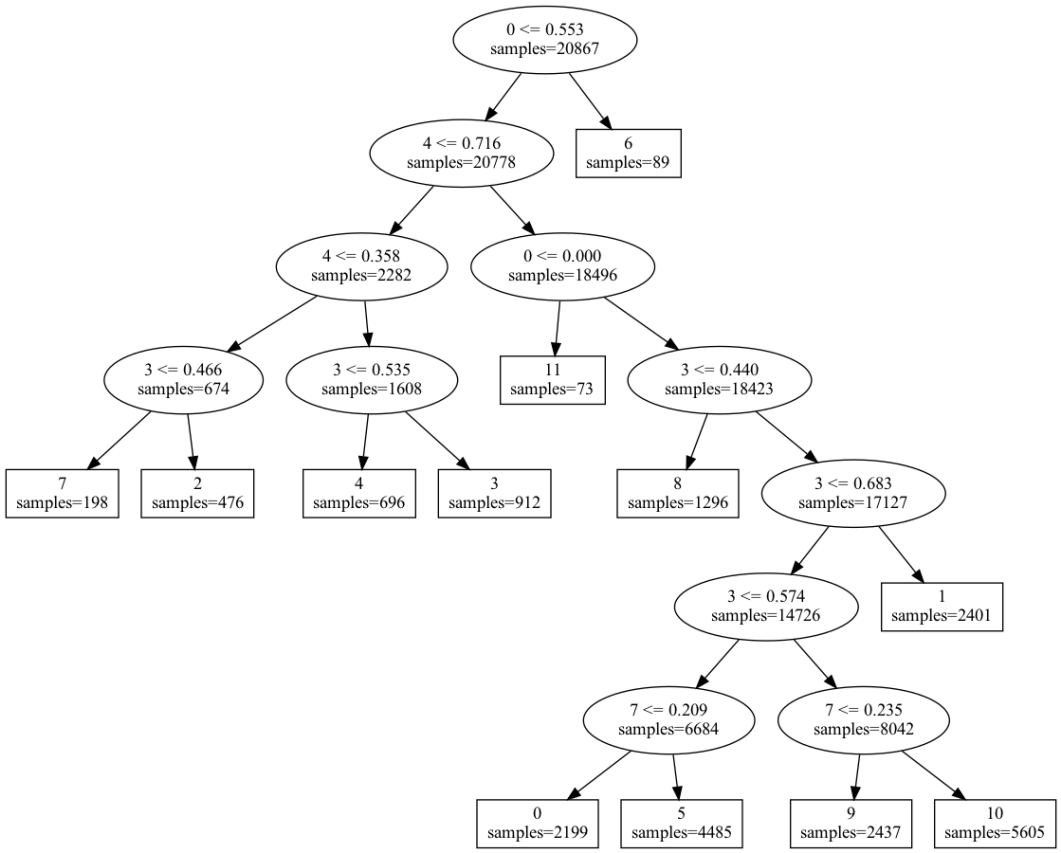


Figure 4.1: Tree from the `ExGreedy` algorithm for the `Avila` dataset, with `WAES` ≈ 5.4 and `WAD` ≈ 6.2.

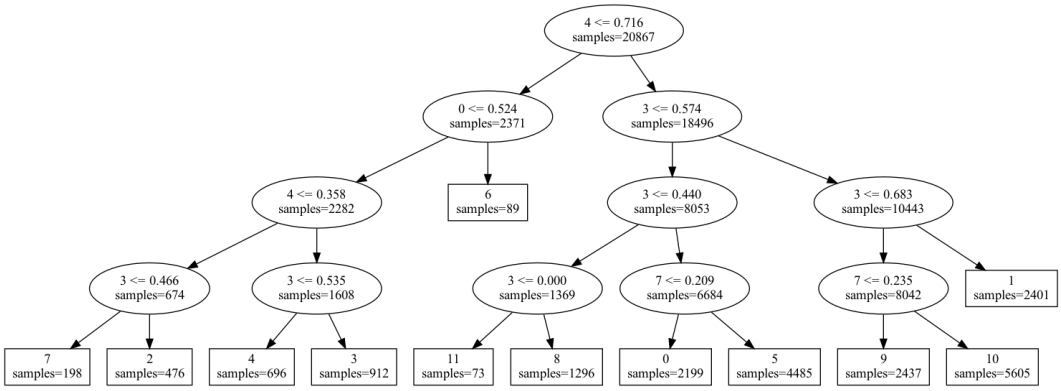


Figure 4.2: Tree from the `ExShallow` algorithm for the `Avila` dataset, with `WAES` ≈ 3.7 and `WAD` ≈ 3.8.

Algorithm 5. The strategy receives as input a set of points $\mathcal{X}'$ and a set $\mathcal{S}'$ of k representatives. We say that two cuts are equivalent with respect to set $\mathcal{X}' \cup \mathcal{S}'$ if they are associated with the same component (both are i-cuts for some i) and induce the same binary partition on $\mathcal{X}' \cup \mathcal{S}'$. Note that there are at most $|\mathcal{X}' \cup \mathcal{S}'|d$ pairwise non-equivalent cuts. At each node the strategy evaluates

$$\texttt{Price}(\gamma, \mathcal{X}', \mathcal{S}') + \lambda \cdot \texttt{DExp}(\gamma, \mathcal{X}', \mathcal{S}') \tag{4-4}$$

for each cut γ in the set of non-equivalent cuts that separate at least two reference centers from $\mathcal{S}'$. Then, it selects the cut γ^* for which (4-4) is minimum.

In Equation (4-4), $\texttt{Price}(\gamma,\mathcal{X}',\mathcal{S}')$ is associated to the quality of the partition: it is smaller if γ leads to a larger reduction in the k-means loss of the partition. Meanwhile, `DExp` (which stands for *Depth Explainability*) is a penalty term associated to the partition's explainability: it is larger if γ leads to a decision tree with high values for `WAD`/`WAES`. The parameter λ is used to govern the trade-off between cost and explainability.

After selecting γ^*, the strategy is recursively performed for each of the groups of the binary partition induced by γ^*. The recursion stops when $\mathcal{S}'$ contains only one reference center.

Algorithm 5 $\texttt{ExShallow}(\mathcal{X}', \mathcal{S}', \lambda)$

$\mathcal{X}'$: set of points; $\mathcal{S}'$: set of reference centers; $\lambda \in \mathbb{R}_{\geq 0}$: trade-off parameter

if $|S'| = 1$ **then**
  Return $\mathcal{X}'$ and the single reference center in S'
else
  $\mathcal{C} \leftarrow$ set of non-equivalent cuts w.r.t. $\mathcal{X}' \cup \mathcal{S}'$ that separate at least two centers in $\mathcal{S}'$
  $\gamma^* \leftarrow \arg\min_{\gamma \in \mathcal{C}} \{\texttt{Price}(\gamma, \mathcal{X}', \mathcal{S}') + \lambda \cdot \texttt{DExp}(\gamma, \mathcal{X}', \mathcal{S}')\}$
  $(\mathcal{X}_L^*, \mathcal{X}_R^*) \leftarrow$ partition of $\mathcal{X}'$ induced by γ^*
  $(\mathcal{S}_L^*, \mathcal{S}_R^*) \leftarrow$ partition of $\mathcal{S}'$ induced by γ^*
  Create a node u
  $u.\texttt{LeftChild} \leftarrow \texttt{ExShallow}(\mathcal{X}_L^*, \mathcal{S}_L^*)$
  $u.\texttt{RightChild} \leftarrow \texttt{ExShallow}(\mathcal{X}_R^*, \mathcal{S}_R^*)$
  Return the tree rooted at u

4.2.2.1 Evaluation of cut γ in terms of partition quality

Let $\mathcal{X}'$ and $\mathcal{S}'$ be, respectively, the sets of points and centers that reach some given node in the decision tree. In addition, let γ be a cut that splits $\mathcal{X}'$ into groups $\mathcal{X}'_{\mathcal{L}}$ and $\mathcal{X}'_{\mathcal{R}}$ and splits $\mathcal{S}'$ into groups $\mathcal{S}'_{\mathcal{L}}$ and $\mathcal{S}'_{\mathcal{R}}$, each of them containing at least one reference center. $\texttt{Price}(\gamma, \mathcal{X}', \mathcal{S}')$ is defined as

$$\texttt{Price}(\gamma, \mathcal{X}', \mathcal{S}') := \frac{\texttt{Cost}(\gamma, \mathcal{X}', \mathcal{S}')}{\texttt{CurrentCost}(\mathcal{X}', \mathcal{S}')}, \tag{4-5}$$

where

$$\texttt{CurrentCost}(\mathcal{X}', \mathcal{S}') := \sum_{\mathbf{x} \in \mathcal{X}'} \min_{\mathbf{c} \in \mathcal{S}'} ||\mathbf{x} - \mathbf{c}||_2^2 \tag{4-6}$$

and `Cost` is the cost function of `ExGreedy` (Equation (4-1)); that is, `CurrentCost` and `Cost` give, respectively, the cost of the partition before and after applying cut γ. In both cases, each point is associated with the closest valid reference center.

4.2.2.2 Evaluation of cut γ in terms of partition explainability

To obtain $\texttt{DExp}(\gamma, \mathcal{X}', \mathcal{S}')$, we first calculate $\widehat{\texttt{WAD}}(\gamma, \mathcal{X}', \mathcal{S}')$, an estimation of the quality of γ for finding a good tree in terms of WAD, and then we adjust $\widehat{\texttt{WAD}}(\gamma, \mathcal{X}', \mathcal{S}')$ to take WAES into account.

Estimating whether a cut is good or not in terms of WAD is a non-obvious task. For other metrics, such as the maximum depth of a tree, this is much simpler: the more balanced the cut, the better. To estimate the quality of the cut γ for our task, we efficiently compute (4-2) for an auxiliary tree that is built specifically for this purpose.

More precisely, $\widehat{\texttt{WAD}}(\gamma, \mathcal{X}', \mathcal{S}')$ is given by the return of the procedure presented in Algorithm 6. $\texttt{EvalWAD}(N, K, r_p, r_c)$ returns the WAD of a tree with K leaves (corresponding to centers) for a set of N points, where each node in the tree splits the points and the centers in the same proportion as γ does, that is, proportionally to $r_p = |\mathcal{X}'_L|/|\mathcal{X}'|$ and $r_c = |\mathcal{S}'_L|/|\mathcal{S}'|$, respectively. We note that these ratios do not change along the algorithm execution and that the resulting decision tree is just a theoretical tree (which may not even be feasible for the instance under consideration), built to estimate how good γ is for minimizing (4-2).

Algorithm 6 $\texttt{EvalWAD}(N, K, r_p, r_c)$

N: Current number of points; K: Current number of reference centers; r_p: Point-split ratio; r_c: Center-split ratio

if $K = 1$ **then**
    Return 0
else
    $K_L \leftarrow K \cdot r_c$
    $K_R \leftarrow K - K_L$
    $N_L \leftarrow N \cdot r_p$
    $N_R \leftarrow N - N_L$
    Return $1 + (N_L \cdot \texttt{EvalWAD}(N_L, K_L, r_p, r_c) + N_R \cdot \texttt{EvalWAD}(N_R, K_R, r_p, r_c))/N$

As an example, Figures 4.3 and 4.4 present two theoretical trees generated by Algorithm 6 for the same number of centers ($K = 4$) and points ($N = 128$), but different values of r_c and r_p. In Figure 4.3, $r_c = r_p = 0.5$; as a result, the tree generated by Algorithm 6 has 4 leaves at level 2 with 25 points in each. In Figure 4.4, $r_c = r_p = 0.25$; as a result, the tree has 3 levels instead of 2, and most points are in one of the deepest leaves.

The value of $\texttt{DExp}(\gamma, \mathcal{X}', \mathcal{S}')$ is given by the return of procedure $\texttt{EvalDExp}(\gamma, \mathcal{X}', \mathcal{S}')$, presented in Algorithm 7. To explain the procedure, let v be the current node of the decision tree under construction. Recall that a cut $\gamma = (i, \theta)$ applied on v induces two edges leaving v, one associated with condi-

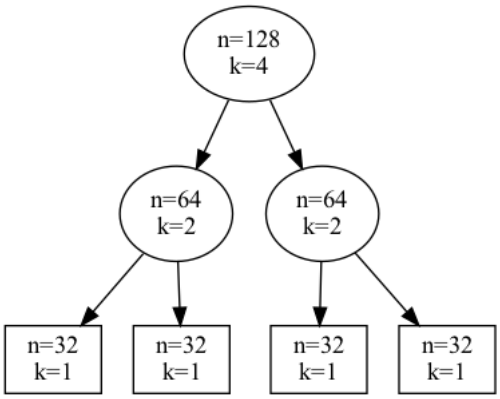


Figure 4.3: Tree generated by Algorithm 6 with $r_c = r_p = 0.5$, $\texttt{WAD} = 2$.

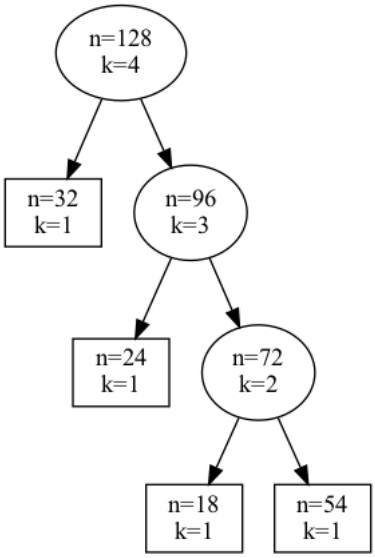


Figure 4.4: Tree generated by Algorithm 6 with $r_c = r_p = 0.25$, $\texttt{WAD} = 2.31$.

tion $x_i \leq \theta$ and the other with condition $x_i > \theta$. We say that an edge leaving v is *killer* if its associated condition turns some non-redundant condition in the path from the root to v into a redundant one. The procedure first determines which edges induced by γ on v are killer and, based on that, it adjusts the value of $\widehat{\texttt{WAD}}(\gamma, \mathcal{X}', \mathcal{S}')$ to take into account the `WAES`. As an example, if only the left edge leaving v is killer then we discount $|\mathcal{X}'_L|/|\mathcal{X}'|$ from $\widehat{\texttt{WAD}}(\gamma, \mathcal{X}', \mathcal{S}')$, because one condition in the path from the root to v becomes redundant to explain the clusters of the left subtree of v.

By design, `EvalDExp` prioritizes the choice of cuts at node v that are associated with coordinates that have already been used by some cut in the path from the root to v. This way the strategy tends to produce redundant conditions and, therefore, to find good trees in terms of `WAES`.

To summarize, `ExShallow` follows the steps of Algorithm 5. At line 5, it calls `EvalDexp`, presented in Algorithm 7, to calculate $\texttt{DExp}(\gamma, \mathcal{X}', \mathcal{S}')$, while

Algorithm 7 `EvalDExp`(γ, $\mathcal{X}'$, $\mathcal{S}'$)

γ: cut; $\mathcal{X}'$: set of points; $\mathcal{S}'$: set of centers

1: $v \leftarrow$ current node in the decision tree
2: $(\mathcal{X}'_L, \mathcal{X}'_R) \leftarrow$ partition of $\mathcal{X}'$ induced by γ
3: $(\mathcal{S}'_L, \mathcal{S}'_R) \leftarrow$ partition of $\mathcal{S}'$ induced by γ
4: $r_p = {\mathcal{X}'}_L/\mathcal{X}'$
5: $r_c = {\mathcal{S}'}_L/\mathcal{S}'$
6: $\widehat{\texttt{WAD}} = \texttt{EvalWAD}(|\mathcal{X}'|, |\mathcal{S}'|, r_p, r_c)$
7: **if** no edge induced by γ on v is killer **then**
8: Return $\widehat{\texttt{WAD}}$
9: **else**
10: **if** only the left edge induced by γ on v is killer **then**
11: Return $\widehat{\texttt{WAD}} - |\mathcal{X}'_L|/|\mathcal{X}'|$
12: **else**
13: **if** only the right edge induced by γ on v is killer **then**
14: Return $\widehat{\texttt{WAD}} - |\mathcal{X}'_R|/|\mathcal{X}'|$
15: **else**
16: Return $\widehat{\texttt{WAD}} - 1$

$\texttt{Price}(\gamma, \mathcal{X}', \mathcal{S}')$ is calculated via Equations (4-5), (4-6) and (4-1).

4.2.3 Implementation details and time-complexity analysis

`ExShallow` can be implemented in $O(nkd \cdot \texttt{WAD}(\mathcal{D}))$ time, where $\texttt{WAD}(\mathcal{D})$ is the `WAD` of the decision tree $\mathcal{D}$ built by the algorithm. Given the set of points $\mathcal{X}$ and the reference centers $\mathcal{S}$, the algorithm first obtains d sorted lists, where the i-th list corresponds to the set of points in $\mathcal{X} \cup \mathcal{S}$ sorted by component i. This initial sorting step takes $O(d(n+k)\log(n+k))$ time and is only performed at the root of the tree.

Having the d sorted lists at node v, we use the implementation from Section 4.1.1 to compute (4-1) for all valid cuts in $\mathcal{O}(dn_vk_v)$ time, where n_v and k_v are, respectively, the number of points and centers that reach v. The computation of $\widehat{\texttt{WAD}}$, via Algorithm 6, takes $O(k_v)$ time per cut, or $O(dn_vk_v)$ time for all cuts.

To find out which of the edges are killer in Algorithm 7, we maintain a data structure `A` with $2d$ entries. For each $i \in [d]$, `A`[i].left (resp. `A`[i].right) stores the number of left (resp. right) edges that leave i-nodes that lie in the path from the root to the current node. A left (resp. right) edge leaving an i-node is killer if and only if `A`[i].left > 0 (resp. `A`[i].right > 0).

The data structure `A` can be updated in $O(1)$ time: if the chosen cut at node v is an i-cut, then right before the recursive call at line 9 (resp. line 10) of `ExShallow` (Algorithm 5) we increment by one unit `A`[i].left (resp. `A`[i].right), and when we return from the recursion we decrease the respective counter by

1.

After selecting the cut at node v, the d sorted lists for the children of v are obtained in $O(n_v d)$ time from the sorted lists for v.

Thus, the total cost of the algorithm to build a tree $\mathcal{D}$ is proportional to

$$\sum_{v \in \mathcal{D}} n_v \cdot d \cdot k_v \leq \sum_{i=1}^{n} \ell_i \cdot d \cdot k, \tag{4-7}$$

where ℓ_i is the depth of data point i at $\mathcal{D}$. The rightmost term of (4-7), however, is equal to $\texttt{WAD}(\mathcal{D}) \cdot ndk$.

The $\mathcal{O}(ndk \cdot \texttt{WAD}(\mathcal{D}))$ time complexity suggests that trees with low `WAD` are faster to build – which is good for our purposes, since by design our algorithm tries to build trees with this property.

4.2.4 Setting the trade-off parameter

In a typical case, users are interested in obtaining an explainable clustering with low cost. To achieve this goal they have to properly set the value of λ. One possibility is performing a brute-force search over some set of values to find the one that yields the most suitable tree. However, this could be computationally expensive and also impractical from the users' perspective, as they would have to analyze many trees. Fortunately, we can avoid that.

First we note that a reasonable interpretation for λ is how much we are willing to locally spend on cost, in percentage, to reduce by one unit the average size of the explanations. As an example, setting $\lambda = 0.1$ means that we accept an additive loss of up to 10% in terms of the partition cost to have explanations one unit shorter on average.

Under this perspective, we shall avoid large values for λ, since partitions with high costs are not likely to produce coherent clusters. As we show in our experiments (Section 4.5), we can obtain significant improvements over the existing methods by setting $\lambda = 0.03$.

A good property of `Price` (4-5) is that its value for cuts of low `Cost` (4-1), the most relevant ones, lies in the interval $[1, 4k + 1]$, the same one in which, except for a constant factor, both `WAD` (4-2) and `WAES` (4-3) lie. Hence, we are trading off quantities with similar magnitudes, which is beneficial. This is formalized below.

Lemma 9 *Let $\mathcal{X}'$ and $\mathcal{S}'$ be the set of data points and reference centers that reach a given node v. Then, there is a cut γ' that satisfies* $1 \leq \texttt{Price}(\gamma', \mathcal{X}', \mathcal{S}') \leq 4|\mathcal{S}'| + 1$.

Proof. The left-hand side follows because any assignment between points and reference centers that is valid after applying a cut is also valid before the cut, so that $\texttt{CurrentCost}(\mathcal{X}', \mathcal{S}') \leq \texttt{Cost}(\gamma, \mathcal{X}', \mathcal{S}')$ for every cut γ.

For the right-hand side, let max_i and min_i be the maximum and minimum values of the i-th component among the centers in $\mathcal{S}'$, respectively. Moreover, let $b_i = max_i - min_i$ and let $p(\gamma)$ be the number of points in $\mathcal{X}'$ that are separated from their closest centers in $\mathcal{S}'$ when a cut γ is employed. We have that

$$\texttt{Cost}(\gamma, \mathcal{X}', \mathcal{S}') \leq \texttt{Cost}(\mathcal{X}', \mathcal{S}') + p(\gamma) \sum_{i=1}^{d} b_i^2.$$

The reason is that $\sum_{i=1}^{d} b_i^2$ is an upper bound on the contribution for the k-means cost of a point that is separated from its closest center.

On the other hand, it follows from Lemma 5.7 of (Moshkovitz et al. (2020)) that

$$\texttt{CurrentCost}(\mathcal{X}', \mathcal{S}') \geq \frac{p^*}{4|\mathcal{S}'|} \sum_{i=1}^{d} b_i^2,$$

where p^* is the number of points separated from their closest centers by the valid cut that separates the minimum number of points.

Thus, if γ is a cut that separates p^* points from its closest centers, we get that

$$\texttt{Cost}(\gamma, \mathcal{X}', \mathcal{S}') \leq (4|\mathcal{S}'| + 1)\texttt{CurrentCost}(\mathcal{X}', \mathcal{S}'),$$

establishing the result. ■

4.2.5 Illustration of `ExShallow` and the importance of `DExp`

In this section we present an example on a toy dataset to demonstrate how `ExShallow` may construct better trees in terms of explainability, and the importance of the
tt DExp parameter in doing so. Let $\mathbf{0}$ and $\mathbf{1}$ be points in $\mathbb{R}^k$ with all components equal to 0 and 1, respectively. The set of points in our instance, $\mathcal{X}$, is given by $X_0 \cup \ldots \cup X_{k-1}$, where $X_0 = \{\mathbf{0}\}$ and X_i, with $i \geq 1$, is a group that satisfies the following conditions:

1. X_i has k^{3i} points, so that $|X_i| = k^3|X_{i-1}|$, for $i \geq 1$;
2. The centroid (mean of its points) of X_i is the point with all components equal to $i \cdot n$;
3. The distance of every point in X_i to the centroid of X_i is at most 1;

4. Each point in X_i is a multiple of direction $\mathbf{1}$.

Properties 2 and 3 mean each subset $X_0, X_1, \ldots, X_{k-1}$ may be isolated from the others by an axis-aligned cut. A k-partition that isolates all the subsets will be the best one in terms of `Price` (4-5): any *clean* cut (one that does not separate points from their reference centers) has a `Cost` (4-1) of at most n, while any cut that is not clean has a `Cost` of at least kn^2, so all chosen cuts must be clean. The question is which set of clean cuts will be chosen, and in what order.

If $\lambda = 0$, all clean cuts are equally good, irrespective of the depths of each cluster in the final decision tree that induces the partition. So it may well be the case that the points in X_{k-1} end up in the deepest possible leaf, at level $k-1$, and that each cut is performed along a different dimension. Because the vast majority of the points in the dataset are concentrated in X_{k-1}, such a decision tree would have $\texttt{WAD} = \texttt{WAES} \approx k-1$.

Now consider $\lambda \in (0, 1]$. Clean cuts would still be preferred; they would have a cost (4-4) no higher than $1+k$, while cuts that are not clean would have a cost of at least nk. But, due to the `DExp` term, the first clean cut, performed at the root of the tree, would necessarily be one that separates X_{k-1} from the other groups.

To see that more clearly, note that when `EvalDExp` (Algorithm 7) is evaluated for a clean cut that separates X_{k-1} from the other groups, it calls `EvalWAD` (Algorithm 6) with $r_c = 1 - 1/k$ and $r_p = 1 - k^{3k-3}/n$. `EvalWAD` would then calculate the `WAD` of an auxiliary tree that has exactly one leaf at depth 1 containing $k^{3(k-1)}$ points. Since in this tree the other points lie at depth at most $k-1$ and the number of such points can be upper-bounded by $(k-1)k^{3(k-2)}$, its `WAD`, for $k > 3$, is at most

$$\frac{k^{3(k-1)} + (k-1)^2 k^{3(k-2)}}{n} \leq 1 + \frac{1}{k} \leq \frac{4}{3}, \tag{4-8}$$

where we use the fact that $n > k^{3(k-1)}$ to obtain the first inequality.

On the other hand, the auxiliary tree for a clean cut at the root that isolates X_0 would have $\texttt{WAD} \geq 2 - 1/n > 4/3$ because only one point lies at depth 1 in the auxiliary tree. Moreover, any other clean cut induces a tree with no leaf at depth 1, so the `WAD` of their auxiliary trees would be at least 2.

Therefore, the cut applied to the root separates X_{k-1} from the rest of $\mathcal{X}$. Since no cuts have yet been applied, this cut is not killer; so `EvalDExp` returns the same value as `EvalWAD`, and `ExShallow` selects any cut that isolates X_{k-1}.

Then the algorithm recurses on the partitions $\mathcal{X}_L = X_0 \cup \ldots \cup X_{k-2}$ and $\mathcal{X}_R = X_{k-1}$. The latter is an isolated cluster, so no cut is applied to it; for the

former, the same argument as above leads to the conclusion that a cut that isolates X_{k-2} will be chosen. This time, however, cuts from the same dimension as the one chosen at the root will be killer, and therefore will have $\texttt{DExp} < \widehat{\texttt{WAD}}$. We can generalize this to the rest of the tree and conclude that, at any level i, a cut that isolates X_{k-i} from $X_0 \cup \cdots \cup X_{k-i-1}$ will be chosen, and that all cuts in the tree will be along the same axis.

The final tree constructed by `ExShallow` with $\lambda \in (0, 1]$ will have the same characteristics as the one `EvalWAD` generated to evaluate the cuts at the root that isolated X_{k-1}, with $\texttt{WAD}, \texttt{WAES} \approx 1$ – a great improvement over the worst-case scenario of $\texttt{WAD} = \texttt{WAES} \approx k - 1$ for using `ExShallow` with $\lambda = 0$.

4.3 `ExBisection`: Explainable clustering from scratch

Both `ExGreedy` (Section 4.1) and `ExShallow` (Section 4.2), as well as other explainable k-means clustering algorithms found in the literature ((Moshkovitz et al. (2020), Frost et al. (2020), Makarychev & Shan (2021b), Esfandiari et al. (2021)); see Section 4.4), rely on a (hopefully optimal) unexplained partition $\mathcal{P}_u$ as a starting point, and establish a rule for finding axis-aligned cuts of the data according to the clusters and/or representatives of $\mathcal{P}_u$. The initial set of reference centers can be built by any algorithm for the (non-explainable) k-means clustering problem, such as Lloyd's algorithm (Lloyd (1982), Leskovec et al. (2020)). `ExBisection`, presented below, is an explainable clustering algorithm that is built "from scratch," that is, without the need of a previous partition. As far as I can tell, this is the first such algorithm presented in the literature.

Algorithm 8 builds a decision tree that induces a k-partition of the data by hierarchically choosing the axis-aligned cut that most reduces the sum of intra-cluster variances. It can be seen as a restricted case of the clustering algorithm through bisection minimizing the k-means cost (Steinbach at al. (2000)).

Algorithm 8 `ExBisection`$(\mathcal{P}, k, k^*)$

$\mathcal{P}$: set of clusters; k: number of clusters in $\mathcal{P}$; k^*: number of desired clusters

1: **if** $k = k^*$ **then**
2: Return $\mathcal{P}$
3: **else**
4: $\gamma^* \leftarrow$ axis-aligned cut that creates a new cluster in $\mathcal{P}$ that minimizes the sum of intra-cluster variances
5: $\mathcal{P}' \leftarrow$ the $k+1$-partition generated by applying γ to $\mathcal{P}$
6: Return `ExBisection`$(\mathcal{P}', k+1, k^*)$

4.3.1 Complexity analysis

Step 4 in Algorithm 8 involves finding the axis-aligned cut that will split in two one of the clusters in the current partition while minimizing the k-means cost function. To do this efficiently, we can sort the data across all dimensions, storing the sorting indices in an $n \times d$ matrix. To track which point is in which cluster, we use an additional vector with n dimensions.

At the outset, $\mathcal{P}$ has a single cluster, comprising all elements in the dataset. We build the sorting matrix in $\mathcal{O}(dn \log n)$ time, and calculate the k-means cost of this single cluster in $\mathcal{O}(dn)$. To find the first cut, we run through all possible cuts in all d dimensions, keeping track of the k-means cost of both clusters each cut would generate. As the cuts are being considered in sorted order, updating the cost of each cluster takes $\mathcal{O}(d)$ time. Since we have nd cuts to evaluate, finding the best cut takes $\mathcal{O}(nd^2)$ time. We therefore have a time complexity of $\mathcal{O}(dn(\log n + kd))$ for the algorithm as a whole.

Other than keeping track of the sorted elements across all dimensions and clusters, we can also store the best cut, and the gain cost reduction it would provide if implemented, for each cluster. For instance, if $|\mathcal{P}| = 2$, we will find the best cut for the two clusters $C_1, C_2 \in \mathcal{P}$, but only one of these cuts will be implemented. Without loss of generality, assume C_1's cut leads to a larger reduction in the k-means cost function of $\mathcal{P}$, and that after implementing it $\mathcal{P}' = \{C_2, C_3, C_4\}$, where C_3, C_4 are created by splitting C_1 with the best possible cut. When evaluating $\mathcal{P}'$, we don't need to find the best cut for C_2 again; we just need to worry about C_3 and C_4. More generally, if a cut is implemented in C^*, in the next iteration of the algorithm we only need to find the best cuts for the two clusters generated by splitting C^* into two. While this does not affect the theoretical complexity of Algorithm 8, it can improve its running time in practice.

4.4 Other algorithms for explainable k-means clustering

We briefly present below other algorithms for explainable k-means clustering found in the literature. `IMM` (Moshkovitz et al. (2020)) and `ExKMC` (Frost et al. (2020)) are similar to `ExGreedy` and `ExShallow` in that they start from an unrestrained partition $\mathcal{P}_u$ and build an explainable partition by selecting axis-aligned cuts that minimize some cost associated to it: the number of points in the dataset separated from their representatives from $\mathcal{P}_u$ (`IMM`) or the cost of an explainable partition that must select its representatives among those in $\mathcal{P}_u$ (`ExKMC`). By contrast, `RandomThresholds` (Makarychev & Shan (2021b),

Esfandiari et al. (2021), Gamlath et al. (2021)) is simply concerned with finding axis-aligned cuts that separate the *representatives* from $\mathcal{P}_u$ well, without using the dataset being clustered to evaluate the cost associated to these cuts. These three algorithms, along with `ExGreedy`, `ExShallow`, and `ExBisection`, will be empirically evaluated in Section 4.5 below.

4.4.1 `IMM`: Minimizing mistakes

Iterative Mistake Minimization (Moshkovitz et al. (2020)), summarized in Algorithm 9, starts from an unconstrained partition $\mathcal{P}_u$ and builds an explainable partition by iteratively finding cuts that minimize the number of *mistakes*, or points that are separated from their representatives in $\mathcal{P}_u$. Its running time is $\mathcal{O}(kdn \log n)$, not considering the time to find $\mathcal{P}_u$. It is proven in Theorem 3 of (Moshkovitz et al. (2020)) that the partition generated by `IMM` is an $\mathcal{O}(k^2)$ approximation to the cost of the optimal unrestricted partition.

Algorithm 9 `IMM`$(\mathcal{X}, M)$

$\mathcal{X}$: set of points; M: set of k representatives

1: Associate each point $x \in \mathcal{X}$ to its closest representative in M
2: $\Gamma = \{\}$
3: $i = 1$
4: **while** $i < k$ **do**
5: $\quad \gamma^* \leftarrow$ axis-aligned cut that separates at least one representative from another, and that minimizes the number of points in $\mathcal{X}$ that are separated from their closest representatives.
6: $\quad \Gamma \leftarrow \Gamma \cup \{\gamma^*\}$
7: $\quad i \leftarrow i + 1$
8: Return the partition induced by the cuts in Γ

4.4.2 `ExKMC`: Maintaining explainability

Similarly to `ExBisection`, Explainable k-Means Clustering, or `ExKMC` (Frost et al. (2020)), iteratively selects the axis-aligned cut that leads to the best explainable k-partition at each step. The main difference between both algorithm is that `ExKMC` uses an existing, unrestricted partition $\mathcal{P}_u$ to evaluate the cuts: the *surrogate cost* of a cluster $C \in \mathcal{P}$ is given by

$$\texttt{SurrogateCost}(C, M) = \min \left\{ \sum_{x \in C} ||x - \mu||_2^2 \right\} \forall\, \mu \in \{C \cap M\}, \qquad (4\text{-}9)$$

where M is the set of representatives from $\mathcal{P}_u$. In other words, the cost of a cluster during the execution of `ExKMC` is the smallest possible k-means cost of the cluster if its representative is one of the representatives from $\mathcal{P}_u$.

Algorithm 10 `ExKMC`$(\mathcal{X}, M)$

$\mathcal{X}$: set of points; M: set of k representatives

1: Associate each point $x \in \mathcal{X}$ to its closest representative in M
2: $\Gamma = \{\}$
3: $i = 1$
4: **while** $i < k$ **do**
5: $\quad \gamma^* \leftarrow$ axis-aligned cut that separates at least one representative from another, and that minimizes the sum of the surrogate cost (4-9) over all clusters in the partition induced by $\Gamma \cup \{\gamma^*\}$.
6: $\quad \Gamma \leftarrow \Gamma \cup \{\gamma^*\}$
7: $\quad i \leftarrow i + 1$
8: Return the partition induced by the cuts in Γ

The advantage of this approach over `ExBisection` is the time complexity: as shown in (Frost et al. (2020)), `ExKMC` runs in $\mathcal{O}(dn(\log n + k))$, a factor of d faster in `ExBisection` when $kd > \log n$. However, when running `ExKMC` from scratch one must incur the additional cost of finding an unrestricted partition in the first place.

4.4.3 `RandomThresholds`: near-optimal explainable k-means clustering

Presented in different papers independently (Makarychev & Shan (2021a), Esfandiari et al. (2021), Gamlath et al. (2021)), `RandomThresholds` builds an explainable partition based on an unconstrained partition $\mathcal{P}_u$ by randomly selecting axis-aligned cuts that create a good separation of the representatives from $\mathcal{P}_u$. Notably, the algorithm never looks at the dataset being partitioned; only the representatives are considered. We summarize it in Algorithm 11, based on the implementation from (Makarychev & Shan (2021a)).

`RandomThresholds` is not presented as a practical algorithm; instead, as mentioned in Section 2.1.1.1, it has been extensively used to improve the upper bound of the price of explainability of both the k-means and the k-medians explainable clustering problems. As the best known theoretical algorithm for the explainable k-means problem, however, it is natural to wonder how it fares in practice. To our knowledge, Section 4.5 below, adapted from (Laber et al. (2023)), presents the first empirical evaluation of the algorithm.

Algorithm 11 `RandomThresholds`($\mathcal{S}$, ε)

$\mathcal{S}$: set of k representatives; ε: threshold parameter

$\mathcal{T} \leftarrow$ tree with a single node containing all representatives in $\mathcal{S}$
while $\mathcal{T}$ contains a leaf with two representatives **do**
  Sample $i \in \{1, 2, ..., d\}, \theta \in (0, 1), \sigma \in \{-1, 1\}$ uniformly at random
  for leaf $u \in \mathcal{T}$ with more than 1 representative **do**
    $m^u \leftarrow$ median of all representatives in u
    $R_u \leftarrow \max\{||s - m^u||^2 \,|\, s \in S_u\}$
    $S_u^l \leftarrow \{s \in S_u \,|\, s_i \leq m_i^u + (\sigma + \varepsilon)\sqrt{\theta} R_u\}$
    $S_u^r \leftarrow \{s \in S_u \,|\, s_i \geq m_i^u + (\sigma - \varepsilon)\sqrt{\theta} R_u\}$
    if S_u^l and S_u^r are not empty **then**
      Split u using cut $(i, m^u + \sigma\sqrt{\theta} R_u)$
      Assign S_u^l and S_u^r to u's children
Return $\mathcal{T}$

4.5 Experiments

In this section we present and analyze the results of experiments that compare the performances of the practical algorithms described in the previous sections. We used 16 datasets of different sizes and characteristics, running 30 seeded iterations in each of them. For each iteration, we find an unrestricted partition of the data by running Lloyd's algorithm (Lloyd (1982)) with the ++ initialization (Arthur & Vassilvitskii (2006)), as implemented in Python's `scikit-learn` package (Pedregosa et al. (2011)). This unrestricted partition is provided to `IMM` and to `ExKMC`, as implemented in the `ExKMC` package (Frost et al. (2020)), and to `ExGreedy`, `RandomThreshold`, and `ExShallow`, implemented by us and available at `https://github.com/lmurtinho/ShallowTree`. `ExBisection`, which is deterministic, is run once for each dataset.

We also performed statistical tests with Python's `scipy` package (Virtanen et al. (2020)) to verify the statistical significance of our results. For the algorithms that depend on an initial unexplained partition, and therefore partition the data in different ways depending on the starting point, we performed one-sided t-tests, assuming the same variance for both distributions, with a confidence level of 95%. We also calculated confidence intervals for the results of `ExShallow` for each dataset and checked whether the (deterministic) results for `ExBisection` are within those intervals.

Table 4.1 presents the size, dimension, and number of classes (which we use as the number of clusters) of the datasets in which we perform the experiments. All datasets are available online, and our code includes a script for retrieving and running tests on them. The numbers of instances,

Table 4.1: Dataset summary: n is the number of data points, d is the number of dimensions, and k is the number of classes/desired clusters.

Dataset	n	d	k	**Source**
Anuran	7,195	22	10	UCI
Avila	20,867	10	12	UCI (De Stefano et al. (2018))
Beer	1,514,999	5	104	OpenML
BNG (audiology)	1,000,000	85	24	OpenML
Cifar10	60,000	3,072	10	(Krizhevsky, 2009)
Collins	1,000	19	30	OpenML
Covtype	581,012	54	7	OpenML (Collobert et al., 2002)
Digits	1797	64	10	UCI (Alpadyin & Kaynak, 1998)
Iris	150	4	3	UCI (Fisher, 1936)
Letter	20,000	16	26	(Hsu & Lin, 2002)
Mice	552	77	8	OpenML (Higuera et al., 2015)
20Newsgroups	18,846	1,069	20	`http://qwone.com/~jason/20Newsgroups/`
Pendigits	10,992	16	10	UCI
Poker	1,025,010	10	10	UCI
Sensorless	58,509	48	11	UCI
Vowel	990	10	11	UCI

dimensions, and features are those of the final dataset used in our experiments (after removing missing values and one-hot encoding categorical variables, for instance). All datasets are anonymized and present no offensive content.

4.5.1 Results

Table 4.2 shows the main results of our experiments for the 16 datasets and for 6 different explainable clustering algorithms: `ExShallow` with $\lambda = 0.03$ (SHA), `ExBisection` (BIS), `ExGreedy` (GRD), `IMM` (IMM), `ExKMC` (KMC), and `RandomThreshold` (RDM). Best results are shown in bold. The partition costs are normalized by the cost of the unrestricted partition used as a starting point for the explainable clustering algorithms.

`ExShallow` produces the best partitions, in terms of normalized cost, for 10 of the 16 datasets; `ExBisection` and `ExGreedy` do so for 6 datasets each. The best results for the other algorithms from the literature is 3 (`IMM`). Notably, `RandomThresholds`, the algorithm with the best theoretical guarantees, only reaches the best normalized partition cost in the `20Newsgroups` dataset – in which all algorithms but `ExShallow` do the same.

For explainability measures, `ExBisection` induces partitions with the lowest `WAES` for 7 datasets, and the lowest `WAD` for 6 of them. `ExShallow` reaches the best result for 4 datasets in terms of `WAES` and 5 in terms of `WAD`; however, as we'll see in Sections 4.5.2 and 4.5.3 below, it is possible to adjust its λ parameter to reach better results in these metrics. `ExGreedy`

Table 4.2: Full results of experiments for all datasets and algorithms. `ExShallow` (SHA) is run with $\lambda = 0.03$. Best results are in bold. Partition costs are normalized by the cost of the unrestricted partition used as a starting point for the explainable clustering algorithms.

		Normalized Partition Cost						`WAES`						`WAD`					
Dataset	k	SHA	BIS	GRD	IMM	KMC	RDM	SHA	BIS	GRD	IMM	KMC	RDM	SHA	BIS	GRD	IMM	KMC	RDM
anuran	10	1.16	1.21	**1.15**	1.28	1.32	1.71	3.75	**3.24**	4.17	5.37	3.41	3.89	3.79	3.51	4.27	5.67	**3.41**	4.05
avila	12	**1.05**	1.13	**1.05**	1.07	1.18	1.35	3.87	**3.20**	5.58	5.25	3.26	4.87	4.60	**4.18**	6.64	6.61	4.47	6.47
beer	104	1.16	**1.07**	1.19	1.83	1.27	1.55	7.35	6.39	8.13	7.80	**6.34**	7.27	10.47	**7.23**	15.09	54.31	7.35	11.08ß
bng	24	1.05	**1.01**	1.02	1.04	1.03	1.05	**3.50**	4.66	5.41	8.82	4.60	4.31	**3.50**	4.66	5.41	11.82	4.60	4.43
cifar10	10	1.16	**1.15**	1.17	1.22	1.19	1.26	**3.37**	3.49	3.60	5.70	3.63	3.42	**3.37**	3.49	3.60	5.70	3.63	3.42
collins	30	1.18	**1.16**	1.17	1.23	1.23	1.42	5.56	**5.04**	13.12	12.81	5.61	7.65	5.86	**5.24**	15.29	17.00	5.83	8.35
covtype	7	**1.03**	1.10	**1.03**	**1.03**	1.13	1.34	2.61	**2.38**	2.62	2.61	2.45	2.54	3.15	2.90	3.56	3.55	**2.82**	3.31
digits	10	**1.19**	**1.19**	1.21	1.23	1.22	1.42	3.96	4.04	5.65	5.36	3.80	**3.55**	3.96	4.04	5.65	5.36	3.80	**3.59**
iris	3	**1.04**	1.10	**1.04**	**1.04**	**1.04**	1.45	1.67	**1.33**	1.67	1.44	1.44	1.52	**1.67**	2.00	**1.67**	**1.67**	**1.67**	1.68
letter	26	**1.19**	1.30	1.23	1.30	1.36	1.53	5.26	**4.94**	11.37	12.64	5.44	7.03	5.48	**5.09**	12.50	14.85	5.54	7.72
mice	8	**1.07**	1.09	1.09	1.12	1.15	1.37	3.17	**2.75**	3.32	3.53	3.12	3.08	3.24	**3.00**	3.58	3.76	3.13	3.40
newsgroups	20	1.05	**1.01**	**1.01**	**1.01**	**1.01**	**1.01**	**1.12**	11.29	15.61	15.53	13.80	14.99	**1.22**	11.39	15.63	15.53	13.80	15.02
pendigits	10	**1.14**	1.18	1.14	1.24	1.32	1.70	3.70	3.80	4.43	4.31	**3.49**	3.56	3.77	3.80	4.46	4.44	**3.50**	3.71
poker	10	**1.10**	1.11	1.10	1.10	1.12	1.14	3.35	3.47	3.37	3.37	**3.23**	3.24	3.35	3.47	3.37	3.37	**3.23**	3.26
sensorless	11	**1.02**	1.05	**1.02**	1.03	1.07	1.32	**2.99**	3.84	4.24	4.10	3.99	4.06	**3.84**	4.13	4.52	4.44	4.07	4.29
vowel	11	**1.21**	1.21	1.25	1.36	1.29	1.50	3.89	3.64	5.26	5.74	**3.63**	3.84	3.94	**3.64**	5.76	6.41	3.64	4.09

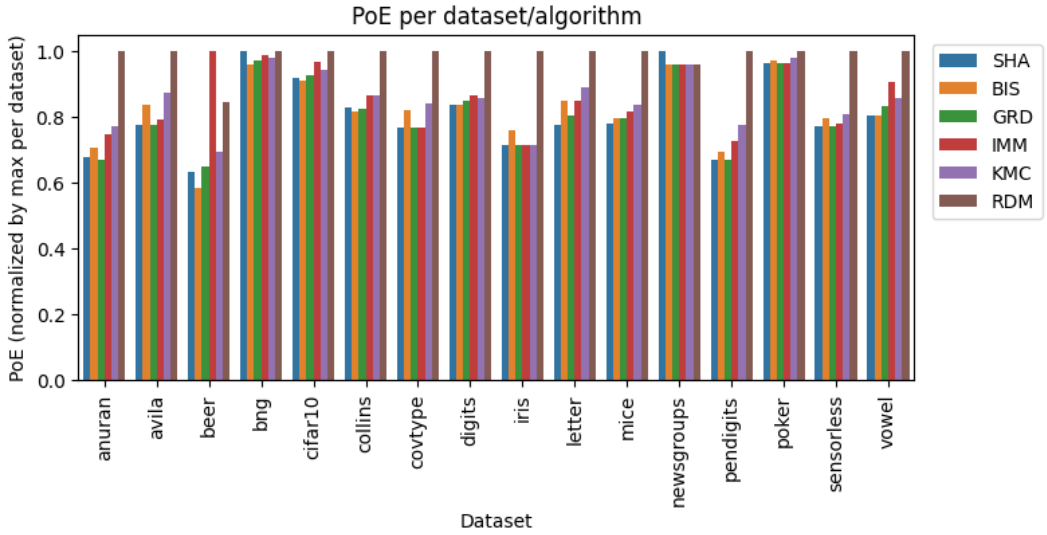


Figure 4.5: Price of explainability (normalized by the maximum value per dataset) per dataset and algorithm.

behaves relatively poorly in terms of explainability, as does, once again, `RandomThresholds`.

Figures 4.5, 4.6, and 4.7 show, respectively, the `PoE`, `WAES` and `WAD` per dataset and algorithm, normalized by the maximum value for each dataset. In terms of `PoE`, we can see that, for the most part, the algorithms tend to present similar results, but both `RandomThresholds` and (for the `beer` dataset) `IMM` sometimes present significantly higher prices than the other four algorithms. In terms of `WAES`, the negative outliers are `IMM` and `ExGreedy`, and `ExShallow` performs much better than the five other algorithms in the `newsgroups` and `sensorless` datasets. The observations regarding `WAES` also apply to `WAD`, with the addition of the relatively bad results for `RandomThresholds` in the `avila` dataset.

One noteworthy result is that `ExBisection` frequently outputs trees that are more explainable, on average, than the ones from `ExShallow`, even though the latter has an explicit penalty term designed to avoid the construction of less explainable trees. We conjecture that this happens because, as an algorithm that builds explainable partitions from scratch, `ExBisection` is not restricted to cuts that separate representatives from an original unrestricted partition, and can therefore find cuts that are better in terms of explainability even though it is not explicitly looking for them.

In summary, our experiments suggest that `ExShallow` is almost always at least close to the best result in terms of both cost and explainability, and frequently has a significant advantage in at least one of these dimensions (and frequently both of them) when compared to the other algorithms. `ExBisection`

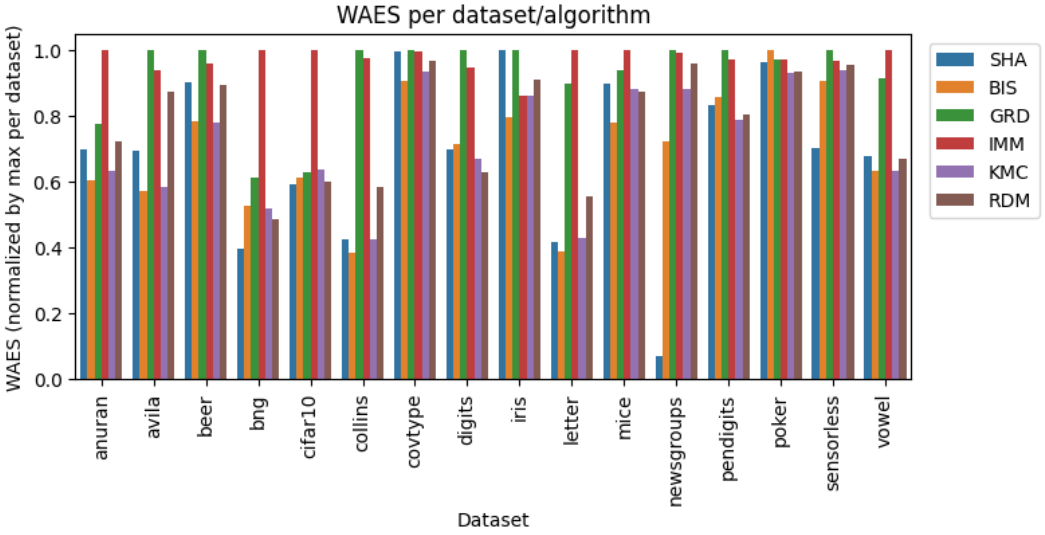


Figure 4.6: `WAES` (normalized by the maximum value per dataset) per dataset and algorithm.

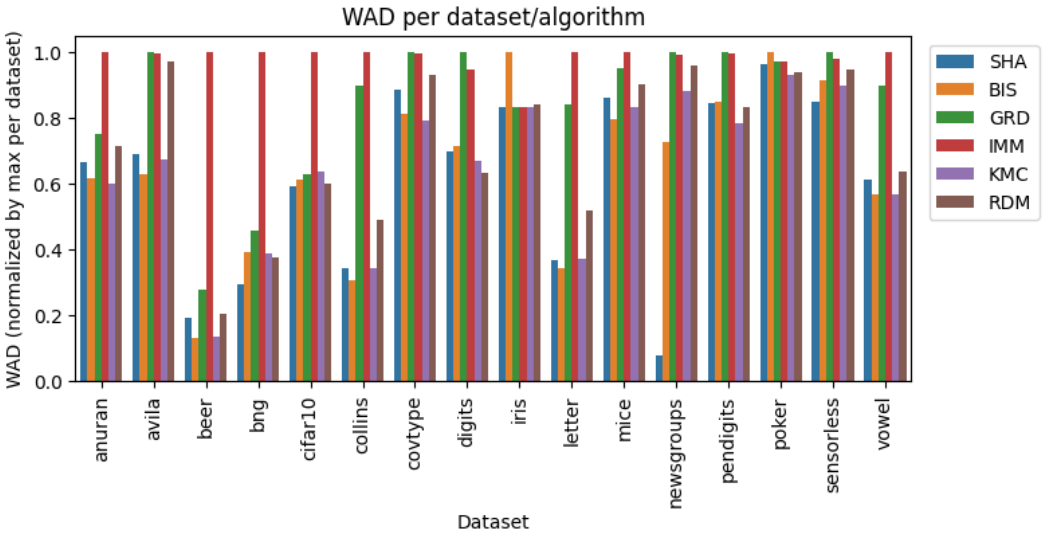


Figure 4.7: `WAD` (normalized by the maximum value per dataset) per dataset and algorithm.

Table 4.3: Mean normalized information score for all datasets and algorithms. `ExShallow` (SHA) is run with $\lambda = 0.03$. Best results for each dataset are in bold. The clusters returned by the unexplained partition (via Lloyd's algorithm) are treated as the ground truth.

Dataset	k	SHA	GRD	IMM	KMC	RDM
anuran	10	0.70	**0.72**	0.70	0.64	0.47
avila	12	0.73	0.72	**0.73**	0.68	0.58
beer	104	**0.83**	0.82	0.76	0.81	0.70
bng	24	0.29	**0.41**	0.25	0.38	0.24
cifar10	10	**0.29**	**0.29**	0.25	0.27	0.12
collins	30	**0.55**	0.54	0.54	0.53	0.39
covtype	7	**0.83**	**0.83**	**0.83**	0.72	0.62
digits	10	**0.58**	0.55	0.55	0.54	0.26
iris	3	**0.91**	**0.91**	**0.91**	**0.91**	0.73
letter	26	**0.61**	0.58	0.56	0.53	0.41
mice	8	**0.72**	0.71	0.71	0.65	0.48
newsgroups	20	0.10	0.55	**0.56**	0.53	0.51
pendigits	10	0.77	**0.77**	0.72	0.67	0.48
poker	10	0.41	0.41	0.41	0.40	**0.49**
sensorless	11	0.91	**0.92**	**0.92**	0.88	0.75
vowel	11	**0.58**	0.55	0.53	0.52	0.33

also tends to be close to the best observed results in terms of both cost and explainability, but with markedly worse results than `ExShallow` in the `newsgroups` and (for `WAES`) `sensorless` datasets. `ExGreedy` performs very well in terms of `PoE`, but with high values for `WAES` and `WAD` in several datasets.

For algorithms presented elsewhere in the literature, `IMM` performs relatively poorly in terms of explainability, and, for the `beer` dataset, in terms of `PoE` as well; `ExKMC` has a better performance, comparable to that of `ExBisection` in most datasets; and `RandomThresholds` has arguably the worse results in terms of `PoE`, which may be somewhat surprising as it is the algorithm with the best theoretical guarantees.

We also report in Table 4.3 the normalized mutual information score (NMI) (Strehl & Ghosh (2002)) of the partitions generated by the explainable algorithms, considering that the ground truth is the unrestrained partition from which they are derived; a value of 1 corresponds to a perfect correspondence between partitions. We remove `ExBisection` from this analysis because its partition does not use centroids from an unrestricted partition as an input.

On average, `ExShallow`'s partitions are closest to the unrestrained ones for 7 datasets, and as good as those generated by its competitors in another one. `ExShallow` returns the worst partitions in terms of NMI for a single dataset, `20Newsgroups` – for which, as can be seen in Table 4.2, its results in terms

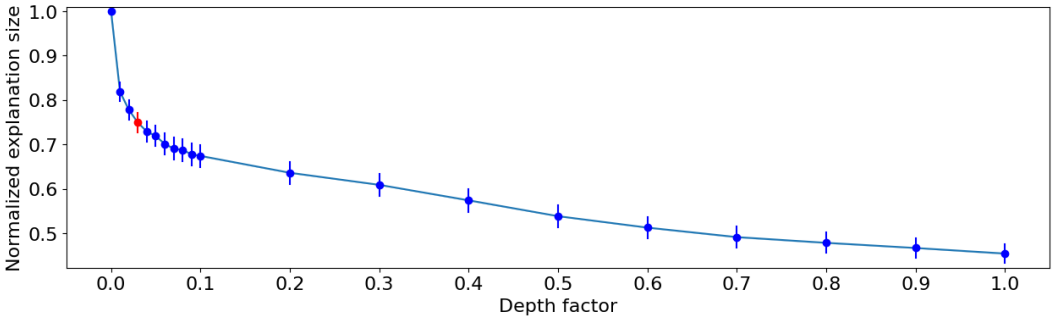


Figure 4.8: Mean `WAES` per depth factor (for all datasets), normalized by the results for $\lambda = 0$ for each dataset. Error bars (with a confidence interval of 95%) are calculated using Python's `scipy` package (Virtanen et al. (2020)).

of explainability are quite remarkable: while the other algorithms produce partitions explained by over 11 cuts on average, the partitions generated by `ExShallow` are explained by an average of fewer than 2 cuts.

4.5.2 `ExShallow`: Sensitivity of cost and explainability to variations in λ

Figure 4.8 shows how the average `WAES` of the partitions produced by `ExShallow` changes as λ increases. To allow for a comparison between datasets, the values are normalized by those of the tree when $\lambda = 0$ (i.e., when our cost function does not penalize cuts that lead to less explainable partitions). For each dataset, we ran 10 seeded iterations of Lloyd's algorithm and used the resulting partitions as a starting point for each instance of `ExShallow` with different values of λ.

`ExShallow` behaves as expected, with larger values of λ associated with trees having lower `WAES`, on average. (Results for `WAD` are omitted as they are very similar in terms of correlation with λ.) We observe a sharp drop for small increments of λ when starting from zero. The red value is 0.03, the one employed in the previous experiments.

Figure 4.9 shows how the mean cost of the partitions produced by our algorithm changes as λ increases. To allow for a comparison between datasets, the costs are normalized by the cost of the unrestricted partition generated by Lloyd's algorithm. The behavior is, in general, the expected one, with larger values of λ associated with higher costs.

Combining these figures leads to the important, and perhaps surprising, empirical conclusion that working with a small λ is very beneficial, as it significantly reduces the average weighted depth and explanation size without increasing the average cost of the partition.

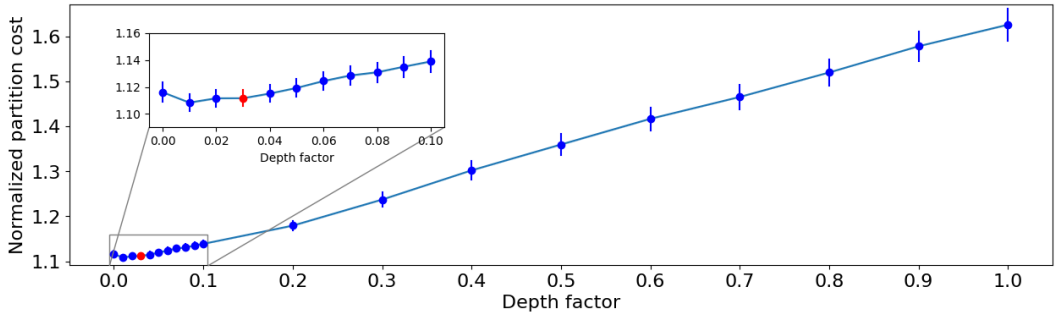


Figure 4.9: Mean normalized partition cost per depth factor (for all datasets), normalized by the cost of the unrestricted partition used to build the explainable partition. Error bars (with a confidence interval of 95%) are calculated using `scipy`.

4.5.3 `ExShallow`: Calibrating the trade-off between quality and explainability

The results presented in Figures 4.8 and 4.9 suggest that calibrating λ may lead to significant improvements when `ExShallow` does not initially return partitions that are satisfactory in terms of either quality (cost) or explainability (`WAES` and/or `WAD`). We believe the results presented in Table 4.2 indicate that `ExShallow` "out of the box" is at least competitive with, and arguably superior to, the most recent comparable algorithms in the literature, but there is some room for improvement. For instance, although the partition induced by `ExShallow` for the `20Newsgroups` dataset is much more explainable than those induced by the competition, the quality of the partition (both in terms of cost and NMI) suffers from it; and in many cases `ExKMC` induces partitions that are slightly more explainable, although their quality tends to be worse.

We can use the λ parameter to adjust the trade-off between partition quality and explainability in `ExShallow`, something that is not possible in the other algorithms presented here. To do so, we devised a simple binary search strategy, that starts from our default value of $\lambda = 0.03$ and then, if necessary, decreases it to try and find a partition with smaller cost, or increases it to try and find a partition with smaller `WAES`. Given a goal cost c^* and a goal `WAES` w^*, the binary search aims to find a partition with cost $c \leq c^*$ and `WAES` $w \leq w^*$; if it is unable to do so, it returns the partition with the smallest `WAES` given that its cost does not exceed c^*.

We present the results of this binary search, over 30 seeded iterations for each algorithm, in Table 4.4. Considering that `ExKMC` frequently beats `ExShallow` in terms of `WAES`, we used its results as our goal, to verify if we can "dominate" its results (i.e., induce partitions that have, on average, both smaller costs and explanation sizes) in the datasets under analysis.

Table 4.4: Comparison between results for `ExKMC` and `ExShallow`*. Best results for each dataset are in bold. Statistically better values at a 95% confidence level are in blue; statistically worse values at the same confidence level are in red.

		NPC		WAES		WAD		NMI	
Dataset	k	ExKMC	ExShallow*	ExKMC	ExShallow*	ExKMC	ExShallow*	ExKMC	ExShallow*
Anuran	10	1.32	**1.20**	3.41	**3.19**	3.41	**3.33**	0.64	**0.68**
Avila	12	1.18	**1.15**	3.26	**3.24**	4.47	**3.76**	**0.68**	0.64
Beer	104	1.27	**1.22**	**6.34**	7.25	**7.35**	10.53	0.81	**0.82**
BNG	24	1.03	**1.02**	4.60	**4.50**	4.60	**4.50**	0.38	0.38
Cifar10	10	1.19	**1.16**	3.63	**3.37**	3.63	**3.37**	0.27	**0.29**
Collins	30	1.23	**1.20**	5.61	**4.97**	5.83	**5.42**	0.53	0.53
Covtype	7	1.13	**1.12**	2.45	**2.44**	2.82	**2.65**	0.72	**0.75**
Digits	10	1.22	**1.19**	3.80	**3.65**	3.80	**3.65**	0.54	**0.56**
Iris	3	1.04	1.04	**1.44**	1.67	1.67	1.67	0.91	0.91
Letter	26	1.36	**1.24**	5.44	**4.81**	5.54	**5.02**	0.53	**0.58**
Mice	8	1.15	**1.10**	3.12	**2.97**	3.13	**3.11**	0.65	**0.70**
20Newsgroups	20	1.01	1.01	13.80	**13.45**	13.80	**13.78**	0.53	0.53
Pendigits	10	1.32	**1.15**	3.49	**3.28**	3.50	**3.37**	0.67	**0.75**
Poker	10	1.12	**1.11**	**3.23**	3.33	**3.23**	3.33	0.40	0.40
Sensorless	11	1.07	**1.02**	3.99	**2.99**	4.07	**3.84**	0.88	**0.91**
Vowel	11	1.29	**1.24**	3.63	**3.41**	3.64	**3.50**	0.52	**0.56**
Median		1.17	1.15	3.63	3.35	3.72	3.58	0.59	0.61

In terms of cost, `ExShallow`* (`ExShallow` with λ optimized by the procedure described above) beats `ExKMC` in all but two datasets, where both algorithms are tied; in terms of `WAES`, `ExShallow`* beats `ExKMC` in 13 datasets and is beaten by it in 3. Most notably, in the two datasets (`BNG` and `20Newsgroups`) for which `ExKMC` induces less costly partitions than `ExShallow`, `ExShallow`* induces partitions that beat the ones generated by `ExKMC` in both dimensions.

4.5.4 Running times

Table 4.5 presents the average running times for each dataset and algorithm. For all algorithms except `ExBisection` (BIS), running times are an average over 30 iterations and include the time needed to run Lloyd's algorithm and generate the initial reference centers used as input. `ExBisection` ran once for each dataset and does not use representatives from an unexplained partition as input. We note that `RandomThreshold` (RDM) and `ExBisection` are both implemented purely in Python, while their competitors' most time-consuming routines are implemented in either C or Cython, both much faster languages.

Even allowing for this discrepancy, `ExBisection` is frequently much slower than the other algorithms. This is due to the fact that its running time, as mentioned in Section 4.3, is $\mathcal{O}(nkd^2)$. We can see in Table 4.5 that `ExBisection`'s running time is particularly worse than the other algorithms' when d is large – and, by contrast, it performs much faster than them in `Beer`,

Table 4.5: Average running times (in seconds) for each algorithm and dataset. Experiments were performed on 8 484 Intel Core i7-4790 processors @3.60GHz with 32 GB of RAM.

Dataset	k	n	d	SHA	SHA*	BIS	GRD	IMM	KMC	RDM
anuran	10	7195	22	0.75	0.74	0.69	0.73	0.55	0.61	1.48
avila	12	20867	10	1.84	3.81	0.28	1.93	1.60	1.77	3.14
beer	104	1514999	5	730.90	938.72	42.59	731.74	752.97	760.48	735.14
bng	24	1000000	85	1033.53	1669.02	4891.01	1068.85	930.79	956.54	2361.65
cifar10	10	60000	3072	550.68	550.83	19100.18	562.80	416.71	437.85	1464.22
collins	30	1000	19	0.46	0.50	0.22	0.52	0.38	0.41	0.66
covtype	7	581012	54	59.29	58.01	65.42	61.62	42.50	48.76	189.44
digits	10	1797	64	0.44	0.44	0.41	0.47	0.31	0.35	0.99
iris	3	150	4	0.02	0.06	0.00	0.02	0.02	0.02	0.02
letter	26	20000	16	4.82	4.84	1.87	5.55	4.40	4.57	8.07
mice	8	552	77	0.24	0.41	0.46	0.23	0.16	0.17	0.40
newsgroups	20	18846	1069	57.51	311.83	2415.34	107.11	54.56	63.59	241.87
pendigits	10	10992	16	1.01	1.02	0.27	1.04	0.83	0.92	1.88
poker	10	1025010	10	97.32	328.40	17.11	97.30	87.32	95.68	157.18
sensorless	11	58509	48	7.21	7.22	16.95	7.69	4.04	5.00	21.38
vowel	11	990	10	0.20	0.24	0.07	0.20	0.19	0.19	0.25

a dataset with many elements ($n > 10^6$) but few dimensions ($d = 5$).

For the remaining algorithms, the running times for each dataset tend to be comparable, as the bulk of the time is spent on Lloyd's algorithm. `ExShallow`* performs worse than its competitors in some datasets, as it runs through several iterations of Algorithm 5 to find the best value for λ. And `RandomThresholds`'s relatively poor performance is likely due to it being implemented purely in Python, as mentioned above.

5
Separability with minimum size constraints

This chapter presents the results of our research on separability clustering with a minimum size per cluster. Section 5.1 briefly describes the well-know `Single-Linkage` algorithm and shows that saying it maximizes the minimum spacing between clusters, as presented in the literature, is insufficient to describe the partitions it generates. In Section 5.2, we expand the theoretical understanding of `Single-Linkage` by proving that it also maximizes the minimum-spanning-tree cost of the partition, `MST-Cost`, which we introduce in the same section. In Section 5.3, we adapt the `Single-Linkage` algorithm, along with an algorithm for machine scheduling while maximizing the minimum load, to approximate an optimal clustering for the problem of maximizing `Min-Spacing` with a minimum-size guarantee per cluster. This algorithm adaptation is then used in Section 5.4 as a building block for an algorithm that approximates the optimal clustering for the problem of maximizing `MST-Cost` with a minimum-size guarantee per cluster. The results of our experiments on 10 different datasets are presented in Section 5.5.

5.1
The `Single-Linkage` algorithm and the `Min-Spacing` criterium

The `Single-Linkage` algorithm starts with n groups, each of them consisting of a point in $\mathcal{X}$. Then, at each iteration, it merges the two groups with minimum spacing (1-2) into a new group. Thus, by the end of the iteration $n - k$ it obtains a clustering with k groups. In (Kleinberg & Tardos (2006)) it is proved that the `Single-Linkage` obtains a k-clustering with maximum `Min-Spacing` (1-2).

Theorem 5.1 ((Kleinberg & Tardos (2006)), chap 4.7) *The* `Single-Linkage` *algorithm obtains the k-clustering with maximum* `Min-Spacing` (1-2) *for instance* $(\mathcal{X}, \texttt{dist})$.

Figure 5.1, however, shows that Theorem 5.1 does not fully explain the behavior of `Single-Linkage`, as partitions very different than those it generates may also maximize `Min-Spacing`. The subfigure to the right shows a clustering built by `Single-Linkage`; the subfigure to the left shows a

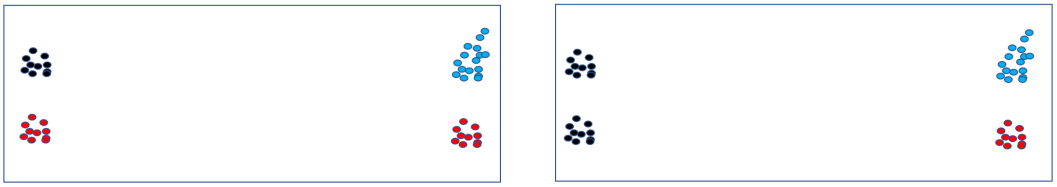


Figure 5.1: Two partitions with 3 groups (defined by colors) that maximize `Min-Spacing`. The clustering built by `Single-Linkage` is to the left; a different clustering, that does not maximize `MST-Cost`, is to the right.

very different clustering, but one that also maximizes the minimum spacing – showing that this condition alone is insufficient to properly characterize `Single-Linkage`'s behavior.

To analyze our algorithms we make use of well-known properties of minimum spanning trees such as the cut property and the cycle property, reproduced below.

Theorem 5.2 (Cut Property.) *Let $G = (V, E)$ be a graph with distinct weights on its edges. Let $S \subset V$ be a non-empty cut in G. If e is the edge with minimum cost among those that have one endpoint in S and the other one in $V \setminus S$, then e belongs to every minimum spanning tree for G.*

Proof. See Property 4.17 in (Kleinberg & Tardos (2006)). ■

Theorem 5.3 (Cycle Property.) *Let $G = (V, E)$ be a graph with distinct weights on its edges and let C be a cycle in G. Then, the edge with the largest weight in C does not belong to any minimum spanning tree for G.*

Proof. See Property 4.20 in (Kleinberg & Tardos (2006)). ■

For ease of presentation, we assume that all values of `dist` are distinct. We note, however, that our results hold if this assumption is dropped.

5.2 Relating `Min-Spacing` and `MST-Cost` criteria

Let $G_{\mathcal{P}}$ be the complete weighted graph induced by $\mathcal{P}$ where each node represents a group and each edge between nodes $P_1, P_2 \in \mathcal{P}$ has weight equivalent to `Min-Spacing` (1-2), where $x \in P_1$ and $y \in P_2$. In the *minimum-spanning-tree clustering problem*, our goal is to maximize

$$\text{MST-Cost}(\mathcal{P}) = \text{MST}(G_{\mathcal{P}}), \tag{5-1}$$

where $\text{MST}(G)$ is the cost of the minimum spanning tree of G.

`Single-Linkage` and minimum spanning trees are closely related: the former can be seen as Kruskal's algorithm for building minimum spanning

trees with an early stopping rule. We prove in this section that, more than maximizing `Min-Spacing`, `Single-Linkage` maximizes `MST-Cost`, which implies the maximization of the former. These results, presented below, are a consequence of Lemma 10, that generalizes the result of Theorem 5.1.

In what follows, $\mathcal{C}_{SL}$ is a k-clustering obtained by `Single-Linkage` for instance $I = (\mathcal{X}, \texttt{dist}, k)$ and T_{SL} is a minimum spanning tree for $G_{\mathcal{C}_{SL}}$. Moreover, w_i^{SL} is the weight of the i-th smallest weight of T_{SL}.

Lemma 10 *Let $\mathcal{C}$ be a k-clustering for I and let w_i be the weight of the i-th smallest weight in a minimum spanning tree T for the graph $G_{\mathcal{C}}$. Then, $w_i^{SL} \geq w_i$.*

To prove Lemma 10 we will use the following characterization of minimum spanning trees, whose correctness follows directly from Theorem 5.3:

Theorem 5.4 (Minimum spanning tree conditions.) *Let $G = (V, E)$ be a weighted graph. A spanning tree T for G is a minimum spanning tree if and only if, for each edge $e = uv \in E$, the weight $w(e)$ of e satisfies $w(e) \geq w(e')$ for every edge e' in the path that connects u to v in T.*

The following proposition will also be useful. Recall that $\texttt{spacing}(C_i, C_j)$ is defined (1-3) as the minimum distance between two points from clusters C_i, C_j.

Proposition 5.5 *Let $\mathcal{C}'$ be a k-clustering for instance I and let T' be a minimum spanning tree for $G_{\mathcal{C}'}$. Moreover, let C'_i and C'_j be groups of $\mathcal{C}'$ such that $\texttt{spacing}(C'_i, C'_j) = \texttt{Min-Spacing}(\mathcal{C}')$. Then, the tree T^a that results from the contraction of the nodes C'_i and C'_j in T' is a minimum spanning tree for $G_{\mathcal{C}^a}$, where $\mathcal{C}^a$ is the $(k-1)$-clustering obtained from $\mathcal{C}'$ by merging C'_i and C'_j.*

Proof. We show that T^a satisfies the conditions of Theorem 5.4 when $G = G_{\mathcal{C}^a}$. For that, we will use the fact that T' satisfies the conditions of Theorem 5.4 when $G = G_{\mathcal{C}'}$.

Let x and y be nodes of T^a. For the sake of contradiction, we assume that edge xy does not satisfy the conditions of Theorem 5.4 when $G = G_{\mathcal{C}^a}$ and $T = T^a$. Let w^* be the weight of edge xy and let e be an edge in the path that connects x to y in T^a such that $w(e) > w^*$.

We have two cases:

Case 1) $x \neq C'_i \cup C'_j$ and $y \neq C'_i \cup C'_j$. Then, e is also an edge in the path that connects x to y in T'. This implies that xy does not satisfy the required conditions when $G = G_{\mathcal{C}'}$ and $T = T'$, which is a contradiction.

Case 2) $x = C'_i \cup C'_j$ or $y \neq C'_i \cup C'_j$. Without loss of generality, assume $x = C'_i \cup C'_j$. Let w'_i and w'_j be, respectively, the weights of the edges (y, C'_i) and (y, C'_j) in $G_{\mathcal{C}'}$.

We have that $w^* = \min\{w'_i, w'_j\}$. Without loss of generality, assume $w^* = w'_i$. Then, e is also an edge in the path that connects y to C'_i in T'. Again, this implies that xy does not satisfy the required conditions when $G = G_{\mathcal{C}'}$ and $T = T'$, which is a contradiction. ■

Proof of Lemma 10. It follows from Proposition 5.5 that the tree T^{i-1} that is obtained by contracting the $i-1$ cheapest edges of T is a minimum spanning tree for $G_{\mathcal{C}^{i-1}}$, where $\mathcal{C}^{i-1}$ is a clustering for instance I that contains $(k-(i-1))$ groups. The cheapest edge of T^{i-1} is exactly the i-th cheapest edge of T. Thus, $\texttt{Min-Spacing}(\mathcal{C}^{i-1}) = w_i$.

Similarly, w_i^{SL} is exactly the `Min-Spacing` of a clustering with $(k-(i-1)$ groups that is obtained by `Single-Linkage` for instance I. Thus, it follows from Theorem 5.1 that $w_i^{SL} \geq w_i$. ■

From Lemma 10 it is straightforward to prove that `Single-Linkage` maximizes `MST-Cost`, and that this is a stronger characterization than maximizing `Min-Spacing`:

Theorem 5.6 *The clustering $\mathcal{C}_{SL}$ returned by* `Single-Linkage` *for instance $(\mathcal{X}, \texttt{dist}, k)$ maximizes the* `MST-Cost` *criterion.*

Proof. Let $\mathcal{C}$ be a k-clustering for $(\mathcal{X}, \texttt{dist}, k)$ and let w_i be the weight of the i-th cheapest edge of the minimum spanning tree for $G_{\mathcal{C}}$. Since $w_i^{SL} \geq w_i$ for $i = 1, \ldots, k-1$, we have that

$$\texttt{MST-Cost}(\mathcal{C}_{SL}) = \sum_{i=1}^{k-1} w_i^{SL} \geq \sum_{i=1}^{k-1} w_i = \texttt{MST-Cost}(\mathcal{C}).$$

■

Theorem 5.7 *Let $\mathcal{C}^*$ be a clustering that maximizes the* `MST-Cost` *criterion for instance $(\mathcal{X}, \texttt{dist}, k)$. Then, it also maximizes* `Min-Spacing` *for this same instance.*

Proof. Let us assume that $\mathcal{C}^*$ maximizes the `MST-Cost` criterion but it does not maximize the `Min-Spacing` criterion. Thus, $w_1^{SL} > w_1^*$, where w_1^* is the minimum spacing of $\mathcal{C}^*$. It follows from the previous lemma that

$$\texttt{MST-Cost}(\mathcal{C}_{SL}) = \sum_{i=1}^{k-1} w_i^{SL} > \sum_{i=1}^{k-1} w_i^* = \texttt{MST-Cost}(\mathcal{C}^*),$$

which contradicts the assumption that $\mathcal{C}^*$ maximizes the `MST-Cost` criterion. ■

The next example (in the spirit of Figure 5.1) shows that a partition that maximizes `Min-Spacing` may have a poor result in terms of `MST-Cost`.

Example 1 *Let D be a positive number much larger than k. Moreover, let $[t]$ be the set of the t first positive integers and $S = \{(D \cdot i, j) | i, j \in [k-1]\} \cup (D, k)$ be a set of $(k-1)^2 + 1$ points in $\mathbb{R}^2$.*

`Single-Linkage` *builds a k-clustering with* `Min-Spacing` $= 1$ *and* `MST-Cost` $= 1 + (k-2)D$ *for S.*

However, the k-clustering $(C_1, \ldots, C_k)$, where $C_j = \{(D \cdot i, j) | i = 1, \ldots, k-1\}$, for $j < k$ and $C_k = \{(D, k)\}$ has `Min-Spacing` 1 *and* `MST-Cost`$= (k-1)$.

5.3 AlgoMinSp: maximizing the minimum spacing with minimum-size approximation guarantees

We start with a polynomial-time approximation scheme for the `Min-Spacing` criterion. Our algorithm, `AlgoMinSp` (Algorithm 12), combines `Single-Linkage` with `MaxMinSched` (Csirik et al. (1992), Woeginger (1997)), an algorithm for the *max-min scheduling problem* with identical machines, described below. For $k' > k$, `AlgoMinSp` allows merging clusters from a (k')-partition generated by `Single-Linkage` in a way to obtain a $(k, (1-\epsilon)L)$-partition with a minimum distance equal to or greater than the maximum minimum distance of a (k, L)-partition, where ϵ is the approximation factor of `MaxMinSched`.

The max-min scheduling problem can be described as follows: given m machines and a set of n jobs with processing times $p_1, \ldots, p_n$, find an assignment of jobs to the machines so that the load of the machine with minimum load is maximized. This problem admits a polynomial-time approximation scheme (Woeginger (1997)).

Let `MaxMinSched`(P, k, ϵ) be a routine that implements this scheme. It receives as input a parameter $\epsilon > 0$, an integer k and a list of numbers P (corresponding to processing times), and returns a partition of P into k lists (corresponding to machines) such that the sum of the numbers of the list with minimum sum is at least $(1-\epsilon)OPT$, where OPT is the minimum load of a machine in an optimal solution of for the max-min scheduling problem when the list of processing times is P and the number of machines is k.

Algorithm 12, as proved in the next theorem, obtains a $(k, L(1-\epsilon))$-clustering whose `Min-Spacing` is at least the `Min-Spacing` of an optimal

(k, L)-clustering. For that, it looks for the largest integer t for which the clustering $\mathcal{A}_t$ obtained by executing t steps of `Single-Linkage` and then combining the resulting groups into k groups (via `MaxMinSched`) is a $(k, L(1-\epsilon))$-clustering. We assume that `MaxMinSched`, in addition of returning the partition of the sizes, also returns the group associated to each size.

Algorithm 12 `AlgoMinSp`($\mathcal{X}$; `dist`; k; $\epsilon > 0$;L)

$t \leftarrow n - k$
while $t \geq 0$ **do**
    Run t merging steps of the `Single-Linkage` for input $\mathcal{X}$
    Let $C_1, \ldots, C_{n-t}$ the groups obtained by the end of the t steps
    $P \leftarrow (|C_1|, \ldots, |C_{n-t}|)$
    $A_t \leftarrow$ `MaxMinSched`(P, k, ϵ)
    if the smallest group in A_t has size greater than or equal to $L(1-\epsilon)$ **then**
        Return A_t
    else
        $t \leftarrow t - 1$

Theorem 5.8 *Fix $\epsilon > 0$. The clustering $\mathcal{A}_t$ returned by Algorithm 12 is a $(k, (1-\epsilon)L)$-clustering that satisfies Min-Spacing($\mathcal{A}_t$) $\geq$ Min-Spacing($\mathcal{C}^*$), where $\mathcal{C}^*$ is the (k, L)-clustering with maximum* `Min-Spacing`.

Proof. By design, $\mathcal{A}_t$ has k groups with at least $L(1-\epsilon)$ points in each of them.

For the sake of contradiction, assume that Min-Spacing($\mathcal{A}_t$) $<$ Min-Spacing($\mathcal{C}^*$).

Let $\mathcal{C} = (C_1, \ldots, C_{n-t})$ be the list of $n-t$ groups obtained after t merging steps of `Single-Linkage` are performed. Without loss of generality, assume that C_1 and C_2 are the two groups with a minimum spacing in this list, so that Min-Spacing($\mathcal{C}$) = `spacing`(C_1, C_2). Since $\mathcal{A}_t$ is a k-clustering that is obtained by merging groups in $\mathcal{C}$ we have Min-Spacing($\mathcal{A}_t$) $\geq$ Min-Spacing($\mathcal{C}$) = `spacing`(C_1, C_2).

For $i = 1, \ldots, n-t$ we have that $C_i \subseteq C$ for some $C \in \mathcal{C}^*$, otherwise we would have Min-Spacing($\mathcal{A}_t$) $\geq$ Min-Spacing($\mathcal{C}^*$). In addition, we must have $C_1 \cup C_2 \subseteq C$ for some $C \in \mathcal{C}^*$, otherwise, again, we would have Min-Spacing($\mathcal{A}_t$) $\geq$ Min-Spacing($\mathcal{C}^*$).

We can conclude that there is a feasible solution with minimum load not smaller than L for the max-min scheduling problem with processing times $P' = (|C_1 \cup C_2|, |C_3|, \ldots, |C_{n-t}|)$ and k machines. Thus, by running $t+1$ steps of `Single-Linkage` followed by `MaxMinSched`(P', k, ϵ), we would get a

k-clustering whose smallest group has at least $L(1-\epsilon)$ points. This implies that the algorithm would have stopped after performing $t+1$ merging steps, which is a contradiction. ■

5.3.1 Implementation details

As presented, Algorithm 12 may run `Single-Linkage` $n-k$ times, which is potentially quite expensive. But rather than run t merging steps of `Single-Linkage` for every t being tested, as we do in line 3 of Algorithm 12, we can run `Single-Linkage` once from start to finish – i.e., from n singletons to 2 clusters – and then perform a binary search to find the smallest value of t such that a t-partition of $\mathcal{X}$ can be agglomerated into k clusters with the desired minimum size. By traversing the full dendrogram induced by `Single-Linkage` and keeping track of the number of elements per cluster, we can find in $\mathcal{O}(n)$ if a given t is feasible, so it takes $\mathcal{O}(n \log n)$ to run the binary search and find the first t that can be used. In the `MaxMinSched` step (line 6), A_t starts with k empty clusters, and the clusters from the t-partition are iteratively merged to the smallest cluster in A_t in ascending order of size.

5.3.2 Approximation limits for minimum-spacing clustering with minimum size

The next theorem shows that Algorithm 12 has essentially tight guarantees under the hypothesis that $P \neq NP$.

Theorem 5.9 *Unless $P = NP$, for any $\alpha = poly(n)$, the problem of finding the (k, L)-clustering that maximizes the* `Min-Spacing` *criterion does not admit a $(1, \frac{1}{\alpha})$-approximation.*

Proof. We make a reduction from the $(T/4, T/2)$-restricted 3-PARTITION problem. Given a multiset $S = \{s_1, \ldots, s_{3m}\}$ of positive integers that satisfies $\sum_i s_i = mT$, the 3-PARTITION problem consists of deciding whether or not there exists a partition of S into m triples such that the sum of the numbers in each one is equal to T. In the $(T/4, T/2)$-restricted 3-PARTITION problem, there is an additional requirement that each number of S should be in the interval $(T/4, T/2)$. This problem is strongly NP-COMPLETE (Garey & Johnson (2011)).

The instance $I = (X, k, L, \texttt{dist})$ for our clustering problem is built as follows: we set $L = T$, $k = m$; for $i = 1, \ldots, 3m$, let $\mathcal{X}_i$ be a set with s_i points so that the distance between points in the same group $\mathcal{X}_i$ is 1 while the distance between points in different groups is $\alpha + 1$. We set $\mathcal{X} = \mathcal{X}_1 \cup \ldots \cup \mathcal{X}_{3m}$. Note

that we are employing a pseudo-polynomial reduction, but this is fine as the $(T/4, T/2)$-restricted 3-PARTITION problem is strongly NP-COMPLETE.

Assume that there is an $(1, 1/\alpha)$-approximation for our problem and let $\mathcal{C}$ be the clustering returned by this algorithm for instance I. We argue that the answer to the 3-PARTITION problem is YES if and only if Min-Spacing$(\mathcal{C}) = \alpha + 1$.

First, we show that, if the answer is YES, there is a k-clustering $\mathcal{C}^*$ for I with Min-Spacing$(\mathcal{C}^*) = \alpha + 1$. In fact, let $S_1, \ldots, S_m$ be a solution of the 3-PARTITION problem and let $\{s_{i_1}, s_{i_2}, s_{i_3}\}$ be the numbers in S_i. Let $\mathcal{C}^*$ be a k-clustering where the i-th group is comprised by all points in $\mathcal{X}_{i_1} \cup \mathcal{X}_{i_2} \cup \mathcal{X}_{i_3}$. Clearly, each group has L points and the `Min-Spacing` of this clustering is $\alpha + 1$. Since our algorithm is a $(1, 1/\alpha)$-approximation it returns a clustering $\mathcal{C}$ with `Min-Spacing`$(\mathcal{C}) \geq (\alpha+1)/\alpha$. Since the `Min-Spacing` of any clustering for instance I is either 1 or $\alpha + 1$ we have that `Min-Spacing`$(\mathcal{C})$=$\alpha + 1$.

On the other hand, if the clustering has `Min-Spacing` $\alpha + 1$ then all points in $\mathcal{X}_i$, for each i, must be in the same group. Moreover, due to the restriction that every $|\mathcal{X}_i| = s_i \in (L/4, L/2)$, we should have exactly 3 $\mathcal{X}_i$'s in each of the m groups. Thus, the answer is YES. ■

5.4 ConstrainedMaxMST: maximizing `MST-Cost` with minimum-size approximation guarantees

Our next algorithm, ConstrainedMaxMST, runs AlgoMinSP for $2, \ldots, k$ clusters, in each iteration separating the obtained clusters to achieve a k-partition, and returns the best of these $k - 1$ possibilities in terms of `MST-Cost`. The obtained approximation guarantee is

$$\left(\frac{\rho(1-\epsilon)}{2}, \frac{1}{H_{k-1}}\right),$$

where $\rho := \min\{\frac{n/k}{L}, 2\}$ and $H_{k-1} = \sum_{i=1}^{k-1} \frac{1}{i}$ is the $(k-1)$-th harmonic number. Note that H_{k-1} is $\Theta(\log k)$.

For each $\ell = 2, \ldots, k$, the algorithm calls `AlgoMinSp` (Algorithm 12) to build a clustering $\mathcal{A}'_\ell$ with ℓ groups and then it transforms $\mathcal{A}'_\ell$ (lines 5-13) into a clustering $\mathcal{A}_\ell$ with k groups. In the end, it returns the clustering, among the $k - 1$ considered, with maximum `MST-Cost`.

Algorithm 13 `Constrained-MaxMST`$(\mathcal{X}; \texttt{dist}; k; L; \epsilon)$

1: **for** $\ell = 2, \ldots, k$ **do**
2: $\quad\mathcal{A}'_\ell \leftarrow \texttt{AlgoMinSp}(\mathcal{X},\texttt{dist},\ell,\epsilon,L)$
3: $\quad\texttt{NonVisited} \leftarrow \mathcal{A}'_\ell$
4: $\quad\mathcal{A}_\ell \leftarrow \emptyset$
5: $\quad$**for** each A' in $\mathcal{A}'_\ell$, iterating from the largest group to the smallest **do**
6: $\quad\quad\texttt{NonVisited} \leftarrow \texttt{NonVisited}$ - A'
7: $\quad\quad\texttt{SplitNumber} \leftarrow \left\lfloor \frac{2|A'|}{\rho(1-\epsilon)L} \right\rfloor$
8: $\quad\quad$**if** $|\mathcal{A}_\ell| + |\texttt{NonVisited}| + \texttt{SplitNumber} < k$ **then**
9: $\quad\quad\quad$Split A' into `SplitNumber` as balanced as possible groups and add them to $\mathcal{A}_\ell$
10: $\quad\quad$**else**
11: $\quad\quad\quad$Split A' into $k - |\mathcal{A}_\ell| - |\texttt{NonVisited}|$ as balanced as possible groups; add them to $\mathcal{A}_\ell$
12: $\quad\quad\quad$Add all groups in `NonVisited` to $\mathcal{A}_\ell$
13: $\quad\quad\quad$Break
14: **Return** the clustering A_ℓ, among the $k-1$ obtained, that has the maximum `MST-Cost`

Note that, while scanning the groups in $\mathcal{A}'_\ell$ by non-increasing size order is not necessary to establish the guarantees presented below, this tends to prevent the formation of groups with sizes smaller than L.

Lemma 11 *Fix $\epsilon > 0$. Thus, for each ℓ, every group in $\mathcal{A}_\ell$ has at least $\left\lfloor \frac{\rho(1-\epsilon)L}{2} \right\rfloor$ points.*

Proof. The groups that are added to $\mathcal{A}_\ell$ in line 12 have at least $(1-\epsilon)L$ points while the number of points of those that are added at either line 11 or 9 is at least

$$\left\lfloor \frac{|A'|}{\lfloor 2|A'|/\rho(1-\epsilon)L \rfloor} \right\rfloor \geq \left\lfloor \frac{\rho(1-\epsilon)L}{2} \right\rfloor.$$

Moreover, if the **For** is not interrupted by the **Break** command, the total number of groups in $\mathcal{A}_\ell$ is

$$\sum_{A' \in \mathcal{A}'_\ell} \left\lfloor \frac{2|A'|}{\rho(1-\epsilon)L} \right\rfloor \geq \sum_{A' \in \mathcal{A}'_\ell} \frac{2|A'|}{\rho(1-\epsilon)L} - \ell \geq \frac{2n}{\rho(1-\epsilon)L} - k \geq \frac{2k}{(1-\epsilon)} - k \geq k.$$

Since the **For** is interrupted as soon as k groups can be obtained, $\mathcal{A}_\ell$ has k groups. ■

For the next results, we use $\mathcal{C}^*$ to denote the (k, L)-clustering with maximum `MST-Cost` and w_i^* to denote the cost of the i-th cheapest edge in the

minimum spanning tree for $G_{\mathcal{C}^*}$. Our first lemma can be seen as a generalization of Theorem 5.8.

Lemma 12 *For each ℓ,* `Min-Spacing`$(\mathcal{A}'_\ell) \geq w^*_{k-\ell+1}$.

Proof. Let T^* be the minimum spanning tree for $G_{\mathcal{C}^*}$. If we remove the $\ell - 1$ most expensive edges of T^* we obtain a forest F with ℓ connected components. The clustering $\mathcal{C}^*_\ell$ comprised by ℓ groups in which the i-th group corresponds to the i-th connected component of F is an (ℓ, L)-clustering and Min-Spacing$(\mathcal{C}^*_\ell) = w^*_{k-\ell+1}$.

Let OPT be the Min-Spacing of the (ℓ, L)-clustering with maximum Min-Spacing. Thus, by Theorem 5.8 Min-Spacing$(\mathcal{A}'_\ell) \geq OPT \geq$ Min-Spacing$(\mathcal{C}^*_\ell) = w^*_{k-\ell+1}$. ■

A simple consequence of the previous lemma is that the `MST-Cost` of clustering $\mathcal{A}'_\ell$ is at least $(\ell - 1) \cdot w^*_{k-\ell+1}$. The next lemma shows that this bound also holds for the clustering $\mathcal{A}_\ell$. The proof consists of showing that each edge of a minimum spanning tree for $\mathcal{A}'_\ell$ is also an edge of a minimum spanning tree for $\mathcal{A}_\ell$.

Lemma 13 *For each $\ell = 2, \ldots, k$ we have* `MST-Cost`$(\mathcal{A}_\ell) \geq (\ell - 1) \cdot w^*_{k-\ell+1}$.

Proof. Let T_ℓ and T'_ℓ be, respectively, the minimum spanning tree for $G_{\mathcal{A}_\ell}$ and $G_{\mathcal{A}'_\ell}$. By the previous lemma, each of the $(\ell - 1)$ edges of T'_ℓ has cost at least Min-Spacing$(\mathcal{A}'_\ell) \geq w^*_{k-\ell+1}$. Thus, to establish the result, it is enough to argue that each edge of T'_ℓ also belongs to T_ℓ.

We say that a group $A \in \mathcal{A}_\ell$ is generated from a group $A' \in \mathcal{A}'_\ell$ if $A = A'$ or A is one of the balanced groups that is generated when A' is split in the internal **For** of Algorithm 13. We say that a vertex x in $G_{\mathcal{A}_\ell}$ is generated from a vertex x' in $G_{\mathcal{A}'_\ell}$ if the group corresponding to x is generated by the corresponding to x'.

Let $e' = u'v'$ be an edge in T'_ℓ and let S' be a cut in graph $G_{\mathcal{A}'_\ell}$ whose vertices are those from the connected component of $T'_\ell \setminus e'$ that includes u'. We define the cut S of $G_{\mathcal{A}_\ell}$ as follows: $S = \{x \in G_{\mathcal{A}_\ell} | x$ is generated from some $x' \in S'\}$.

Let u and v be vertices generated from u' and v', respectively, that satisfy $w(uv) = w(u'v')$. It is enough to show that uv is the cheapest edge that crosses S, since it follows from Theorem 5.2 that this implies $uv \in T_\ell$.

We prove it by contradiction. Let us assume that there is another edge $f = yz$ that crosses S and has weight smaller than $w(uv)$. Let y' and z' be vertices in $G_{\mathcal{A}'_\ell}$ that generate y and z, respectively, and let $f' = y'z'$. Thus, $w(f') \leq w(f) < w(uv) = w(u'v') = w(e')$. However, this contradicts Theorem

5.3, because it implies that the edge with the largest weight in the cycle of $G_{\mathcal{A}'_\ell}$ comprised by edge f' and the path in T'_ℓ the connects y' to z' belongs to the T'_ℓ. ■

The next theorem is the main result of this section.

Theorem 5.10 *Fix $\epsilon > 0$. Algorithm 13 is a $(\frac{(1-\epsilon)\rho}{2}, \frac{1}{H_{k-1}})$-approximation for the problem of finding the (k, L)–clustering that maximizes the* `MST-Cost` *criterion.*

Proof. Let $\mathcal{C}$ be the clustering returned by Algorithm 13. Lemma 11 guarantees that $\mathcal{C}$ is a $(k, \lfloor\frac{(1-\epsilon)\rho L}{2}\rfloor)$-clustering.

Thus, we just need to argue about MST-Cost($\mathcal{C}$). We have that

$$\text{MST-Cost}(\mathcal{C}^*) = \sum_{i=2}^{k} w^*_{k-i+1}$$

and, due to Lemma 13, MST-Cost($\mathcal{C}$) $\geq \max\{(\ell-1)\cdot w^*_{k-\ell+1} | 2 \leq \ell \leq k\}$.

Let $\tilde{\ell}$ be the value ℓ that maximizes $(\ell-1)\cdot w^*_{k-\ell+1}$. It follows that $w^*_{k-i+1} \leq ((\tilde{\ell}-1)/(i-1))w^*_{k-\tilde{\ell}+1}$. for $i = 2, \ldots, k$. Thus,

$$\frac{\text{MST-Cost}(\mathcal{C}^*)}{\text{MST-Cost}(\mathcal{C})} \leq \frac{\sum_{i=2}^{k} w^*_{k-i+1}}{(\tilde{\ell}-1)\cdot w^*_{k-\tilde{\ell}+1}} \leq \frac{(\tilde{\ell}-1)\cdot w^*_{k-\tilde{\ell}+1}\cdot\sum_{i=2}^{k}\frac{1}{i-1}}{(\tilde{\ell}-1)\cdot w^*_{k-\tilde{\ell}+1}} = H_{k-1}.$$

■

5.4.1 Approximation limits for `MST-Cost` clustering with minimum size

We show below that the optimization of `MST-Cost` is APX-HARD (for fixed k) when a hard constraint on the number of points per group is imposed. The proof is very similar to that of Theorem 5.9.

Theorem 5.11 *Unless $P = NP$, for any $\alpha = poly(n)$, there is no $(1, \frac{k-2}{k-1} + \frac{1}{\alpha(k-1)})$-approximation for the problem of finding the (k, L)-clustering that maximizes the* `MST-Cost` *criterion.*

Proof. As in the proof of Theorem 5.9, we make a reduction from the $(T/4, T/2)$-restricted 3-PARTITION problem. The instance $I = (X, k, L, \texttt{dist})$ for our clustering problem is built as follows: we set $L = T$, $k = m$; for $i = 1, \ldots, 3m$ let $\mathcal{X}_i$ be a set with s_i points so that the distance between points in the same group $\mathcal{X}_i$ is $1/2$ while the distance between points in different groups is α. We set $\mathcal{X} = \mathcal{X}_1 \cup \ldots \cup \mathcal{X}_{3m}$.

Let us assume that there is an $(1, \frac{k-2}{k-1} + \frac{1}{\alpha(k-1)})$-approximation for our problem and let $\mathcal{C}$ be the clustering returned by this algorithm for instance I.

We argue that the answer to the 3-PARTITION problem is YES if and only if MST-Cost$(\mathcal{C}) = (k-1)\alpha$.

First, we show that if the answer is YES, there is a k-clustering $\mathcal{C}^*$ for I with `MST-Cost`$(\mathcal{C}^*) = (k-1)\alpha$. Let $S_1, \ldots, S_m$ be a solution of the 3-PARTITION problem and let $\{s_{i_1}, s_{i_2}, s_{i_3}\}$ the numbers in S_i. Let $\mathcal{C}^*$ be a k-clustering where the ith group is comprised by all points in $\mathcal{X}_{i_1} \cup \mathcal{X}_{i_2} \cup \mathcal{X}_{i_3}$. Clearly, each group has L points and the `MST-Cost` of this clustering is $(k-1)\alpha$. Since our algorithm is a $(1, \frac{k-2}{k-1} + \frac{1}{\alpha(k-1)})$-approximation, it returns a clustering $\mathcal{C}$ with `MST-Cost`$(\mathcal{C}) > (k-2)\alpha + 1$. Since the `MST-Cost` of any clustering for instance I is either $(k-1)\alpha$ or at most $(k-2)\alpha + 1/2$, we have that `MST-Cost`$(\mathcal{C})$=$(k-1)\alpha$.

On the other hand, if the clustering has Min-Sp $(k-1)\alpha$ then all points in $\mathcal{X}_i$, for each i, must be in the same group. Moreover, due to the restriction that every $|\mathcal{X}_i| = s_i \in (L/4, L/2)$, we should have exactly 3 $\mathcal{X}_i$'s in each of the m groups. Thus, the answer is YES. ■

5.5 Experiments

To evaluate the performance of Algorithms 12 and 13, we ran experiments with 10 different datasets, comparing the results with those of `Single-Linkage` and of the traditional k-means algorithm from (Lloyd (1982)) with a ++ initialization (Arthur & Vassilvitskii (2006)). For the implementation of routine `MaxMinSched`, employed by Algorithm 12, we used the Longest Processing Time rule (Csirik et al. (1992)), which has the advantage of being fast while guaranteeing a 3/4 approximation for the max-min scheduling problem. The code for running the algorithms can be found at `https://github.com/lmurtinho/SizeConstrainedSpacing`.

Our first experiment investigates the size of the groups produced by `Single-Linkage` for the 10 datasets, whose dimensions can be found in the first two columns of Table 5.1. Figure 1.3 shows the proportion of singletons for each dataset with the growth of k. For all datasets but `Vowel` and `Mice` the majority of groups are singletons, even for small values of k. This undesirable behavior motivates our constraint on the minimum size of a group.

In our second experiment, we compare the values of `Min-Spacing` and `MST-Cost` achieved by our algorithms with those of k-means. While k-means is not a particularly strong competitor in the sense that it was not designed to optimize our criteria, it is a very popular choice among practitioners. Moreover, for datasets with well-separated groups, the minimization of the squared sum of errors (pursued by k-means) should also imply the maximization of inter-

Table 5.1: `Min-Spacing` and `MST-Cost` for the different methods and datasets.

	Dimensions		`Min-Spacing`			`MST-Cost`		
	n	k	Algo 12	Algo 13	k-means	Algo 12	Algo 13	k-means
anuran	7,195	10	**0.19**	0.09	0.05	1.71	**1.87**	1.01
avila	20,867	12	**0.07**	0.04	0	0.77	**0.81**	0.66
collins	1,000	30	**0.42**	**0.42**	0.22	**12.42**	**12.42**	8.58
digits	1,797	10	**19.74**	**19.74**	13.79	**178.22**	**178.22**	145.13
letter	20,000	26	**0.2**	0.11	0.07	4.98	**5.67**	1.98
mice	552	8	**0.79**	**0.79**	0.24	**5.66**	**5.66**	2.37
newsgroups	18,846	20	**1**	1	0.17	19	**19**	8.4
pendigits	10,992	10	**23.89**	9.08	8.31	215.11	**217.01**	119.85
sensorless	58,509	11	**0.13**	0.08	0.03	1.31	**1.36**	1.29
vowel	990	11	**0.49**	**0.49**	0.11	**4.94**	**4.94**	1.84

group criteria.

Table 5.1 presents the results of this experiment. The values chosen for k are the numbers of classes (dependent variables) in the datasets, while for the `dist` function we employed the Euclidean distance. The values associated with the criteria are averages of 10 executions, with each execution corresponding to a different seed provided to k-means. To set the value of L for the i-th execution of our algorithms, we take the size of the smallest group generated by k-means for this execution and multiply it by 4/3. This way, we guarantee that the size of the smallest group produced by our methods, for each execution, is not smaller than that of k-means, which makes the comparison among the optimization criteria fairer.

With respect to the `Min-Spacing` criterion, Algorithm 12 is at least as good as Algorithm 13 for every dataset (being superior on 6) and both Algorithm 12 and 13 outperform k-means on all datasets. On the other hand, with respect to the `MST-Cost` criterion, Algorithm 13 is at least as good as Algorithm 12 for every dataset (being better on 6) and, again, both algorithms outperform k-means for all datasets.

Table 5.2 shows the running time of our algorithms for the datasets that consumed more time. Experiments were run in an Ubuntu 20.04.5 LTS with 40 cores and 115 GB RAM.

We observe that the overhead introduced by Algorithm 12 with respect to `Single-Linkage` is negligible while Algorithm 13, as expected, is more costly. In Section 5.5.3, we show that a strategy that only considers values of ℓ that can be written as $\lceil k/2^t \rceil$, for $t = 0, \ldots, \lfloor \log k \rfloor$, in the first loop of Algorithm 13 provides a significant gain of running time while incurring a small loss in the `MST-Cost`. We note that the $\log k$ bound of Theorem 5.10 is still valid for

Table 5.2: Running time (in seconds) of `Single-Linkage` and our methods. Experiments were run in an Ubuntu 20.04.5 LTS with 40 cores and 115 GB RAM.

Dataset	`Single-Linkage`	Algo 1	Algo 2
sensorless	93.8	99.4	701.4
newsgroups	274.2	276.4	440.4
letter	4.6	5.9	116.4
avila	4.0	5.5	62.5
pendigits	1.4	2.1	16.0

this strategy.

5.5.1 `MST-Cost`: comparison between empirical results and upper bound of Algorithm 13

Algorithm 13 can in practice obtain partitions that are much closer to the optimal `MST-Cost` than the bound guaranteed by Theorem 5.10. In Table 5.3, we present, for each dataset: the average `MST-Cost` obtained by Algorithm 13; the upper bound on `MST-Cost` given by the sum of `Min-Spacing` for all partitions found by an execution of the algorithm; the approximation ratio of Algorithm 13, given by its `MST-Cost` divided by the upper bound; and the theoretical approximation ratio $1/H_{k-1}$ from Theorem 5.10.

For all 10 datasets, Algorithm 13 performs significantly better than its theoretical approximation ratio. The smallest gap between theoretical and empirical result occurs for `avila` dataset, in which the algorithm is 21 percentage points closer to the optimal `MST-Cost` than Theorem 5.10 guarantees; on the other extreme, for dataset `newsgroups`, it actually achieves the best possible `MST-Cost`. These results increase our confidence that Algorithm 13 is a good option for finding well-separated groups.

5.5.2 Average size of smallest clusters

Table 5.4 presents the average size of the smallest group generated by Algorithms 12 and 13 as well as k-means. Values tend to be close across all algorithms, and for all iterations of the experiments the smallest group returned by Algorithms 12 and 13 is at least as large as the smallest group from the corresponding k-means clustering (recall that L is set as $4s/3$, where s is the size of the smallest group produced by k-means). In particular, thanks to the

Table 5.3: Comparison between the `MST-Cost` of Algorithm 13 and the upper bound given by Lemma 12, given by the sum of `Min-Spacing` for all partitions found by an execution of the algorithm. $1/H_{k-1}$, where H_{k-1} is the $(k-1)$-th harmonic number, is the theoretical lower bound from Lemma 13.

	k	`MST-Cost`	$\sum_{\ell=2}^{k}$ `Min-Spacing`$(\mathcal{A}'_\ell)$	Approximation Ratio	$1/H_{k-1}$
anuran	10	1.87	2.42	0.77	0.35
avila	12	0.81	1.48	0.55	0.34
collins	30	12.42	13.81	0.9	0.33
digits	10	178.22	201.47	0.88	0.32
letter	26	5.67	5.76	0.98	0.31
mice	8	5.66	7.12	0.79	0.31
newsgroups	20	19	19	1	0.3
pendigits	10	217.01	303.37	0.72	0.3
sensorless	11	1.36	2.23	0.61	0.29
vowel	11	4.94	5.57	0.89	0.29

rule of iterating from the largest cluster to the smallest when building our k-clustering from an ℓ-clustering (line 5), the theoretical possibility that the smallest group induced by Algorithm 13 is 1/2 of the desired size does not appear to happen in practice.

5.5.3 Fast version of Algorithm 13

The $\log k$ bound of Theorem 5.10 is still valid for Algorithm 13 if, instead of investigating all values of ℓ from 2 to k, it considers only the values that can be written as $\lceil k/2^t \rceil$, for $t = 0, \ldots, \lfloor \log k \rfloor$. In Table 5.5 we compare the results of this fast version of the algorithm with those of the full version.

Even considering the overhead of running `Single-Linkage`, which cannot be avoided for both versions of Algorithm 13, we see a reduction of at least 30% in the algorithm's running time when using this fast version. The loss in terms of `MST-Cost`, on the other hand, is less than 10% in the worst scenario, and in 5 of the 10 datasets analyzed both versions return the same clustering.

5.5.4 Distribution of results for `Min-Spacing` and `MST-Cost`

Figures 5.2 to 5.7 show the boxplots for the `Min-Spacing` and the `MST-Cost`, respectively, per dataset and algorithm. Algorithm 13 presents some large variations, when compared to both k-means and Algorithm 12, in terms of `Min-Spacing` (Figures 5.2, 5.3, 5.4) for some datasets; as it is designed to

Table 5.4: Average size of the smallest cluster in a k-clustering, per algorithm and dataset.

	Dimensions		Average size of smallest cluster		
	n	k	Algorithm 12	Algorithm 13	k-means
anuran	7,195	10	264.5	266.7	264.1
avila	20,867	12	85.3	85.4	83.2
collins	1,000	30	7.7	7.7	7.7
digits	1,797	10	96.2	96.2	92.7
letter	20,000	26	192	258.7	191.5
mice	552	8	47.6	47.6	47
newsgroups	18,846	20	188.1	188.2	188.1
pendigits	10,992	10	483.2	476.4	463.8
sensorless	58,509	11	1,842.5	1,725.5	1,655.6
vowel	990	11	57.9	57.9	57.9

Table 5.5: `Min-Spacing`, `MST-Cost` and execution time for Algorithm 13.

	Dimensions		`MST-Cost`		Time (seconds)	
	n	k	Fast	Full	Fast	Full
anuran	7,195	10	1.71	**1.87**	**2.55**	6.77
avila	20,867	12	0.77	**0.81**	**20.47**	62.51
collins	1,000	30	**12.42**	**12.42**	**0.76**	4.48
digits	1,797	10	**178.22**	**178.22**	**0.55**	1.78
letter	20,000	26	5.66	**5.67**	**23.44**	116.44
mice	552	8	**5.66**	**5.66**	**0.27**	0.44
newsgroups	18,846	20	**19**	**19**	**307.58**	440.38
pendigits	10,992	10	215.11	**217.01**	**6.13**	16.02
sensorless	58,509	11	1.34	**1.36**	**274.00**	701.42
vowel	990	11	**4.94**	**4.94**	**0.18**	0.59

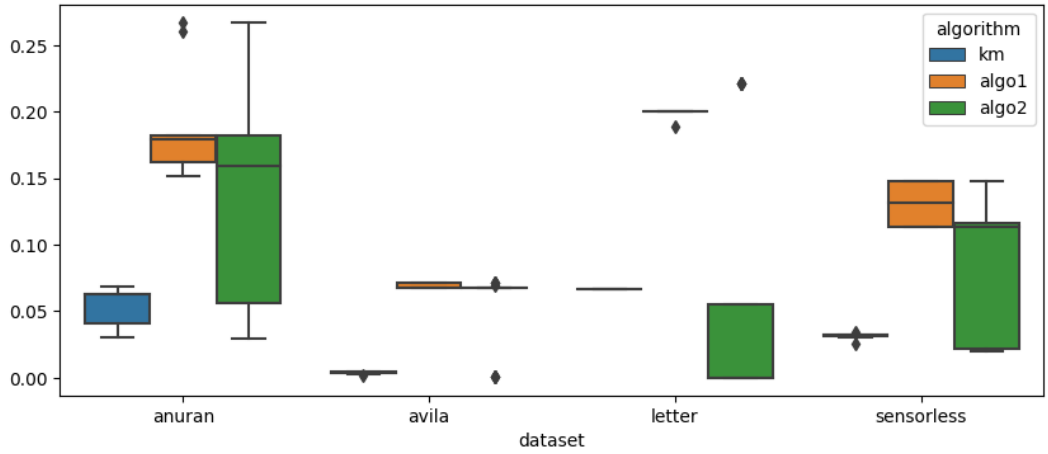


Figure 5.2: Boxplots of `Min-Spacing` per algorithm for the `anuran`, `avila`, `letter`, and `sensorless` datasets.

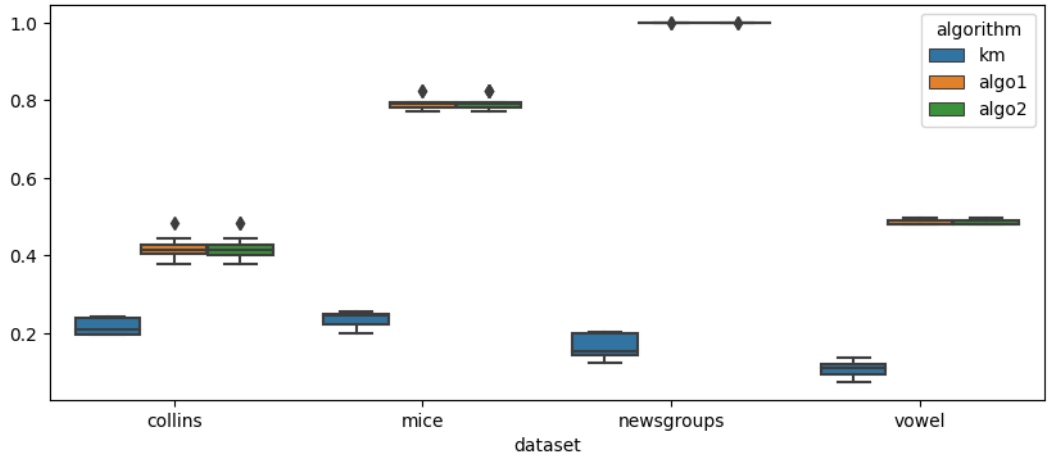


Figure 5.3: Boxplots of `Min-Spacing` per algorithm for the `collins`, `mice`, `newsgroups`, and `vowel` datasets.

maximize the `MST-Cost`, this behavior should not be too concerning. Also in terms of `Min-Spacing`, both algorithms presented in the paper clearly outperform k-means in almost all datasets, even considering the variation in results. The same can be said for `MST-Cost` (Figures 5.5, 5.6, 5.7), in which, additionally, the range of results returned by Algorithm 13 is much more in line with those returned by the other two algorithms.

5.5.5 Trade-off between size of smallest cluster and inter-group separability criteria

In Figures 5.8 and 5.9 we present scatterplots for our 5 smallest datasets showing how the quality of the clusterings generated by Algorithms 12 and 13 (considering, respectively, `Min-Spacing` and `MST-Cost` as criteria) increases

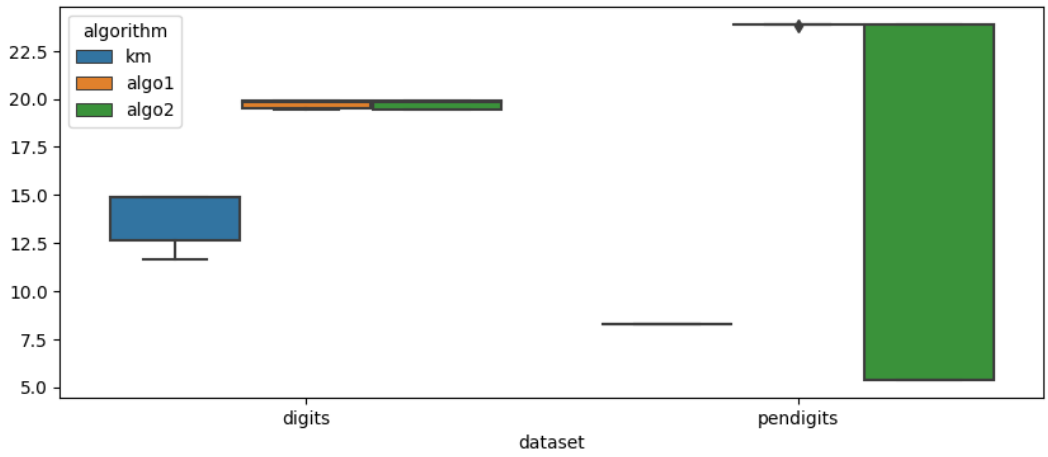


Figure 5.4: Boxplots of `Min-Spacing` per algorithm for the `digits` and `pendigits` datasets.

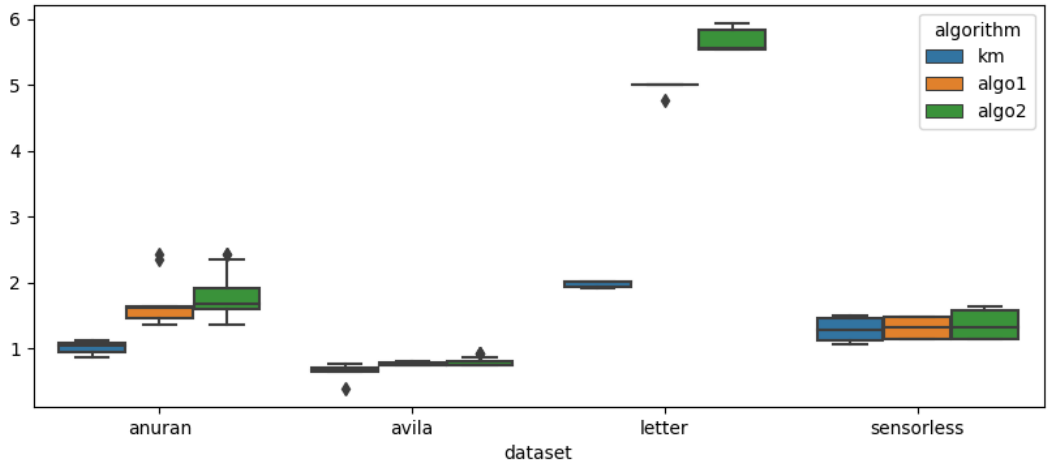


Figure 5.5: Boxplots of `MST-Cost` per algorithm for the `anuran`, `avila`, `letter`, and `sensorless` datasets.

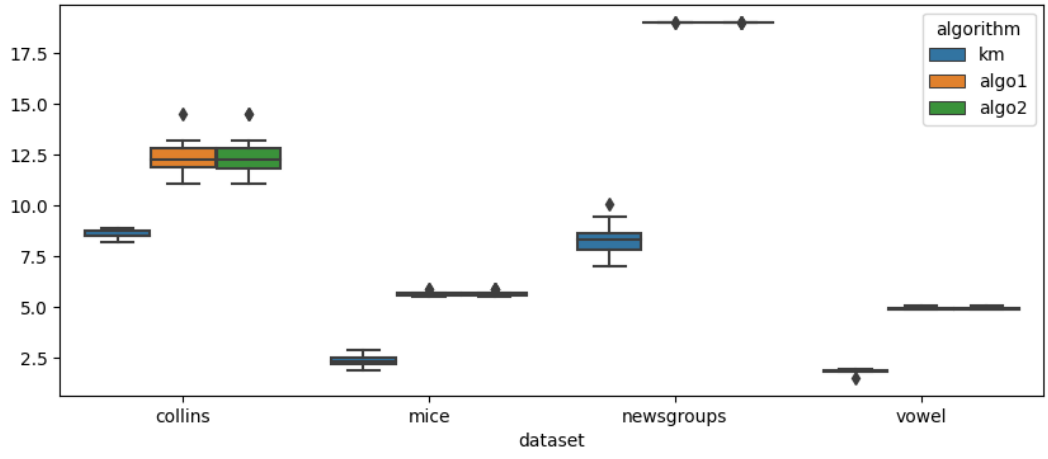


Figure 5.6: Boxplots of `MST-Cost` per algorithm for the `collins`, `mice`, `newsgroups`, and `vowel` datasets.

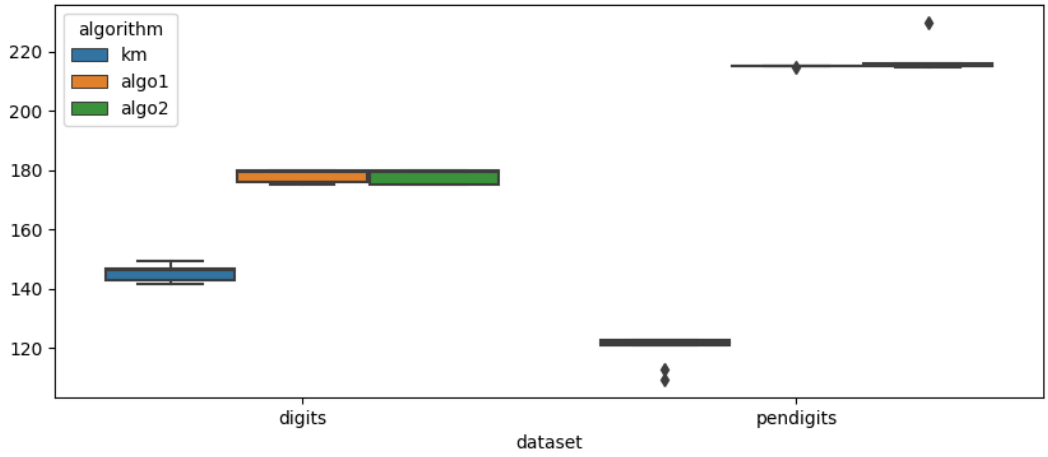


Figure 5.7: Boxplots of `MST-Cost` per algorithm for the `digits` and `pendigits` datasets.

as we allow for clusters of smaller sizes. For all datasets, as expected, allowing for smallest clusters leads to higher `Min-Spacing` and `MST-Cost`. It is still noteworthy that the algorithms presented in this paper can be used not only to find a good partition with a hard limit on the size of the smallest cluster, but also to find the best balance between minimum size and a good separation of clusters.

5.5.6 Effect of randomness on Algorithm 13's results

While Algorithm 12 is fully deterministic, in Algorithm 13 the split of clusters from an ℓ-clustering to turn it into a k-clustering is performed randomly. In practice, however, this does not affect the results of the algorithm.

For each dataset, we ran 10 seeded iterations of Algorithm 13 for each value of L used in the experiments. We then calculate the standard deviation of `MST-Cost` for each value of L. As shown in Table 5.6, for 8 of the datasets analyzed the `MST-Cost` of the clustering returned by Algorithm 13 is always the same for a given value of L; for `letter` and `sensorless`, there is some variation, but it is very small compared to the average `MST-Cost` returned by the algorithm.

5.5.7 Relative k-means cost for Algorithms 12 and 13

In Table 5.7 we present the k-means cost (1-1) for both Algorithm 12 and Algorithm 13 as a multiple of the cost incurred by the k-means algorithm. As expected, since both algorithms were devised for maximizing inter-group criteria, they perform poorly in light of this intra-group loss function — with

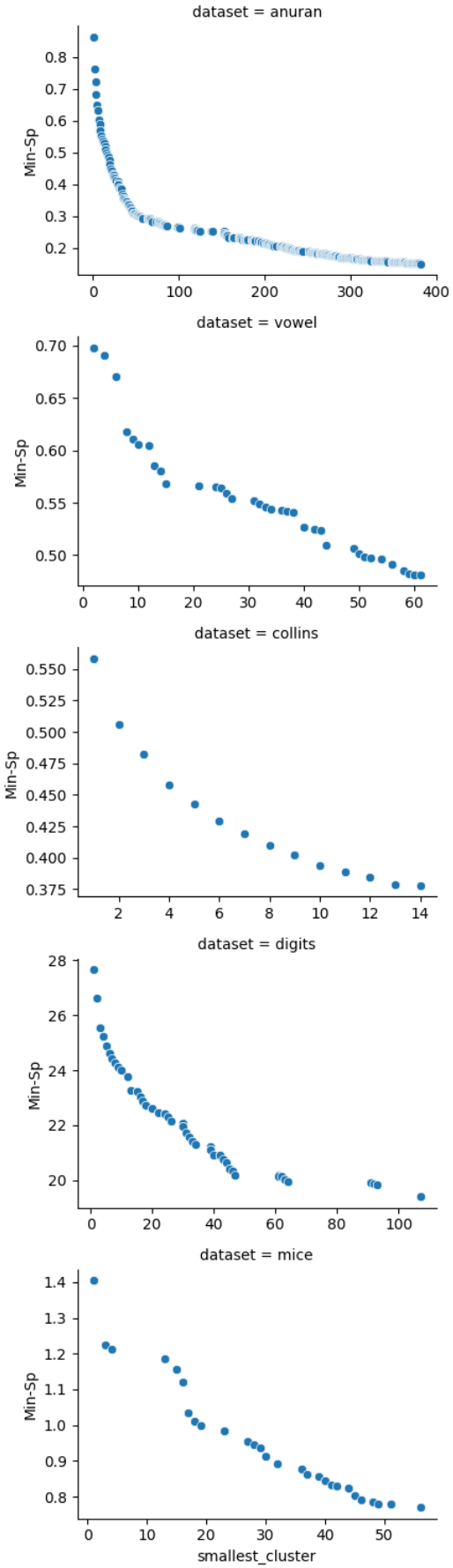


Figure 5.8: Algorithm 12: trade-off between the size of the smallest cluster and `Min-Spacing`.

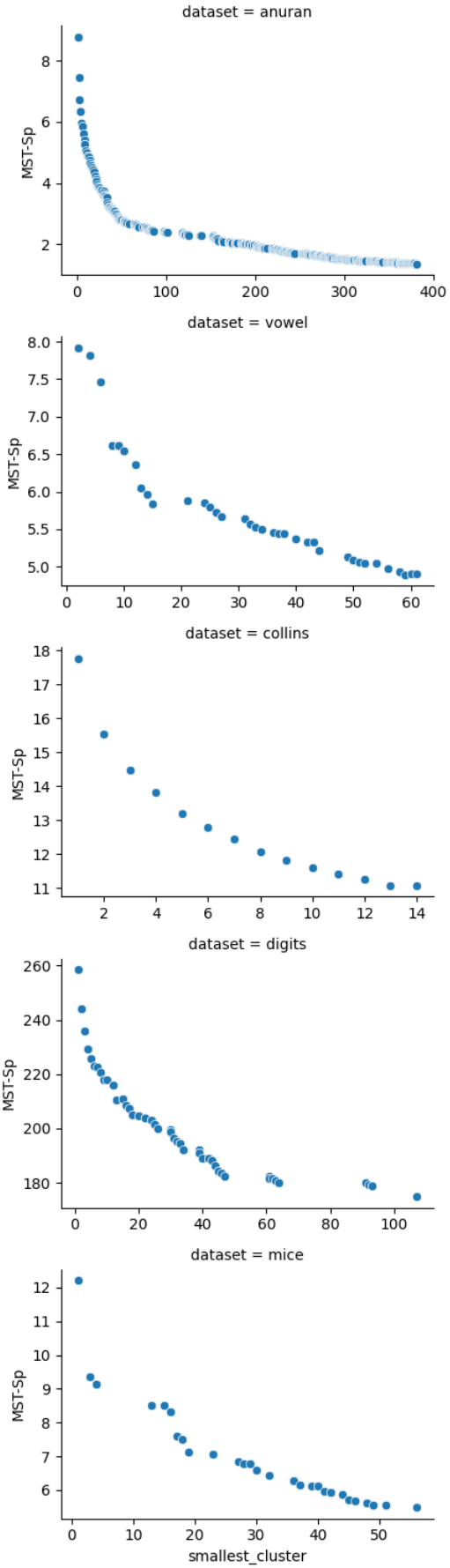


Figure 5.9: Algorithm 13: trade-off between the size of the smallest cluster and `MST-Cost`.

Table 5.6: Standard deviation of `MST-Cost` for Algorithm 13.

	Dimensions		`MST-Cost`	
	n	k	μ	σ
anuran	7,195	10	1.87	-
avila	20,867	12	0.81	-
collins	1,000	30	12.42	-
digits	1,797	10	178.22	-
letter	20,000	26	5.67	0.34
mice	552	8	5.66	-
newsgroups	18,846	20	19	-
pendigits	10,992	10	217.01	-
sensorless	58,509	11	1.96	0.002
vowel	990	11	4.94	-

the sole exception of the `20Newsgroups` dataset, for which both algorithms incur a loss only 5% above that of k-means. Across datasets, the performance of both algorithms is similar for this loss, with only small variations.

Table 5.7: k-means cost (1-1) for Algorithms 12 and 13.

	Dimensions		Loss (relative to k-means)	
	n	k	Algorithm 12	Algorithm 13
anuran	7,195	10	2.50	**2.07**
avila	20,867	12	**2.81**	**2.81**
collins	1,000	30	**2.33**	**2.33**
digits	1,797	10	**1.33**	**1.33**
letter	20,000	26	**2.30**	2.52
mice	552	8	**2.52**	**2.52**
newsgroups	18,846	20	**1.05**	**1.05**
pendigits	10,992	10	2.18	**2.10**
sensorless	58,509	11	**4.32**	5.25
vowel	990	11	**2.07**	**2.07**

6
Conclusions

We analyzed in this thesis two tasks of clustering involving constraints. In the first one, we want to build partitions that are explainable, in the sense that they can be induced by binary decision trees. In the second, we want to build partitions that maximize some form of interclustering distance while maintaining a minimum number of elements per cluster. In the sections below, we briefly present our main conclusions, and some possible avenues of further research, for each of these two topics.

6.1
Explainable clustering via decision trees

We present both theoretical and practical contributions to the task of explainable clustering via decision trees. On the theoretical side, we provide guarantees for the price of explainability of four different clustering problems: k-centers (3-1), k-medians (3-2), k-means (1-1), and minimum spacing (1-2).

We also present three practical algorithms for the explainable k-means problem: `ExGreedy`, `ExShallow`, and `ExBisection`. To analyze their results and compare them to other algorithms in the literature, we evaluate the performance of the algorithm not only in terms of the k-means cost function (1-1) but also in terms of how explainable the decision tree that induces the partition actually is. To do so, we use two measures of explainability: the *weighted average depth* (`WAD`) and the *weighted average explanation size* (`WAES`). `ExShallow`, which explicitly considers these metrics when selecting cuts to build the tree, is shown to have good results (compared to other algorithms in the literature) both in terms of price and of explainability, while `ExBisection` is the first algorithm for the explainable k-means problem that can build an explainable partition "from scratch," i.e., without using an unrestricted partition as a starting point. Notably, we show that all three algorithms tend to have better results in terms of price than `RandomThresholds`, the algorithm that reaches the best theoretical guarantees for the k-medians and k-means problems.

In terms of further theoretical research, (Gupta et al. (2023)) conjectures that `RandomThresholds` may be an $\mathcal{O}(k)$ algorithm for the price of

explainability of k-means, which would close the gap to the lower bound from (Moshkovitz et al. (2020)). Proving this result would be a satisfying closure to the theoretical analysis of `RandomThresholds`. It would be also valuable to try and find theoretical guarantees for algorithms that use different strategies for building explainable clusterings, which have better experimental results than `RandomThresholds`.

Exploring other notions of explainability may also be fruitful. (Izza et al. (2022)) challenges the notion that decision trees are inherently explainable – and mention as a potential problem path redundancy, which we used to build arguably *more* explainable clusters in `ExShallow`. Our idea, building on the concept of *explanation size* from (Feitosa et al. (2022)), is that paths with redundancy (i.e., including more than one cut aligned to the same axis) would lead to shorter explanations, because the cluster in this path would need only one of these redundant cuts to be explained. But it might be the case that the presentation of this explanation as a tree may be suboptimal – and that considering other ways to present explainable partitions may lead to different algorithms, with different approximations in terms of price.

6.2 Inter-clustering problems with a minimum size per cluster

Our first contribution regarding inter-clustering problems is to introduce the notion of the *minimum-spanning-tree cost* (`MST-Cost`), i.e., the cost of a minimum spanning tree of the complete weighted graph induced by a partition of the data in the following way:

- Each node in the graph corresponds to a cluster in the partition.
- Each edge has weight equivalent to the minimum spacing (`Min-Spacing`) between the two clusters represented by the nodes it connects.

We show the importance of this metric for inter-clustering by proving that the `Single-Linkage` algorithm finds a partition that optimizes not only `Min-Spacing`, as is well established in the literature, but also `MST-Cost`, and that the latter implies the former.

Then, focusing on the problem of creating a partition with well-separated clusters with a size constraint, we introduce approximation algorithms for both `Min-Spacing` and `MST-Cost`. For the first objective, our algorithm finds a k-clustering whose smallest cluster has at least $(1 - \epsilon)L$ members and whose `Min-Spacing` is no smaller than the optimal `Min-Spacing` for a k-clustering whose smallest cluster has L elements. For the second one, we find an

$\left(\frac{(1-\epsilon)\rho}{2}, \frac{1}{H_{k-1}}\right)$-approximation, where $\rho := \min\left\{\frac{n/k}{L}, 2\right\}$ and H_{k-1} is the $(k-1)$-th harmonic number.

We also present results on approximation hardness for both tasks. For the problem of maximizing `Min-Spacing` with a minimum size per cluster, we show that, unless $P = NP$, the problem does not admit a $(1, \frac{1}{\alpha})$-approximation, making our algorithm essentially tight. For `MST-Cost`, we show that, unless $P = NP$, there is no $\left(1, \frac{k-2}{k-1} + \frac{1}{\alpha(k-1)}\right)$-approximation for any $\alpha = poly(n)$. In this case, our algorithm is not tight; however, our experiments on 10 different datasets show that in practice the algorithm tends to achieve partitions with much better approximation ratios than our guarantee would suggest.

The experiments also show that our algorithms can generate partitions with much better separation than the k-means algorithm while avoiding small clusters, while `Single-Linkage` would return partitions with a large number of singletons; and that there is a trade-off (that can be "tuned" in our algorithms via the ϵ parameter) between separability and the size of the smallest cluster.

One potential limitation of our proposed algorithms is their usage on massive datasets (in particular for `MST-Cost`), since they execute `Single-Linkage` one or many times. If the distant function between points is explicitly given, then the $\Omega(n^2)$ time spent by `Single-Linkage` is unavoidable. However, if the distances can be calculated from the original dataset, then faster algorithms might be obtained.

The main theoretical question that remains open in our work is whether there exist constant approximation algorithms for the maximization of `MST-Cost`. In addition to addressing this question, interesting directions for future research include handling different inter-cluster measures, as well as other constraints on the structure of clustering.

Bibliography

[Alpadyin & Kaynak, 1998] ALPAYDIN, E.; KAYNAK, C.. **Cascading classifiers**. Kybernetika, 34(4):369–374, 1998.

[Arthur & Vassilvitskii (2006)] ARTHUR, D.; VASSILVITSKII, S.. **K-means++: The advantages of careful seeding**. In: PROCEEDINGS OF THE EIGHTEENTH ANNUAL ACM-SIAM SYMPOSIUM ON DISCRETE ALGORITHMS, SODA '07, p. 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[Arutyunova et al. (2024)] ARUTYUNOVA, A.; GROSSWENDT, A.; RÖGLIN, H.; SCHMIDT, M. ; WARGALLA, J.. **Upper and lower bounds for complete linkage in general metric spaces**. Machine Learning, 113(1):489–518, 2024.

[Bandyapadhyay et al. (2023)] BANDYAPADHYAY, S.; FOMIN, F. V.; GOLOVACH, P. A.; LOCHET, W.; PUROHIT, N. ; SIMONOV, K.. **How to find a good explanation for clustering?** Artificial Intelligence, 322:103948, 2023.

[Bertsimas et al. (2018)] BERTSIMAS, D.; ORFANOUDAKI, A. ; WIBERG, H.. **Interpretable clustering via optimal trees**. arXiv, 2018.

[Blanco et al. (2020)] BLANCO-JUSTICIA, A.; DOMINGO-FERRER, J.; MARTÍNEZ, S. ; SÁNCHEZ, D.. **Machine learning explainability via microaggregation and shallow decision trees**. Knowledge-Based Systems, 194:105532, 2020.

[Bradley et al. (2000)] BRADLEY, P. S.; BENNETT, K. P. ; DEMIRIZ, A.. **Constrained k-means clustering**. Microsoft Research, Redmond, 20(0):0, 2000.

[Bratko (1997)] BRATKO, I.. **Machine learning: Between accuracy and interpretability**. In: Della Riccia, G.; Lenz, H.-J. ; Kruse, R., editors, LEARNING, NETWORKS AND STATISTICS, p. 163–177, Vienna, 1997. Springer Vienna.

[Breiman (2017)] BREIMAN, L.; FRIEDMAN, J. H.; OLSHEN, R. A. ; STONE, C. J.. **Classification and regression trees**. Routledge, 2017.

[Burkart & Huber (2021)] BURKART, N.; HUBER, M. F.. **A survey on the explainability of supervised machine learning**. J. Artif. Intell. Res., 70:245–317, 2021.

[Carlsson & Facundo (2010)] CARLSSON, G. E.; MÉMOLI, F.. **Characterization, stability and convergence of hierarchical clustering methods**. J. Mach. Learn. Res., 11:1425–1470, 2010.

[Caruana et al. (1999)] CARUANA, R.; KANGARLOO, H.; DIONISIO, J. D.; SINHA, U. ; JOHNSON, D.. **Case-based explanation of non-case-based learning methods**. Proc AMIA Symp, p. 212–215, 1999.

[Charikar & Chatziafratis (2017)] CHARIKAR, M.; CHATZIAFRATIS, V.. **Approximate hierarchical clustering via sparsest cut and spreading metrics**. In: Klein, P. N., editor, PROCEEDINGS OF THE TWENTY-EIGHTH ANNUAL ACM-SIAM SYMPOSIUM ON DISCRETE ALGORITHMS, SODA 2017, BARCELONA, SPAIN, HOTEL PORTA FIRA, JANUARY 16-19, p. 841–854. SIAM, 2017.

[Charikar & Hu (2021)] CHARIKAR, M.; HU, L.. **Near-optimal explainable k-means for all dimensions**. In: PROCEEDINGS OF THE 2022 ANNUAL ACM-SIAM SYMPOSIUM ON DISCRETE ALGORITHMS (SODA), p. 2580–2606, 2022.

[Charikar et al. (2002)] CHARIKAR, M.; FAGIN, R.; GURUSWAMI, V.; KLEINBERG, J.; RAGHAVAN, P. ; SAHAI, A.. **Query strategies for priced information**. Journal of Computer and System Sciences, 64(4):785–819, 2002.

[Chen et al. (2019)] CHEN, C.; LI, O.; TAO, D.; BARNETT, A.; RUDIN, C. ; SU, J. K.. **This looks like that: Deep learning for interpretable image recognition**. In: Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. ; Garnett, R., editors, ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, volumen 32. Curran Associates, Inc., 2019.

[Cohen-Addad et al. (2019)] COHEN-ADDAD, V.; KANADE, V.; MALLMANN-TRENN, F. ; MATHIEU, C.. **Hierarchical clustering: Objective functions and algorithms**. J. ACM, 66(4):26:1–26:42, 2019.

[Collobert et al., 2002] COLLOBERT, R.; BENGIO, S. ; BENGIO, Y.. **A parallel mixture of SVMs for very large scale problems**. In: Dietterich,

T.; Becker, S. ; Ghahramani, Z., editors, ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, volumen 14, p. 633–640. MIT Press, 2001.

[Csirik et al. (1992)] CSIRIK, J.; KELLERER, H. ; WOEGINGER, G. J.. **The exact LPT-bound for maximizing the minimum completion time**. Oper. Res. Lett., 11(5):281–287, 1992.

[Daniely et al. (2012)] DANIELY, A.; LINIAL, N. ; SAKS, M.. **Clustering is difficult only when it does not matter**. arXiv preprint arXiv:1205.4891, 2012.

[Dasgupta & Laber (2024)] DASGUPTA, S.; LABER, E.. **New bounds on the cohesion of complete-link and other linkage methods for agglomeration clustering**. Proceedings of the 41st International Conference on Machine Learning (forthcoming), 2024.

[Dasgupta & Long (2005)] DASGUPTA, S.; LONG, P. M.. **Performance guarantees for hierarchical clustering**. Journal of Computer and System Sciences, 70(4):555–569, 2005.

[Dasgupta (2016)] DASGUPTA, S.. **A cost function for similarity-based hierarchical clustering**. In: Wichs, D.; Mansour, Y., editors, PROCEEDINGS OF THE 48TH ANNUAL ACM SIGACT SYMPOSIUM ON THEORY OF COMPUTING, STOC 2016, CAMBRIDGE, MA, USA, JUNE 18-21, 2016, p. 118–127. ACM, 2016.

[Deng et al. (2023)] DENG, C.; GAVVA, S. T.; S., K. C.; PATEL, P. ; SRINIVASAN, A.. **Impossibility of depth reduction in explainable clustering**. arXiv preprint arXiv:2305.02850, 2023.

[De Stefano et al. (2018)] DE STEFANO, C.; MANIACI, M.; FONTANELLA, F. ; SCOTTO DI FRECA, A.. **Reliable writer identification in medieval manuscripts through page layout features: The "Avila" Bible case**. Engineering Applications of Artificial Intelligence, 72:99–110, 2018.

[Doshi-Velez & Kim (2018)] DOSHI-VELEZ, F.; KIM, B.. **Considerations for Evaluation and Generalization in Interpretable Machine Learning**, p. 3–17. Springer International Publishing, Cham, 2018.

[Esfandiari et al. (2021)] ESFANDIARI, H.; MIRROKNI, V. ; NARAYANAN, S.. **Almost tight approximation algorithms for explainable clustering**. In: PROCEEDINGS OF THE 2022 ANNUAL ACM-SIAM SYMPOSIUM ON DISCRETE ALGORITHMS (SODA), p. 2641–2663, 2022.

[Feitosa et al. (2022)] SOUZA, V. F.; CICALESE, F.; LABER, E. ; MOLINARO, M.. **Decision trees with short explainable rules**. In: Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K. ; Oh, A., editors, ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, volumen 35, p. 12365–12379. Curran Associates, Inc., 2022.

[Fisher, 1936] FISHER, R. A.. **The use of multiple measurements in taxonomic problems**. Annals of Eugenics, 7(2):179–188, 1936.

[Fraiman et al. (2013)] FRAIMAN, R.; GHATTAS, B. ; SVARC, M.. **Interpretable clustering using unsupervised binary trees**. Adv. Data Anal. Classif., 7(2):125–145, 2013.

[Frieze & Jerrum (1997)] FRIEZE, A. M.; JERRUM, M.. **Improved approximation algorithms for MAX k-cut and MAX BISECTION**. Algorithmica, 18(1):67–81, 1997.

[Frost et al. (2020)] FROST, N.; MOSHKOVITZ, M. ; RASHTCHIAN, C.. **ExKMC: Expanding explainable k-means clustering**. arXiv, 2020.

[Gamlath et al. (2021)] GAMLATH, B.; JIA, X.; POLAK, A. ; SVENSSON, O.. **Nearly-tight and oblivious algorithms for explainable clustering**. In: Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P. ; Vaughan, J. W., editors, ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, volumen 34, p. 28929–28939. Curran Associates, Inc., 2021.

[Ganganath et al. (2014)] GANGANATH, N.; CHENG, C.-T. ; TSE, C. K.. **Data clustering with cluster size constraints using a modified k-means algorithm**. In: 2014 INTERNATIONAL CONFERENCE ON CYBER-ENABLED DISTRIBUTED COMPUTING AND KNOWLEDGE DISCOVERY, p. 158–161, 2014.

[Garey & Johnson (2011)] GAREY, M. R.; JOHNSON, D. S.. **Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)**. W. H. Freeman, first edition edition, 1979.

[Ghattas et al. (2017)] GHATTAS, B.; MICHEL, P. ; BOYER, L.. **Clustering nominal data using unsupervised binary decision trees: Comparisons with the state of the art methods**. Pattern Recognition, 67:177–185, 2017.

[Gong et al. (2019)] GONG, Z.; ZHONG, P. ; HU, W.. **Diversity in machine learning**. Ieee Access, 7:64323–64350, 2019.

[Gupta et al. (2023)] GUPTA, A.; PITTU, M. R.; SVENSSON, O. ; YUAN, R.. **The price of explainability for clustering**. In: 2023 IEEE 64TH ANNUAL SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE (FOCS), p. 1131–1148, 2023.

[Higuera et al., 2015] HIGUERA, C.; GARDINER, K. J. ; CIOS, K. J.. **Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome**. PLOS ONE, 10(6):1–28, 06 2015.

[Hofmeyr (2020)] HOFMEYR, D. P.. **Connecting spectral clustering to maximum margins and level sets**. The Journal of Machine Learning Research, 21(1):630–664, 2020.

[Hsu & Lin, 2002] HSU, C.-W.; LIN, C.-J.. **A comparison of methods for multiclass support vector machines**. IEEE Transactions on Neural Networks, 13(2):415–425, 2002.

[Hüyük et al. (2021)] HÜYÜK, A.; JARRETT, D.; TEKIN, C. ; VAN DER SCHAAR, M.. **Explaining by imitating: Understanding decisions by interpretable policy learning**. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, 2021.

[Izza et al. (2022)] IZZA, Y.; IGNATIEV, A. ; MARQUES-SILVA, J.. **On tackling explanation redundancy in decision trees**. Journal of Artificial Intelligence Research, 75:261–321, 2022.

[Jain (2010)] JAIN, A. K.. **Data clustering: 50 years beyond k-means**. Pattern Recognition Letters, 31(8):651–666, 2010. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR).

[Jain et al. (1999)] JAIN, A. K.; MURTY, M. N. ; FLYNN, P. J.. **Data clustering: A review**. ACM Comput. Surv., 31(3):264–323, Sept. 1999.

[Kleinberg & Tardos (2006)] KLEINBERG, J. M.; TARDOS, É.. **Algorithm design**. Addison-Wesley, 2006.

[Kleinberg (2002)] KLEINBERG, J. M.. **An impossibility theorem for clustering**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 2002.

[Krizhevsky, 2009] KRIZHEVSKY, A.. **Learning multiple layers of features from tiny images**. Technical report, 2009.

[Laber & Murtinho (2021)] LABER, E.; MURTINHO, L.. **On the price of explainability for some clustering problems**. In: Meila, M.; Zhang, T., editors, PROCEEDINGS OF THE 38TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, volumen 139 de **Proceedings of Machine Learning Research**, p. 5915–5925. PMLR, 18–24 Jul 2021.

[Laber & Murtinho (2023)] LABER, E.; MURTINHO, L.. **Optimization of inter-group criteria for clustering with minimum size constraints**. In: THIRTY-SEVENTH CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 2023.

[Laber (2024)] LABER, E. S.. **The computational complexity of some explainable clustering problems**. Information Processing Letters, 184:106437, 2024.

[Laber et al. (2023)] LABER, E.; MURTINHO, L. ; OLIVEIRA, F.. **Shallow decision trees for explainable k-means clustering**. Pattern Recognition, 137:109239, 2023.

[Leskovec et al. (2020)] LESKOVEC, J.; RAJARAMAN, A. ; ULLMAN, J. D.. **Mining of massive data sets**. Cambridge university press, 2020.

[Lipton (2018)] LIPTON, Z. C.. **The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.** Queue, 16(3):31–57, jun 2018.

[Liu et al. (2005)] LIU, B.; XIA, Y. ; YU, P.. **Clustering via decision tree construction**. In: Chu, W.; Young Lin, T., editors, FOUNDATIONS AND ADVANCES IN DATA MINING, p. 97–124. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

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

[Lundberg et al. (2017)] LUNDBERG, S. M.; LEE, S.-I.. **A unified approach to interpreting model predictions**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, p. 4765–4774, 2017.

[Makarychev & Shan (2021a)] MAKARYCHEV, K.; SHAN, L.. **Near-optimal algorithms for explainable k-medians and k-means**. In: Meila, M.; Zhang, T., editors, PROCEEDINGS OF THE 38TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, volumen 139 de **Proceedings of Machine Learning Research**, p. 7358–7367. PMLR, 18–24 Jul 2021.

[Makarychev & Shan (2021b)] MAKARYCHEV, K.; SHAN, L.. **Explainable k-means: Don't be greedy, plant bigger trees!** In: PROCEEDINGS OF THE 54TH ANNUAL ACM SIGACT SYMPOSIUM ON THEORY OF COMPUTING, STOC 2022, p. 1629–1642, New York, NY, USA, 2022. Association for Computing Machinery.

[Makarychev & Shan (2023)] MAKARYCHEV, K.; SHAN, L.. **Random cuts are optimal for explainable k-medians**. In: Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M. ; Levine, S., editors, ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, volumen 36, p. 66890–66901. Curran Associates, Inc., 2023.

[Malhotra et al. (2022)] MALHOTRA, A.; MITTAL, S.; MAJUMDAR, P.; CHHABRA, S.; THAKRAL, K.; VATSA, M.; SINGH, R.; CHAUDHURY, S.; PUDROD, A. ; AGRAWAL, A.. **Multi-task driven explainable diagnosis of covid-19 using chest x-ray images**. Pattern Recognition, 122:108243, 2022.

[McSherry (2002)] MCSHERRY, D.. **Explanation of attribute relevance in decision-tree induction**. In: Bramer, M.; Coenen, F. ; Preece, A., editors, RESEARCH AND DEVELOPMENT IN INTELLIGENT SYSTEMS XVIII, p. 39–52, London, 2002. Springer London.

[Mokoena et al. (2022)] MOKOENA, T.; CELIK, T. ; MARIVATE, V.. **Why is this an anomaly? explaining anomalies using sequential explanations**. Pattern Recognition, 121:108227, 2022.

[Moseley & Wang (2017)] MOSELEY, B.; WANG, J. R.. **Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search**. In: Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N. ; Garnett, R., editors, ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 30: ANNUAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS 2017, DECEMBER 4-9, 2017, LONG BEACH, CA, USA, p. 3094–3103, 2017.

[Moshkovitz et al. (2020)] MOSHKOVITZ, M.; DASGUPTA, S.; RASHTCHIAN, C. ; FROST, N.. **Explainable k-means and k-medians clustering**. In: III, H. D.; Singh, A., editors, PROCEEDINGS OF THE 37TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, volumen 119 de **Proceedings of Machine Learning Research**, p. 7055–7065. PMLR, 13–18 Jul 2020.

[Murtagh & Contreras (2012)] MURTAGH, F.; CONTRERAS, P.. **Algorithms for hierarchical clustering: an overview**. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2(1):86–97, 2012.

[Murtagh & Contreras (2017)] MURTAGH, F.; CONTRERAS, P.. **Algorithms for hierarchical clustering: an overview, ii**. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 7(6):e1219, 2017.

[Murtagh (1983)] MURTAGH, F.. **A survey of recent advances in hierarchical clustering algorithms**. The computer journal, 26(4):354–359, 1983.

[Pedregosa et al. (2011)] PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. **Scikit-learn: Machine learning in Python**. Journal of Machine Learning Research, 12:2825–2830, 2011.

[Piltaver et al. (2016)] PILTAVER, R.; LUSTREK, M.; GAMS, M. ; MARTINCIC-IPSIC, S.. **What makes classification trees comprehensible?** Expert Syst. Appl., 62:333–346, 2016.

[Rani et al. (2006)] RANI, P.; LIU, C.; SARKAR, N. ; VANMAN, E.. **An empirical study of machine learning techniques for affect recognition in human–robot interaction**. Pattern Analysis and Applications, 9(1):58–69, May 2006.

[Ribeiro et al. (2016)] RIBEIRO, M. T.; SINGH, S. ; GUESTRIN, C.. **" why should i trust you?" explaining the predictions of any classifier**. In: PROCEEDINGS OF THE 22ND ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, p. 1135–1144, 2016.

[Ros & Guillaume (2019)] ROS, F.; GUILLAUME, S.. **A hierarchical clustering algorithm and an improvement of the single linkage criterion to deal with noise**. Expert Systems with Applications, 128:96–108, 2019.

[Roy & Pokutta (2016)] ROY, A.; POKUTTA, S.. **Hierarchical clustering via spreading metrics**. In: Lee, D. D.; Sugiyama, M.; von Luxburg, U.; Guyon, I. ; Garnett, R., editors, ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 29: ANNUAL CONFERENCE ON NEURAL

INFORMATION PROCESSING SYSTEMS 2016, DECEMBER 5-10, 2016, BARCELONA, SPAIN, p. 2316–2324, 2016.

[Rudin (2019)] RUDIN, C.. **Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead**. Nature Machine Intelligence, 1(5):206–215, May 2019.

[Saisubramanian et al. (2020)] SAISUBRAMANIAN, S.; GALHOTRA, S. ; ZILBERSTEIN, S.. **Balancing the tradeoff between clustering value and interpretability**. In: PROCEEDINGS OF THE AAAI/ACM CONFERENCE ON AI, ETHICS, AND SOCIETY, p. 351–357, New York, NY, USA, 2020. Association for Computing Machinery.

[Scikit-Learn (2024)] SCIKIT-LEARN. **Comparing different clustering algorithms on toy datasets**, 2024. Accessed in 2024-05-29.

[Shi et al. (2022)] SHI, C.; FANG, L.; LV, Z. ; ZHAO, M.. **Explainable scale distillation for hyperspectral image classification**. Pattern Recognition, 122:108316, 2022.

[Slack et al. (2023)] SLACK, D.; KRISHNA, S.; LAKKARAJU, H. ; SINGH, S.. **Explaining machine learning models with interactive natural language conversations using TalkToModel**. Nature Machine Intelligence, 5(8):873–883, Aug. 2023.

[Steinbach at al. (2000)] STEINBACH, M.; KARYPIS, G. ; KUMAR, V.. **A comparison of document clustering techniques**. In: KDD WORKSHOP ON TEXT MINING, 2000.

[Strehl & Ghosh (2002)] STREHL, A.; GHOSH, J.. **Cluster ensembles — a knowledge reuse framework for combining multiple partitions**. J. Mach. Learn. Res., 3:583–617, mar 2003.

[Virtanen et al. (2020)] VIRTANEN, P.; GOMMERS, R.; OLIPHANT, T. E.; HABERLAND, M.; REDDY, T.; COURNAPEAU, D.; BUROVSKI, E.; PETERSON, P.; WECKESSER, W.; BRIGHT, J.; VAN DER WALT, S. J.; BRETT, M.; WILSON, J.; MILLMAN, K. J.; MAYOROV, N.; NELSON, A. R. J.; JONES, E.; KERN, R.; LARSON, E.; CAREY, C. J.; POLAT, İ.; FENG, Y.; MOORE, E. W.; VANDERPLAS, J.; LAXALDE, D.; PERKTOLD, J.; CIMRMAN, R.; HENRIKSEN, I.; QUINTERO, E. A.; HARRIS, C. R.; ARCHIBALD, A. M.; RIBEIRO, A. H.; PEDREGOSA, F.; VAN MULBREGT, P.; VIJAYKUMAR, A.; BARDELLI, A. P.; ROTHBERG, A.; HILBOLL, A.; KLOECKNER, A.; SCOPATZ, A.; LEE, A.; ROKEM, A.; WOODS, C. N.;

FULTON, C.; MASSON, C.; HÄGGSTRÖM, C.; FITZGERALD, C.; NICHOLSON, D. A.; HAGEN, D. R.; PASECHNIK, D. V.; OLIVETTI, E.; MARTIN, E.; WIESER, E.; SILVA, F.; LENDERS, F.; WILHELM, F.; YOUNG, G.; PRICE, G. A.; INGOLD, G.-L.; ALLEN, G. E.; LEE, G. R.; AUDREN, H.; PROBST, I.; DIETRICH, J. P.; SILTERRA, J.; WEBBER, J. T.; SLAVIČ, J.; NOTHMAN, J.; BUCHNER, J.; KULICK, J.; SCHÖNBERGER, J. L.; DE MIRANDA CARDOSO, J. V.; REIMER, J.; HARRINGTON, J.; RODRÍGUEZ, J. L. C.; NUNEZ-IGLESIAS, J.; KUCZYNSKI, J.; TRITZ, K.; THOMA, M.; NEWVILLE, M.; KÜMMERER, M.; BOLINGBROKE, M.; TARTRE, M.; PAK, M.; SMITH, N. J.; NOWACZYK, N.; SHEBANOV, N.; PAVLYK, O.; BRODTKORB, P. A.; LEE, P.; MCGIBBON, R. T.; FELDBAUER, R.; LEWIS, S.; TYGIER, S.; SIEVERT, S.; VIGNA, S.; PETERSON, S.; MORE, S.; PUDLIK, T.; OSHIMA, T.; PINGEL, T. J.; ROBITAILLE, T. P.; SPURA, T.; JONES, T. R.; CERA, T.; LESLIE, T.; ZITO, T.; KRAUSS, T.; UPADHYAY, U.; HALCHENKO, Y. O.; VÁZQUEZ-BAEZA, Y. ; 1.0 CONTRIBUTORS, S.. **Scipy 1.0: fundamental algorithms for scientific computing in Python**. Nature Methods, 17(3):261–272, Mar 2020.

[Woeginger (1997)] WOEGINGER, G. J.. **A polynomial-time approximation scheme for maximizing the minimum machine completion time**. Oper. Res. Lett., 20(4):149–154, 1997.

[Yin & Neubig (2022)] YIN, K.; NEUBIG, G.. **Interpreting language models with contrastive explanations**. In: Goldberg, Y.; Kozareva, Z. ; Zhang, Y., editors, PROCEEDINGS OF THE 2022 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, p. 184–198, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics.

[Zahn (1971)] ZAHN, C. T.. **Graph-theoretical methods for detecting and describing gestalt clusters**. IEEE Transactions on computers, 100(1):68–86, 1971.